# A New Architecture for Neural Enhanced Multiobject Tracking

Shaoxiu Wei, Mingchao Liang, and Florian Meyer
University of California San Diego, La Jolla, CA ({shwei,m3liang,flmeyer}@ucsd.edu)

***Abstract*—Multiobject tracking (MOT) is an important task in robotics, autonomous driving, and maritime surveillance. Traditional work on MOT is model-based and aims to establish algorithms in the framework of sequential Bayesian estimation. More recent methods are fully data-driven and rely on the training of neural networks. The two approaches have demonstrated advantages in certain scenarios. In particular, in problems where plenty of labeled data for the training of neural networks is available, data-driven MOT tends to have advantages compared to traditional methods. A natural thought is whether a general and efficient framework can integrate the two approaches. This paper advances a recently introduced hybrid model-based and data-driven method called neural-enhanced belief propagation (NEBP). Compared to existing work on NEBP for MOT, it introduces a novel neural architecture that can improve data association and new object initialization, two critical aspects of MOT. The proposed tracking method is leading the nuScenes LiDAR-only tracking challenge at the time of submission of this paper.**

***Index Terms*—Multiobject tracking, Bayesian framework, neural networks, LiDAR, autonomous driving,**

## I. Introduction

Multiobject tracking (MOT) [1], [2] is an important capability in a variety of applications, including autonomous navigation, applied ocean sciences, and aerospace surveillance. In MOT, the number of objects to be tracked is unknown, and there is measurement-origin uncertainty, i.e., it is unknown which object generated which measurements. Thus, key aspects of MOT methods are data association, object track initialization, and sequential estimation. Traditional model-based MOT methods have evolved from Bayesian filtering theory and perform object track initialization, motion prediction, data association, and measurement updates based on well-established statistical models. Model-based MOT approaches include vector-type methods such as the joint probabilistic data association (JPDA) filter [3], the multiple hypothesis tracker (MHT) [4], and belief propagation (BP) [5]. Recent developments in this Bayesian framework [6] also include set-type methods [7] such as the probability hypothesis density (PHD) filter [8], the labeled multi-Bernoulli (LMB) filter [9], and Poisson multi-Bernoulli (PMB) filters [5], [10], [11]. To obtain tractable algorithms, model-based methods are typically implemented using a Gaussian mixture or particle-based representation of probability density functions (PDFs). Model-based MOT methods have been tailored to specific applications by introducing additional parameters to be estimated and tracked, such as the time-varying detection probabilities [12], [13], maneuvering motion [12], [14], and objects extents [15], [16].

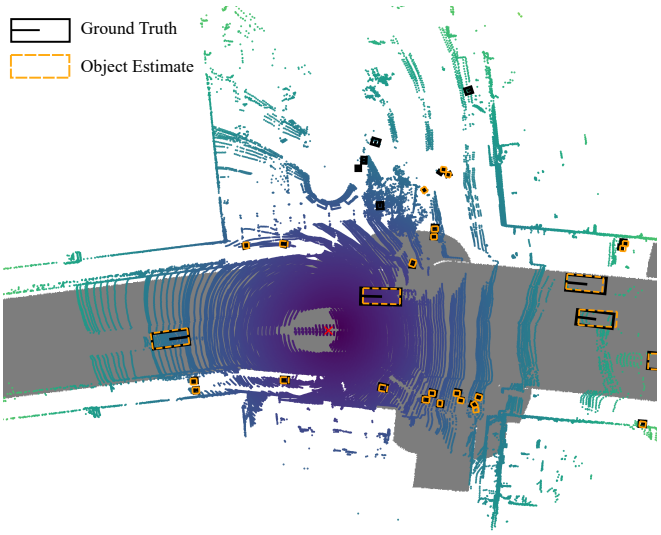


Fig. 1. Considerd multiobject tracking scenario. LiDAR measurements, ground truth, and tracking results of the proposed method. The orange dashed rectangles indicate the object estimates and the black rectangles indicate ground truth..

A recent trend in MOT is using data-driven methods that make use of neural architectures such as the multi-layer perceptrons (MLPs) [17], graph neural networks (GNNs) [18]–[21], and transformers [22]–[24] that are trained with labeled data. Although their structures vary, the primary objective of most data-driven methods is to extract information for data association and object track estimation. For example, [19] used a GNN to perform data association. Due to the expressive power of neural networks [25], data-driven methods are able to extract complex features from data that are difficult to be described by a statistical model.

Recently, neural-enhanced belief propagation (NEBP) for MOT was introduced to combine the advantages of model-based and data-driven approaches [26]. NEBP combines BP-based MOT [5], [27] with a GNN that extracts features from raw sensor data and exchanges information with model-based BP. One significant advantage compared to fully data-driven methods is that established statistical models can still be utilized. At the same time, neural networks enhance traditional models with information learned from raw sensor data. The original NEBP approach can achieve state-of-the-art performance in autonomous driving scenarios [26]. In this paper, we propose NEBP+ for MOT. Compared to the original NEBP approach [26], NEBP+ introduces an improved neural archi-

tecture and makes use of additional types of features. The key insights leveraged by the neural architecture of NEBP+ is that the overall MOT performance can be increased significantly by (i) using precomputed differences of features as input to the neural architecture and (ii) fusing different feature similarities based on learnable weights. The contributions of this paper can be summarized as follows.

1) We propose a novel neural network architecture that enhances the messages of BP-based MOT by learned information and features extracted from raw sensor data.
2) The proposed NEBP+ method is applied to the nuScenes autonomous driving dataset [28] and can achieve state-of-the-art object tracking performance.

At the time of submission of this paper, the proposed tracking method is leading the nuScenes LiDAR-only tracking challenge [28].

## II. System Model and Problem Formulation

In this section, we will introduce the basic assumptions for Bayesian MOT and establish the used notation.

### A. Object States and Transition Model

At time $k$, there are $N_k$ potential objects (POs) $\mathbf{x}_k^i$, $i \in \{1, 2, \ldots, N_k\}$, where $N_k$ is the maximum possible number of objects that have generated a measurement, and the existence of each PO is modeled by a binary random variable $r_k^i \in \{0, 1\}$. A PO $\mathbf{x}_k^i$ exists if and only if $r_k^i = 1$. For simplicity, we use the notation $\mathbf{y}_k^i = [(\mathbf{x}_k^i)^\top \; r_k^i]^\top$ for a PO state that has been augmented by an existence variable. An object detector is applied to sensor data and produce $J_k$ measurements denoted as $\mathbf{z}_k = [(\mathbf{z}_k^1)^\top \; (\mathbf{z}_k^2)^\top \cdots (\mathbf{z}_k^{J_k})^\top]^\top$.

Generally, there are two types of POs:

1) *The Legacy POs*: $\underline{\mathbf{y}}_k^i = [(\underline{\mathbf{x}}_k^i)^\top \; \underline{r}_k^i]^\top, i \in \{1, 2, \ldots, I_k\}$ represent objects that have generated a measurement at a previous time step $k' < k$. The joint state of legacy POs is defined as $\underline{\mathbf{y}}_k = [(\underline{\mathbf{y}}_k^1)^\top \cdots (\underline{\mathbf{y}}_k^{I_k})^\top]^\top$.
2) *The New POs*: $\overline{\mathbf{y}}_k^j = [(\overline{\mathbf{x}}_k^j)^\top \; \overline{r}_k^j]^\top, j \in \{1, 2, \ldots, J_k\}$ represent objects that generate a measurement at time $k$ for the first time. Each measurement $\mathbf{z}_k^j, j \in \{1, 2, \ldots, J_k\}$ will form a new PO $\overline{\mathbf{y}}_k^j$. The joint state of new POs is defined as $\overline{\mathbf{y}}_k = [(\overline{\mathbf{y}}_k^1)^\top \cdots (\overline{\mathbf{y}}_k^{J_k})^\top]^\top$.

When the measurements of the next time step are considered, new POs will become legacy POs. Thus, the number of legacy POs at time $k$ is $I_k = I_{k-1} + J_{k-1}$ and the total number of POs is $N_k = I_k + J_k$. The joint PO states $\mathbf{y}_k$ at time $k$ is defined as $\mathbf{y}_k = [(\underline{\mathbf{y}}_k)^\top \; (\overline{\mathbf{y}}_k)^\top]^\top$.

Given the PO state $\mathbf{y}_{k-1}^i$ at time $k-1$, its transition to the legacy PO $\underline{\mathbf{y}}_k^i$ at time $k$ is assumed to be independent and identical according to a Markovian dynamic model [1]. Then, the transition PDF of the joint PO states $\mathbf{y}_{k-1}$ at time $k-1$ can be expressed as

$$f(\underline{\mathbf{y}}_k|\mathbf{y}_{k-1}) = \prod_{i=1}^{N_{k-1}} f(\underline{\mathbf{x}}_k^i, \underline{r}_k^i|\mathbf{x}_{k-1}^i, r_{k-1}^i).$$

If the PO $i$ does not exist at time $k-1$, i.e., $r_{k-1}^i = 0$, it cannot exist at time $k$. Thus, the state transition PDF can be expressed as

$$f(\underline{\mathbf{x}}_k^i, \underline{r}_k^i|\mathbf{x}_{k-1}^i, 0) = \begin{cases} 0, & \underline{r}_k^i = 1 \\ f_D(\underline{\mathbf{x}}_k^i), & \underline{r}_k^i = 0 \end{cases}$$

where $f_D(\underline{\mathbf{x}}_k^i)$ is an arbitrary "dummy" PDF since states of nonexisting POs are irrelevant. If the PO $i$ exists at time $k-1$, i.e. $r_{k-1}^i = 1$, then it continues to exist with a survival probability $p_s$, i.e.,

$$f(\underline{\mathbf{x}}_k^i, \underline{r}_k^i|\mathbf{x}_{k-1}^i, 1) = \begin{cases} p_s f(\underline{\mathbf{x}}_k^i|\mathbf{x}_{k-1}^i), & \underline{r}_k^i = 1 \\ (1-p_s) f_D(\underline{\mathbf{x}}_k^i), & \underline{r}_k^i = 0. \end{cases}$$

### B. Data Association and Measurement Model

In this paper, we only consider the point object tracking, i.e., each object can generate at most one measurement, and each measurement can originate from at most one object. The former is described by the "object-oriented" data association vector $\mathbf{a}_k = [a_k^1 \; a_k^2 \cdots a_k^{I_k}]^\top$, where each element takes value in $\{0, 1, 2, \ldots, J_k\}$. The latter is described by the "measurement-oriented" data association vector $\mathbf{b}_k = [b_k^1 \; b_k^2 \cdots b_k^{J_k}]^\top$, where each element takes value in $\{0, 1, 2, \ldots, I_k\}$. Under the data association assumption, they are equivalent since one can be determined from the other [5]. The indicator function is used to check whether $\mathbf{a}_k$ and $\mathbf{b}_k$ are consistent, i.e.,

$$\Phi(\mathbf{a}_k, \mathbf{b}_k) = \prod_{i=1}^{I_k} \prod_{j=1}^{J_k} \phi^{i,j}(a_k^i, b_k^j),$$

where

$$\phi^{i,j}(a_k^i, b_k^j) = \begin{cases} 0, & a_k^i = j, b_k^j \neq i \\ & b_k^j = i, a_k^i \neq j \\ 1, & \text{otherwise.} \end{cases}$$

The factor $\Phi(\mathbf{a}_k, \mathbf{b}_k) = 1$ if $\mathbf{a}_k$ and $\mathbf{b}_k$ describe the same valid data association event. For a legacy PO $i$, the state likelihood function is defined as

$$q(\underline{\mathbf{x}}_k^i, 1, a_k^i; \mathbf{z}_k) = \begin{cases} \frac{p_d f(\mathbf{z}_k^j|\underline{\mathbf{x}}_k^i)}{\mu_{\text{FA}} f_{\text{FA}}(\mathbf{z}_k^j)}, & a_k^i = j \\ 1 - p_d, & a_k^i = 0 \end{cases} \tag{1}$$

$$q(\underline{\mathbf{x}}_k^i, 0, a_k^i; \mathbf{z}_k) = 1(a_k^i)$$

where $p_d$ denotes the detection probability and $1(a_k^i)$ denotes the indicator function, i.e. $1(a_k^i) = 1$ if $a_k^i = 0$ and 0 otherwise. If a measurement $j \in \{1, \ldots, J_k\}$ is generated by a PO $i$, it is distributed according to $f(\mathbf{z}_k^j|\underline{\mathbf{x}}_k^i)$. If the measurement $j$ is not generated by any PO, it is considered as the false alarm measurement, which is independent and identically distributed (i.i.d.) according to $f_{\text{FA}}(\cdot)$. The number of false alarm measurements is modeled by a Poisson distribution with mean $\mu_{\text{FA}}$. For new PO $j$, the state likelihood function is defined as

$$v(\overline{\mathbf{x}}_k^i, 1, b_k^j; \mathbf{z}_k^j) = \begin{cases} \frac{\mu_u f_u(\overline{\mathbf{x}}_k^i) f(\mathbf{z}_k^j|\overline{\mathbf{x}}_k^j)}{\mu_{\text{FA}} f_{\text{FA}}(\mathbf{z}_k^j)}, & b_k^j = 0 \\ 0, & b_k^j \neq 0 \end{cases} \tag{2}$$

$$v(\overline{\mathbf{x}}_k^j, 0, b_k^j; \mathbf{z}_k^j) = f_D(\overline{\mathbf{x}}_k^j). \tag{3}$$

New POs are i.i.d. according to $f_u(\overline{\mathbf{x}}_k^i)$, and the number of new POs is modeled by a Poisson distribution with mean $\mu_u$. A detailed derivation of $v(\cdot)$ and $q(\cdot)$ is provided in [5], [26].

### C. Object Declaration and State Estimation

In our Bayesian setting, we declare the existence of POs and estimate their state based on the marginal existence probabilities $p(r_k^i = 1|\mathbf{z}_{1:k})$ and the conditional PDF $f(\mathbf{x}_k^i|r_k^i = 1, \mathbf{z}_{1:k})$. More specifically, a PO is declared to exist if $p(r_k^i = 1|\mathbf{z}_{1:k})$ is above the threshold $T_{\text{dec}}$. For PO $i$ that is declared to exist, a state estimate of $\mathbf{x}_k^i$ is then provided by the minimum-mean-square-error (MMSE) estimator [29], i.e.,

$$\hat{\mathbf{x}}_k^i = \int \mathbf{x}_k^i f(\mathbf{x}_k^i | r_k^i = 1, \mathbf{z}_{1:k}) \mathrm{d}\mathbf{x}_k^i.$$

Note that the existence probability and the conditional PDF are computed as $p(r_k^i = 1|\mathbf{z}_{1:k}) = \int f(\mathbf{x}_k^i, r_k^i = 1|\ \mathbf{z}_{1:k})\mathrm{d}\mathbf{x}_k^i$ and $f(\mathbf{x}_k^i|r_k^i = 1, \mathbf{z}_{1:k}) = f(\mathbf{y}_k^i|\mathbf{z}_{1:k})/p(r_k^i = 1|\mathbf{z}_{1:k})$. Thus, both tasks require computation of the marginal posterior PDF $f(\mathbf{y}_k^i|\mathbf{z}_{1:k}) \triangleq f(\mathbf{x}_k^i, r_k^i|\mathbf{z}_{1:k})$. By applying BP following [5, Sec. VIII-IX], accurate approximations (a.k.a. "beliefs") $\tilde{f}(\mathbf{y}_k^i) \approx f(\mathbf{y}_k^i|\mathbf{z}_{1:k})$ of marginal posterior PDFs are obtained efficiently.

Since a new PO is introduced for each measurement, the number of POs grows with time $k$. Thus, POs whose approximate existence probability is below a threshold $T_{\text{pru}}$ are removed from the state space.

## III. The Methodology

This section introduces the proposed NEBP+ method. It first reviews the factor graph and BP for MOT and then introduces the new neural architecture.

### A. Factor Graph and BP for MOT

With the assumptions in section II and other common assumptions [5], the factorization of the joint posterior PDF $f(\mathbf{y}_{0:k}, \mathbf{a}_{1:k}, \mathbf{b}_{1:k}|\mathbf{z}_{1:k})$ is given as follows

$$\begin{aligned} &f(\mathbf{y}_{0:k}, \mathbf{a}_{1:k}, \mathbf{b}_{1:k}|\mathbf{z}_{1:k}) \\ &\quad\propto \left(\prod_{n'=1}^{N_0} f(\mathbf{y}_0^{n'})\right) \prod_{k'=1}^{k} \left(\prod_{n=1}^{N_{k'-1}} f(\underline{\mathbf{y}}_{k'}^n|\mathbf{y}_{k'-1}^n)\right) \\ &\quad\times \left(\prod_{i=1}^{I_{k'}} q(\underline{\mathbf{y}}_{k'}^i, a_{k'}^i; \mathbf{z}_{k'}) \prod_{j'=1}^{J_{k'}} \phi^{i,j'}(a_{k'}^i, b_{k'}^{j'})\right) \\ &\quad\times \prod_{j=1}^{J_{k'}} v(\overline{\mathbf{y}}_{k'}^j, b_{k'}^j; \mathbf{z}_{k'}^j). \end{aligned}$$

The factorization above provides the basis for a factor graph representation in Fig. 2 [26]. Since the considered factor graph has loops, the order of message passing is given as (i) BP messages are only sent forward in time, (ii) iterative message passing is only performed for data association and at each time step individually. BP for MOT consists of the following three steps.

*1) Prediction:* The prediction step uses the same principle as [5], where the messages passing from factor nodes $f(\underline{\mathbf{y}}_k^i|\mathbf{y}_{k-1}^i)$ to the variable nodes $(\underline{\mathbf{y}}_k^i, a_k^i)$ (See Fig.2) are computed as

$$\begin{aligned} \alpha_k(\underline{\mathbf{x}}_k^i, \underline{r}_k^i) = &\sum_{\underline{r}_{k-1}^i \in \{0,1\}} \int f(\underline{\mathbf{x}}_k^i, \underline{r}_k^i|\mathbf{x}_{k-1}^i, r_{k-1}^i) \\ &\times \tilde{f}(\mathbf{x}_{k-1}^i, r_{k-1}^i) d\mathbf{x}_{k-1}^i, \end{aligned}$$

where $\tilde{f}(\cdot)$ denotes beliefs computed at the last time step [5].

*2) Iterative Probabilistic Data Association:* After the prediction step, an iterative probabilistic data association step is performed for each legacy and new PO. In brief, the objective of this step is to find the global soft data association by exchanging information of local likelihood evaluation among all nodes $(\underline{\mathbf{y}}_k^i, a_k^i)$ and $(\overline{\mathbf{y}}_k^j, b_k^j)$.

At message passing iteration $\ell \in \{1, 2, \ldots, L\}$, the messages $\varphi_k^{i,j,[\ell]}(b_k^j)$ and $\epsilon_k^{j,i,[\ell]}(a_k^i)$ pass from the indicator factor $\phi^{i,j}(a_k^i, b_k^j)$ to the variable node $(\overline{\mathbf{y}}_k^j, b_k^j)$ and $(\underline{\mathbf{y}}_k^i, a_k^i)$. The entire iterative steps are given as

$$\varphi_k^{i,j,[\ell]}(b_k^j) = \sum_{a_k^i=0}^{J_k} \beta_k^i(a_k^i)\phi^{i,j}(a_k^i, b_k^j) \prod_{\substack{j'=1 \\ j' \neq j}}^{J_k} \epsilon_k^{j',i,[\ell]}(a_k^i) \quad (4)$$

$$\epsilon_k^{j,i,[\ell]}(a_k^i) = \sum_{b_k^j=0}^{I_k} \xi_k^j(b_k^j)\phi^{i,j}(a_k^i, b_k^j) \prod_{\substack{i'=1 \\ i' \neq i}}^{I_k} \varphi_k^{i',j,[\ell-1]}(b_k^j) \quad (5)$$

where

$$\beta_k^i(a_k^i) = \sum_{\underline{r}_k^i=\{0,1\}} \int q(\underline{\mathbf{x}}_k^i, \underline{r}_k^i, a_k^i; \mathbf{z}_k) \alpha_k(\underline{\mathbf{x}}_k^i, \underline{r}_k^i) d\underline{\mathbf{x}}_k^i \quad (6)$$

$$\xi_k^j(b_k^j) = \sum_{\overline{r}_k^j=\{0,1\}} \int v(\overline{\mathbf{x}}_k^j, \overline{r}_k^j, b_k^j; \mathbf{z}_k^j) d\overline{\mathbf{x}}_k^j, \quad (7)$$

The iterative message passing process is initialized by setting $\varphi_k^{i,j,[0]}(b_k^j) = 1$ for all $j \in \{1, 2, \ldots, J_k\}$ and $i \in \{1, 2, \ldots, I_k\}$.

*3) Belief Update:* After the last message passing iteration $\ell = L$, the beliefs $\tilde{f}(\underline{\mathbf{y}}_k^i, a_k^i),\ i \in \{1, 2, \ldots, I_k\}$ and $\tilde{f}(\overline{\mathbf{y}}_k^j, b_k^j),\ j \in \{1, 2, \ldots, J_k\}$ are computed according to

$$\tilde{f}(\underline{\mathbf{y}}_k^i, a_k^i) = \frac{1}{\underline{C}_k^i} q(\underline{\mathbf{y}}_k^i, a_k^i; \mathbf{z}_k) \alpha_k(\underline{\mathbf{y}}_k^i) \prod_{j=1}^{J_k} \epsilon^{j,i,[L]}(a_k^i)$$

$$\tilde{f}(\overline{\mathbf{y}}_k^j, b_k^j) = \frac{1}{\overline{C}_k^j} v(\overline{\mathbf{y}}_k^j, b_k^j; \mathbf{z}_k^j) \prod_{i=1}^{I_k} \varphi^{i,j,[L]}(b_k^j)$$

where both notations $\underline{C}_k^i, \overline{C}_k^j$ are normalized constants to make sure that the final beliefs $\tilde{f}(\underline{\mathbf{y}}_k^i, a_k^i), \tilde{f}(\overline{\mathbf{y}}_k^j, b_k^j)$ integrate to unity. By performing marginalization for them to obtain $\tilde{f}(\underline{\mathbf{y}}_k^i), \tilde{f}(\overline{\mathbf{y}}_k^j)$, these beliefs are used for object declaration and estimation [5] and propagated to the next time step.

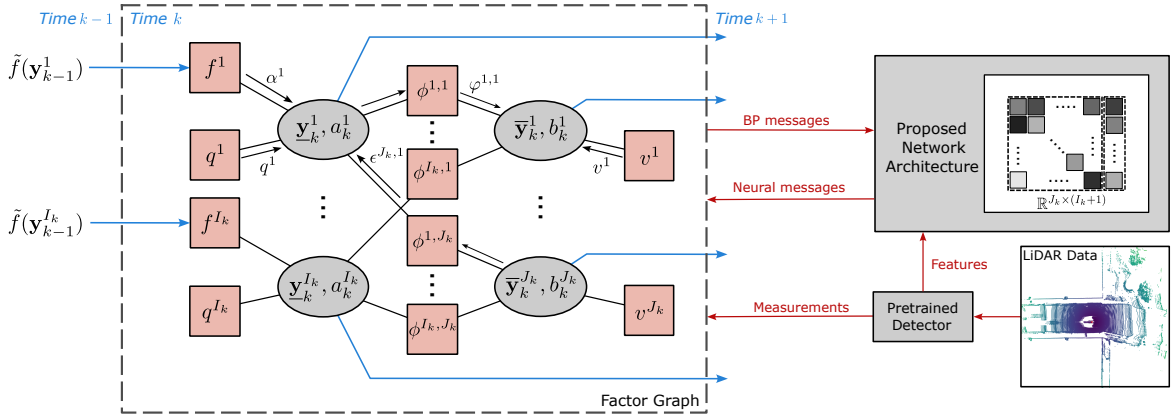


Fig. 2. Block diagram of the proposed NEBP+ approach to MOT. The factor graph, BP messages, and message exchange between the factor graph and neural architecture are shown. The factor graph processes the measurements provided by the detector at the current time step and beliefs from the previous time step. The resulting BP messages are passed to the neural architecture. The neural architecture computes the neural messages based on BP messages and a variety of features provided by the pre-trained detector. (More details on the processing performed by the neural architecture are shown in Fig. 3.) Finally, the neural messages are passed back to enhance the data association process and track initialization. The following shorthand notation is used: $f^i = f(\underline{\mathbf{y}}_k^i|\mathbf{y}_{k-1}^i)$, $q^i = q(\underline{\mathbf{y}}_k^i, a_k^i; \mathbf{z}_k)$, $v^j = v(\overline{\mathbf{y}}_k^j, b_k^j; \mathbf{z}_k)$, $\phi^{i,j} = \phi^{i,j}(a_k^i, b_k^j)$, $\alpha^i = \alpha_k(\underline{\mathbf{x}}_k^i, \underline{r}_k^i)$, $\varphi^{i,j} = \varphi_k^{i,j,[\ell]}(b_k^j)$, and $\epsilon^{j,i} = \epsilon_k^{j,i,[\ell]}(a_k^i)$ .

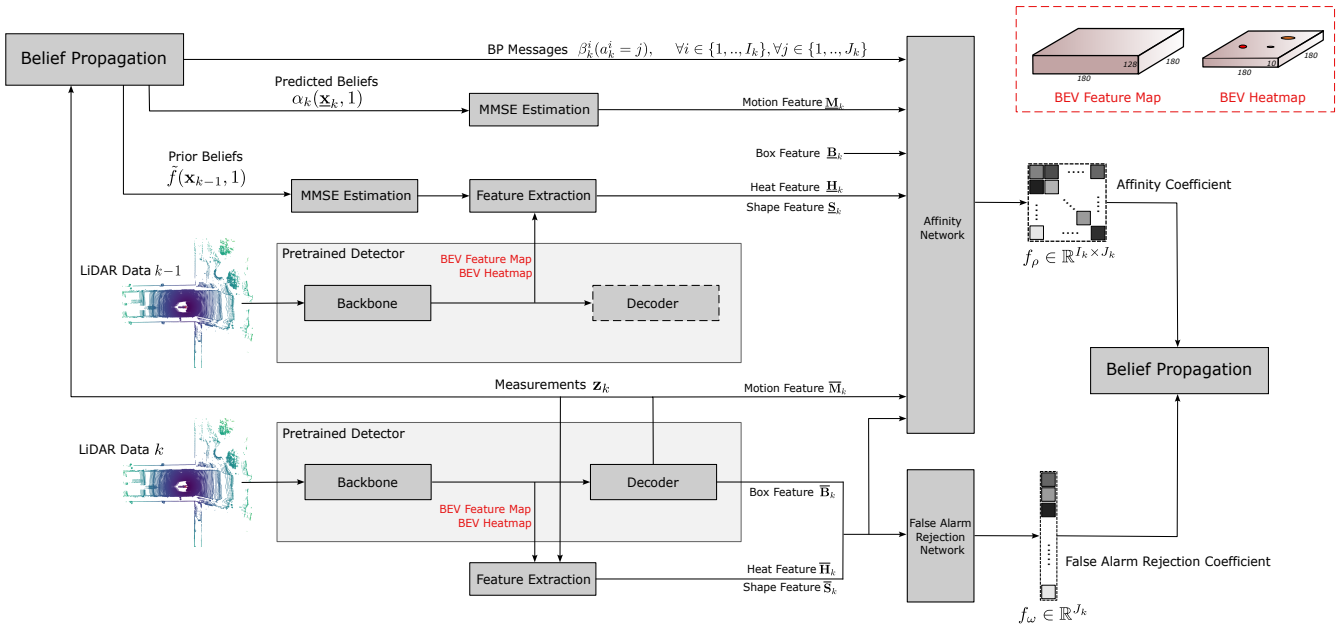


Fig. 3. Neural architecture of the proposed NEBP+ approach. First, motion, box, shape, and heat features are extracted for each measurement and each PO. Next, an affinity coefficient is computed for each pair of PO and measurement, and a false alarm rejection coefficient is computed for each measurement. As discussed in Section III-B, the affinity coefficients are computed based on a linear combination of feature similarity matrixes and BP messages. The false alarm rejection coefficients are calculated based on the box, shape, and heat features extracted for each measurement by the detector.

### *B. The Neural Architecture*

In this section, we will discuss how the proposed neural architecture extracts features from LiDAR data and computes two types of coefficients based on the extracted features. The coefficients are used to enhance data association and object initialization performed by BP.

*1) Feature Extraction:* For each legacy PO $i$ at time $k$, the motion feature $\underline{\mathbf{M}}_k^i \in \mathbb{R}^4$ is obtained by MMSE estimation. The motion feature consists of the objects' 2D position and velocity. The box feature $\underline{\mathbf{B}}_k^i \in \mathbb{R}^3$ consists of the width, height, and length of the bounding box that indicates the size of a PO. The box feature is extracted together with measurements that initiated the PO.

To extract the shape and heat features of POs, the raw data (LiDAR point cloud) is passed into the pre-trained backbone [25] of the FocalFormer3D detector [30]. The pretrained backbone provides a bird's-eye-view (BEV) [31] feature map and heatmap, a unified representation of the LiDAR data in the coordinate system of the sensing vehicle. The BEV feature map is generally a 3D tensor map, where each region of the

map encodes a high dimensional volumetric representation of the object shape information in that region. The BEV heatmap is also a 3D tensor map, which includes the presence and category information of potential objects throughout the entire map [32]. Then, we extract the shape $\underline{\mathbf{S}}_k^i \in \mathbb{R}^{64}$ and heat $\underline{\mathbf{H}}_k^i \in \mathbb{R}^{32}$ features from these two maps respectively through an encoder network (See Fig. 3). The process discussed above can be summarized as

$$(\underline{\mathbf{S}}_k^i, \underline{\mathbf{H}}_k^i) = \mathcal{D}(\mathcal{Z}_{k-1}; \widehat{\mathbf{x}}_{k-1}^i), \tag{8}$$

where $\widehat{\mathbf{x}}_{k-1}^i$ is the MMSE estimation for a legacy PO $i$ at time $k-1$, $\mathcal{Z}_{k-1}$ denotes the raw data at time $k-1$. The network $\mathcal{D}$ consists of the pre-trained backbone of the detector and an additional encoder network. For each measurement indexed by $j$, the motion feature $\overline{\mathbf{M}}_k^j \in \mathbb{R}^4$ is equal to the measurement $\mathbf{z}_k^j$ itself while the box feature $\overline{\mathbf{B}}_k^j \in \mathbb{R}^3$ is also provided by the detector. Similarly, the shape and heat features of each measurement $j$ are obtained as

$$(\overline{\mathbf{S}}_k^j, \overline{\mathbf{H}}_k^j) = \mathcal{D}(\mathcal{Z}_k; \mathbf{z}_k^j) \tag{9}$$

where $\overline{\mathbf{S}}_k^j, \overline{\mathbf{H}}_k^j$ share the same dimension with $\underline{\mathbf{S}}_k^i, \underline{\mathbf{H}}_k^i$, respectively. The 2D position information within $\widehat{\mathbf{x}}_{k-1}^i$ and $\mathbf{z}_k^j$ is used to select the corresponding region in the BEV feature map and heatmap.

*2) The Affinity Coefficient:* By utilizing low-dimensional features (i.e., object position, motion, and size information) and high-dimensional features (i.e., features extracted from the raw LiDAR data), the proposed neural architecture is expected to obtain improved similarity information between objects and measurements. Similarity is represented by "affinity coefficients" that are used to improve data association performed by BP. In particular, the affinity coefficients enhance the association probabilities by taking the information provided by the statistical model, the measurements, and the raw sensor data into account.

Specifically, the affinity coefficient is computed through the "Affinity Network" (see Fig. 3). By comparing the features of each legacy PO and measurement, the network computes four similarity matrices, the elements of which represent their similarity in the motion, box, shape, and heat features, respectively. For each PO $i$ and measurement $j$, the processing steps of the "Affinity Network" are as follows:

- The **motion feature** similarity is obtained as

$$\mathbf{L}_m^{i,j} = \underline{\mathbf{M}}_k^i - \overline{\mathbf{M}}_k^j \tag{10}$$

$$R_m^{i,j} = \mathcal{D}_m(\mathbf{L}_m^{i,j}) \tag{11}$$

  where $\mathbf{L}_m^{i,j} \in \mathbb{R}^4$, $R_m \in \mathbb{R}$ and $\mathcal{D}_m$ is a neural network.
- The **box feature** similarity is given by

$$\mathbf{L}_b^{i,j} = \underline{\mathbf{B}}_k^i - \overline{\mathbf{B}}_k^j \tag{12}$$

$$R_b^{i,j} = \mathcal{D}_b(\mathbf{L}_b^{i,j}) \tag{13}$$

  where $\mathbf{L}_b^{i,j} \in \mathbb{R}^3$, $R_b \in \mathbb{R}$, $\mathcal{D}_b$ is a neural network.
- The **shape and heat feature** similarities are obtained similarly, i.e.,

$$\mathbf{L}_s^{i,j} = \underline{\mathbf{S}}_k^i - \overline{\mathbf{S}}_k^j$$
$$\mathbf{L}_h^{i,j} = \underline{\mathbf{H}}_k^i - \overline{\mathbf{H}}_k^j$$
$$R_s^{i,j} = \mathcal{D}_s(\mathbf{L}_s^{i,j})$$
$$R_h^{i,j} = \mathcal{D}_h(\mathbf{L}_h^{i,j})$$

  where $R_s^{i,j}, R_h^{i,j} \in \mathbb{R}$ represent the shape and heat feature similarities respectively, and $\mathcal{D}_s, \mathcal{D}_h$ are neural networks.
- The **overall similarity matrix** is a weighted sum of the four matrices discussed above and the BP messages. These weights are also provided by a neural network, where we concatenate all distance $\mathbf{L}_m^{i,j}, \mathbf{L}_b^{i,j}, \mathbf{L}_s^{i,j}$, and $\mathbf{L}_h^{i,j}$ calculated above. The weight output $\boldsymbol{\omega}^{i,j}$ is computed as

$$\boldsymbol{\Omega}^{i,j} = [(\mathbf{L}_m^{i,j})^\top \ (\mathbf{L}_b^{i,j})^\top \ (\mathbf{L}_s^{i,j})^\top \ (\mathbf{L}_h^{i,j})^\top]^\top$$
$$\boldsymbol{\omega}^{i,j} = \mathcal{D}_w(\boldsymbol{\Omega}^{i,j})$$

  where $\boldsymbol{\omega}^{i,j} \in (0,1)^5$ and $\boldsymbol{\Omega}^{i,j} \in \mathbb{R}^{103}$. The neural network $\mathcal{D}_w$ is expected to capture the quality of different features to dynamically assign weights to different types of similarities. Then, taking each element in $\boldsymbol{\omega}^{i,j}$ as the corresponding weight for different types of features, the overall pairwise similarity $R^{i,j}$ is obtained by a linear combination, i.e.,

$$R^{i,j} = (\boldsymbol{\omega}^{i,j})^\top \mathbf{R}^{i,j}$$

  where $R^{i,j} = [R_m^{i,j}, R_b^{i,j}, R_s^{i,j}, R_h^{i,j}, R_{\text{BP}}^{i,j}]^\top \in \mathbb{R}^5$ and $R_{\text{BP}}^{i,j} = \beta_k^i(a_k^i = j)$ denotes the BP message in (6). $R_{\text{BP}}^{i,j}$ can be seen as the similarity information provided by the statistical model.

After obtaining the overall similarity matrix, the last step is to apply a learnable nonlinear transformation to it, i.e., we use a neural network $\mathcal{D}_{\text{Aff}}$ to further process $R^{i,j}$

$$f_\rho^{i,j} = \mathcal{D}_{\text{Aff}}(R^{i,j})$$

where $f_\rho^{i,j} \in \mathbb{R}$ denotes the affinity coefficient, which includes the similarity information between a specific PO $i$ and a measurement $j$.

*3) The False Alarm Rejection Coefficient:* To decrease the influence brought by false alarms, the "False Alarm Rejection Network" $\mathcal{D}_{\text{Far}}$ is designed to identify which measurements are likely false alarms (see Fig. 3). Then, for the measurement identified as a potential false alarm, the false alarm distribution in the statistical model used by BP will be locally increased. This false alarm rejection coefficient will help us reduce the probability that the measurement is associated with an existing object and initialize a new object track. The false alarm rejection coefficient $f_\omega^j$ for each measurement $j$ is computed by

$$f_\omega^j = \mathcal{D}_{\text{Far}}(\overline{\mathbf{B}}_k^j, \overline{\mathbf{S}}_k^j, \overline{\mathbf{H}}_k^j),$$

where $f_\omega^j \in (0,1)$ and $\mathcal{D}_{\text{Far}}$ is an MLP with a sigmoid function in the output layer.

### C. Neural Enhanced BP

After obtaining the neural messages $f_\omega^j, f_\rho^{i,j}$, they will be propagated back to factor graph to enhance the BP messages $q(\underline{\mathbf{x}}_k^i, \underline{r}_k^i, a_k^i; \mathbf{z}_k)$ and $v(\overline{\mathbf{x}}_k^j, \overline{r}_k^j, b_k^j; \mathbf{z}_k^j)$. To avoid the codomain of neural messages differing significantly from the value of the above two likelihood functions, as in [26], we calculated the normalized versions of the original BP messages, i.e.,

$$q_s(\underline{\mathbf{x}}_k^i, \underline{r}_k^i, a_k^i; \mathbf{z}_k) = \frac{q(\underline{\mathbf{x}}_k^i, \underline{r}_k^i, a_k^i; \mathbf{z}_k)}{\sum_{a_k^i=0}^{J_k} \sum_{\underline{r}_k^i=\{0,1\}} \int q(\underline{\mathbf{x}}_k^i, \underline{r}_k^i, a_k^i; \mathbf{z}_k) d\underline{\mathbf{x}}_k^i}$$

$$v_s(\overline{\mathbf{x}}_k^j, \overline{r}_k^j, b_k^j; \mathbf{z}_k^j) = \frac{v(\overline{\mathbf{x}}_k^j, \overline{r}_k^j, b_k^j; \mathbf{z}_k^j)}{\sum_{b_k^j=0}^{I_k} \sum_{\overline{r}_k^j=\{0,1\}} \int v(\overline{\mathbf{x}}_k^j, \overline{r}_k^j, b_k^j; \mathbf{z}_k^j) d\overline{\mathbf{x}}_k^j}.$$

Finally, as in [26], the normalized messages are enhanced as

$$\widehat{q}_s(\underline{\mathbf{x}}_k^i, 1, a_k^i = j; \mathbf{z}_k) = f_\omega^j \cdot q_s(\underline{\mathbf{x}}_k^i, 1, a_k^i = j; \mathbf{z}_k) + f_{\text{relu}}(f_\rho^{i,j})$$

$$\widehat{v}_s(\overline{\mathbf{x}}_k^j, 1, b_k^j = 0; \mathbf{z}_k^j) = f_\omega^j \cdot v(\overline{\mathbf{x}}_k^j, 1, b_k^j = 0; \mathbf{z}_k^j)$$

where $f_{\text{relu}}(\cdot)$ denotes rectified linear unit. In other words, the BP messages passed from the likelihood function node are enhanced using the affinity and false alarm rejection coefficients. This enhancement contributes to refining the iterative data association process by recomputing messages $\beta_k^i(a_k^i)$ and $\xi_k^j(b_k^j)$ in (6), (7) using $\widehat{q}_s(\cdot)$ and $\widehat{v}_s(\cdot)$ respectively.

### D. The Loss Function

We use a "weighted" binary cross-entropy loss for the loss function calculation. First, for the affinity coefficient $f_\rho$, the corresponding loss $\mathcal{L}_A$ is calculated by

$$\mathcal{L}_{A1} = -\frac{\sum_{i=1}^{I_k} \sum_{j=1}^{J_k} f_{gt,\rho}^{i,j} \ln(f_\sigma(f_\rho^{i,j}))}{\sum_{i=1}^{I_k} \sum_{j=1}^{J_k} f_{gt,\rho}^{i,j}} \tag{14}$$

$$\mathcal{L}_{A2} = -\frac{\sum_{i=1}^{I_k} \sum_{j=1}^{J_k} (1 - f_{gt,\rho}^{i,j}) \ln(1 - f_\sigma(f_\rho^{i,j}))}{\sum_{i=1}^{I_k} \sum_{j=1}^{J_k} (1 - f_{gt,\rho}^{i,j})} \tag{15}$$

$$\mathcal{L}_A = \mathcal{L}_{A1} + \mathcal{L}_{A2}, \tag{16}$$

where $f_\sigma$ denotes the sigmoid function and $f_{gt,\rho}^{i,j}$ denotes the ground truth data association result, i.e., if the PO $i$ is indeed matched with measurement $j$, then $f_{gt,\rho}^{i,j} = 1$ and otherwise $f_{gt,\rho}^{i,j} = 0$. Please refer to [26] for detailed steps on obtaining the ground truth. $\mathcal{L}_{A1}$ represents the accuracy of estimating the correct data association, while $\mathcal{L}_{A2}$ represents the accuracy of estimating no association. The reason for using the normalization term in (14) and (15) is to deal with the data imbalance problem, i.e., it is equally important whether a PO $i$ is associated with a measurement $j$ or not, regardless of the number of POs and measurements. Compared to [26], the proposed loss function emphasizes the significance of correct measurement-to-objects associations.

For the false alarm rejection coefficient $f_\omega$, the corresponding loss function is given as

$$\mathcal{L}_{F1} = -\frac{\sum_{j=1}^{J_k} f_{gt,\omega}^j \ln(f_\omega^j)}{\sum_{j=1}^{J_k} f_{gt,\omega}^j} \tag{17}$$

$$\mathcal{L}_{F2} = -u \cdot \frac{\sum_{j=1}^{J_k} (1 - f_{gt,\omega}^j) \ln(1 - f_\omega^j)}{\sum_{j=1}^{J_k} (1 - f_{gt,\omega}^j)} \tag{18}$$

$$\mathcal{L}_F = \mathcal{L}_{F1} + \mathcal{L}_{F2} \tag{19}$$

where $f_{gt,\omega}^j$ represents the ground truth label for each measurement [26] and $u \in [0, 1]$ is a tuning parameter. $\mathcal{L}_{F1}$ represents our classification accuracy for true measurements. $\mathcal{L}_{F2}$ represents our classification accuracy for false alarms. The tuning parameter is introduced because missing an object is usually more harmful than having a false alarm.

## IV. Numerical Analysis

The section presents results of nuScenes LiDAR autonomous driving dataset [28] to validate our method. This dataset consists of 1000 autonomous driving scenes and seven object classes. We use the official split of the dataset, where there are 700 scenes for training, 150 for validation, and 150 for testing. Each scene lasts roughly 20 seconds and contains keyframes sampled at 2Hz. The recently introduced FocalFormer3D detector provides measurements for MOT [30] that have been pre-processed using the non-maximum suppression (NMS) technique [33]. We use pre-trained versions of the FocalFormer3D detector and its backbone.

### A. Implementation Details

We use the particle-based implementation of BP-based MOT, where the number of particles is set to $10^4$. The state of a PO $\mathbf{x}_k^i \in \mathbb{R}^6$ consists of its 2D position, velocity, and acceleration. A constant-acceleration motion model models the object dynamics. The measurement $\mathbf{z}_k^j \in \mathbb{R}^4$ consists of the 2D position and velocity. Measurements are modeled using a linear Gaussian measurement model. The corresponding likelihood function is used for compution of $q(\underline{\mathbf{x}}_k^i, 1, a_k^i; \mathbf{z}_k)$ and $v(\overline{\mathbf{x}}_k^i, 1, b_k^j; \mathbf{z}_k^j)$ in (1) and (2). We also use 3D bounding box information and the detection score $s_k^j \in (0, 1]$ provided by the pre-trained detector. In particular, for each new PO, the corresponding bounding box is stored and used as a box feature in the affinity network (see Fig. 3). Together with the probability of existence, the detection score is used to compute the final AMOTA score for each PO (see [26] for details).

The region of interest (ROI) is given by $[P_x - 54, P_x + 54] \times [P_y - 54, P_y + 54]$, where $P_x, P_y$ represent the 2D positions of the ego vehicle. The prior PDF of false alarms $f_{\text{FA}}(\cdot)$ and newly detected objects $f_u(\cdot)$ are uniformly distributed over the ROI. All other parameters used in the proposed NEBP+ method are estimated from the training data as in [26] or set as in [26].

TABLE I
PERFORMANCE OF NUSCENES TEST SET

| Method | Modalities | **AMOTA↑** | AMOTP↓ |
|---|---|---|---|
| Poly-MOT [34] | LiDAR+Camera | 75.4 | 42.2 |
| CAMO-MOT [35] | LiDAR+Camera | 75.3 | 47.2 |
| **NEBP+** | LiDAR | 74.6 | 49.8 |
| BEVFusion [36] | LiDAR+Camera | 74.1 | 40.2 |
| MSMDFusion [37] | LiDAR+Camera | 74.0 | 54.9 |
| FocalFormer3D-F [30] | LiDAR+Camera | 73.9 | 51.4 |
| 3DMOTFormer [22] | LiDAR+Camera | 72.5 | 53.9 |
| FocalFormer3D [30] | LiDAR | 71.5 | 54.9 |
| VoxelNeXt [38] | LiDAR | 71.0 | 51.1 |

TABLE II
PERFORMANCE OF NUSCENES VALIDATION SET

| Method | **AMOTA↑** | AMOTP↓ |
|---|---|---|
| BP [5] | 73.3 | 50.8 |
| NEBP [26] | 74.3 | 54.8 |
| NEBP+ (Only Affinity Coefficient with Motion + BP) | 73.9 | 51.5 |
| NEBP+ (Only Affinity Coefficient with Shape + Heat) | 74.2 | 51.9 |
| NEBP+ (Only Affinity Coefficient) | 75.1 | 54.3 |
| NEBP+ (Only False Alarm Rejection Coefficient) | 74.2 | 55.0 |
| **NEBP+ (Proposed)** | 75.3 | 50.9 |

The encoder network within the neural architecture $\mathcal{D}$ (cf. (8) and (9)) consists of two convolution layers followed by an MLP, which is used to extract heat and shape features. The neural networks $\mathcal{D}_m, \mathcal{D}_b, \mathcal{D}_s, \mathcal{D}_h, \mathcal{D}_\omega, \mathcal{D}_{\text{Aff}}, \mathcal{D}_{\text{Far}}$ are all MLPs with leaky ReLU activation functions in the hidden layers.

### B. Performance Evaluation

*1) Overall Performance:* Two primary metrics for evaluating nuScenes are Average Multi-Object Tracking Accuracy (AMOTA) and Average Multi-Object Tracking Precision (AMOTP) [28]. AMOTA includes errors such as false alarms, missed objects, and identity switches, while AMOTP consists of position errors between ground truth and estimated tracks. Details about these metrics are provided in [28].

The performance of the proposed NEBP+ method applied to nuScenes test data is shown in Tables I. It outperforms all LiDAR-only reference methods in terms of both AMOTA and AMOTP performance. The improved performance of NEBP+ can be explained by the fact that it combines the well-established statistical model for MOT [3]–[5] with learned information provided by the neural architecture.

*2) Ablation Study:* Next, we present results based on the nuScenes validation data. We compare the performance of (i) traditional BP-based MOT, (ii) NEBP proposed in [26] that makes use of the loss functions in (14)–(19), (iii) NEBP+ that only uses motion features and BP messages as the input of the affinity network, (iv) NEBP+ that only uses shape and heat feature as the input of the affinity network. (v) NEBP+ that only uses the affinity coefficient, and (vi) NEBP+ that only uses the false alarm rejection coefficient. AMOTA and AMOTP results are shown in Table II.

We have the following observations based on Table II. (i) The proposed NEBP+ approach yields the largest improvement in AMOTA performance compared to BP. (ii) While NEBP [26] has improved AMOTA performance compared to BP, this comes at the cost of a reduced AMOTP performance. NEBP+, however, has significantly improved AMOTA performance compared to BP while yielding essentially identical AMOTP performance. (iii) In the considered scenario, performance improvements related to the affinity coefficient are much greater than those related to the false alarm rejection coefficient. (iv) Although using only a subset of features increases tracking performance compared to the traditional BP, the best result is achieved when all types of information are leveraged. (v) Even if only low-dimensional information (i.e., motion features and BP messages) is used as input for neural networks, this leads to improved tracking performance. Observation (v) can have two potential explanations. The Gaussian likelihood function and corresponding measurement variance used by model-based BP may not accurately describe the underlying data-generating process. The neural architecture may also have captured complex relationships across motion features that cannot be described by the traditional MOT model used by BP.

## V. CONCLUSION

In this paper, we introduce NEBP+ for MOT. NEBP+ enhances both data association and track initialization of model-based BP by leveraging information learned from raw LiDAR data. Contrary to the original NEBP for MOT method, NEBP+ is based on an improved neural architecture that uses precomputed differences of features as input to the neural architecture and fuses different feature similarities based on learnable weights. Evaluation results based on the nuScenes autonomous driving dataset demonstrate the state-of-the-art performance of NEBP+. The significant performance improvements of NEBP+ in the nuScenes autonomous driving challenge emphasize the superiority of combining well-established statistical models and neural architectures. As the framework of factor graphs and BP, NEBP+ is very flexible and can thus potentially be extended to a variety of further applications ranging from multipath-aided SLAM [39], [40] to marine mammal tracking [41], [42] Another venue for future research is the development of neural-enhanced track-before-detect methods [43].

## VI. ACKNOWLEDGEMENT

This work was supported by the National Science Foundation (NSF) under CAREER Award No. 2146261.

## REFERENCES

[1] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and Data Fusion: A Handbook of Algorithms*. Storrs, CT, USA: Yaakov Bar-Shalom, 2011.
[2] S. Blackman, *Multiple Target Tracking with Radar Applications*. Norwood, MA: Artech House, 1986.
[3] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. San Diego, CA: Academic, 1998.
[4] S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 19, no. 1, pp. 5–18, Jan. 2004.

[5] F. Meyer, T. Kropfreiter, J. L. Williams, R. Lau, F. Hlawatsch, P. Braca, and M. Z. Win, "Message passing algorithms for scalable multitarget tracking," *Proc. IEEE*, vol. 106, no. 2, pp. 221–259, 2018.

[6] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[7] R. Mahler, *Advances in Statistical Multisource-Multitarget Information Fusion*. Norwood, MA, USA: Artech House, 2014.

[8] B.-N. Vo and W.-K. Ma, "The Gaussian mixture probability hypothesis density filter," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4091–4104, Nov. 2006.

[9] S. Reuter, B.-T. Vo, B.-N. Vo, and K. Dietmayer, "The labeled multi-Bernoulli filter," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3246–3260, 2014.

[10] J. L. Williams, "Marginal multi-Bernoulli filters: RFS derivation of MHT, JIPDA, and association-based member," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 51, no. 3, pp. 1664–1687, 2015.

[11] A. F. García-Fernández, J. L. Williams, K. Granström, and L. Svensson, "Poisson multi-Bernoulli mixture filter: Direct derivation and implementation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 4, pp. 1883–1901, Aug. 2018.

[12] G. Soldi, F. Meyer, P. Braca, and F. Hlawatsch, "Self-tuning algorithms for multisensor-multitarget tracking using belief propagation," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 3922–3937, May. 2019.

[13] S. Wei, B. Zhang, and W. Yi, "Trajectory PHD and CPHD filters with unknown detection profile," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, p. 8042–8058, 2022.

[14] R. Gan and S. Godsill, "$\alpha$ -stable Lévy state-space models for manoeuvring object tracking," in *Proc. FUSION-20*, Jul. 2020, pp. 1–7.

[15] K. Granström, M. Baum, and S. Reuter, "Extended object tracking: Introduction, overview and applications," *J. Adv. Inf. Fusion*, vol. 12, no. 2, pp. 139–174, Dec. 2017.

[16] F. Meyer and J. L. Williams, "Scalable detection and tracking of geometric extended objects," *IEEE Trans. Signal Process.*, vol. 69, pp. 6283 – 6298, Oct. 2021.

[17] T. Sadjadpour, J. Li, R. Ambrus, and J. Bohg, "ShaSTA: Modeling shape and spatio-temporal affinities for 3D multi-object tracking," *IEEE Robot. Autom. Lett.*, Oct. 2023.

[18] X. Weng, Y. Wang, Y. Man, and K. M. Kitani, "GNN3DMOT: Graph neural network for 3D multi-object tracking with 2D-3D multi-feature learning," in *Proc. IEEE/CVF CVPR-20*, Jun. 2020, pp. 6498–6507.

[19] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *Proc. IEEE/CVF CVPR-20*, Jun. 2020, pp. 6246–6256.

[20] M. Büchner and A. Valada, "3D multi-object tracking using graph neural networks with cross-edge modality attention," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9707–9714, Jul. 2022.

[21] J.-N. Zaech, A. Liniger, D. Dai, M. Danelljan, and L. Van Gool, "Learnable online graph representations for 3D multi-object tracking," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5103–5110, Jan. 2022.

[22] S. Ding, E. Rehder, L. Schneider, M. Cordts, and J. Gall, "3DMOTFormer: Graph transformer for online 3D multi-object tracking," in *Proc. ICCV-23*, Oct. 2023, pp. 9750–9760.

[23] J. Pinto, G. Hess, Y. Xia, H. Wymeersch, and L. Svensson, "Transformer-based multi-object smoothing with decoupled data association and smoothing," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2312.17261

[24] T. Meinhardt, A. Kirillov, L. Leal-Taixé, and C. Feichtenhofer, "TrackFormer: Multi-object tracking with transformers," in *Proc. IEEE/CVF CVPR-22*, Jun. 2022, pp. 8834–8844.

[25] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF CVPR-18*, Jun. 2018, pp. 4490–4499.

[26] M. Liang and F. Meyer, "Neural enhanced belief propagation for multiobject tracking," *IEEE Trans. Signal Process.*, vol. 72, pp. 15–30, Sep. 2024.

[27] F. Meyer, P. Braca, P. Willett, and F. Hlawatsch, "A scalable algorithm for tracking an unknown number of targets using multiple sensors," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3478–3493, 2017.

[28] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF CVPR-20*, May. 2020, pp. 11 618–11 628.

[29] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ: Prentice-Hall, 1993.

[30] Y. Chen, Z. Yu, Y. Chen, S. Lan, A. Anandkumar, J. Jia, and J. Alvarez, "FocalFormer3D : Focusing on hard instance for 3D object detection," in *Proc. ICCV-23*, Oct. 2023, pp. 8360–8371.

[31] H. Li, C. Sima, J. Dai, W. Wang, L. Lu, H. Wang, J. Zeng, Z. Li, J. Yang, H. Deng, H. Tian, E. Xie, J. Xie, L. Chen, T. Li, Y. Li, Y. Gao, X. Jia, S. Liu, J. Shi, D. Lin, and Y. Qiao, "Delving into the devils of bird's-eye-view perception: A review, evaluation and recipe," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 4, pp. 2151–2170, 2024.

[32] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, Oct. 2018.

[33] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proc. ICPR-06*, vol. 3, Aug. 2006, pp. 850–855.

[34] X. Li, T. Xie, D. Liu, J. Gao, K. Dai, Z. Jiang, L. Zhao, and K. Wang, "Poly-MOT: A polyhedral framework for 3D multi-object tracking," in *Proc. IROS-23*, Oct. 2023, pp. 9391–9398.

[35] L. Wang, X. Zhang, W. Qin, X. Li, J. Gao, L. Yang, Z. Li, J. Li, L. Zhu, H. Wang, and H. Liu, "CAMO-MOT: Combined appearance-motion optimization for 3D multi-object tracking with camera-lidar fusion," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 11 981–11 996, Jun. 2023.

[36] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus, and S. Han, "BEVFusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *Proc. ICRA-23*, Jun. 2023, pp. 2774–2781.

[37] Y. Jiao, Z. Jie, S. Chen, J. Chen, L. Ma, and Y.-G. Jiang, "MSMDFusion: Fusing lidar and camera at multiple scales with multi-depth seeds for 3D object detection," in *Proc. IEEE/CVF CVPR-23*, Jun. 2023, pp. 21 643–21 652.

[38] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "VoxelNeXt: Fully sparse voxelnet for 3D object detection and tracking," in *Proc. IEEE/CVF CVPR-23*, Jun. 2023, pp. 21 674–21 683.

[39] E. Leitinger, F. Meyer, F. Hlawatsch, K. Witrisal, F. Tufvesson, and M. Z. Win, "A belief propagation algorithm for multipath-based SLAM," *IEEE Wirel. Commun.*, vol. 18, no. 12, pp. 5613–5629, 2019.

[40] E. Leitinger, A. Venus, B. Teague, and F. Meyer, "Data fusion for multipath-based SLAM: Combining information from multiple propagation paths," *IEEE Trans. Signal Process.*, vol. 71, pp. 4011–4028, 2023.

[41] J. Jang, F. Meyer, E. R. Snyder, S. M. Wiggins, S. Baumann-Pickering, and J. A. Hildebrand, "Bayesian detection and tracking of odontocetes in 3-D from their echolocation clicks," *J. Acoust. Soc.*, vol. 153, no. 5, pp. 2690–2705, 5 2023.

[42] W. Zhang and F. Meyer, "Multisensor multiobject tracking with improved sampling efficiency," *IEEE Trans. Signal Process.*, vol. 72, pp. 2036–2053, 2024.

[43] M. Liang, T. Kropfreiter, and F. Meyer, "A BP method for track-before-detect," *IEEE Signal Process. Lett.*, vol. 30, pp. 1137–1141, 2023.