GSLAMOT: A Tracklet and Query Graph-based Simultaneous Locating, Mapping, and Multiple Object Tracking System

Shuo Wang
School of Information, Renmin
University of China
Beijing, China
shuowang18@ruc.edu.cn

Yongcai Wang*
School of Information, Renmin
University of China
Beijing, China
ycw@ruc.edu.cn

Zhimin Xu
School of Information, Renmin
University of China
Beijing, China
xuzhimin0124@ruc.edu.cn

Yongyu Guo
School of Information, Renmin
University of China
Beijing, China
yongyuguo@ruc.edu.cn

Wanting Li
School of Information, Renmin
University of China
Beijing, China
lindalee@ruc.edu.cn

Zhe Huang
School of Information, Renmin
University of China
Beijing, China
2021000892@ruc.edu.cn

Xuewei Bai
School of Information, Renmin
University of China
Beijing, China
bai_xuewei@ruc.edu.cn

Deying Li
School of Information, Renmin
University of China
Beijing, China
deyingli@ruc.edu.cn

Abstract

For interacting with mobile objects in unfamiliar environments, simultaneously locating, mapping, and tracking the 3D poses of multiple objects are crucially required. This paper proposes a Tracklet Graph and Query Graph-based framework, i.e., GSLAMOT, to address this challenge. GSLAMOT utilizes camera and LiDAR multimodal information as inputs and divides the representation of the dynamic scene into a semantic map for representing the static environment, a trajectory of the ego-agent, and an online maintained Tracklet Graph (TG) for tracking and predicting the 3D poses of the detected mobile objects. A Query Graph (QG) is constructed in each frame by object detection to query and update TG. For accurate object association, a Multi-criteria Star Graph Association (MSGA) method is proposed to find matched objects between the detections in QG and the predicted tracklets in TG. Then, an Object-centric Graph Optimization (OGO) method is proposed to simultaneously optimize the TG, the semantic map, and the agent trajectory. It triangulates the detected objects into the map to enrich the map's semantic information. We address the efficiency issues to handle the three tightly coupled tasks in parallel. Experiments are conducted on KITTI, Waymo, and an emulated Traffic Congestion dataset that highlights challenging scenarios. Experiments show that GSLAMOT enables accurate crowded object tracking while

conducting SLAM accurately in challenging scenarios, demonstrating more excellent performances than the state-of-the-art methods. The code and dataset is at https://gslamot.github.io.

CCS Concepts

- **Computer systems organization** → **Robotic autonomy**; • **Computing methodologies** → *Simulation evaluation*.

Keywords

localization, 3D tracking, optimization, graph matching

1 Introduction

Conducting self-locating, mapping, i.e., SLAM, and multiple object 3D pose tracking (3D MOT) simultaneously using multimodel information is a fundamental requirement for dynamic scene perception, such as object tracking in autonomous driving, unmanned aerial vehicles (UAVs), and robotics human-machine collaboration [5, 16, 17, 19, 24, 41].

Multiple factors work collectively making this problem very challenging. (1) SLAM and 3D MOT need to be conducted concurrently and depend on each other. SLAM relies on object detection to eliminate the impacts of the dynamic objects for accurately tracking the agent poses and for static mapping, whereas 3D MOT relies on the accurate pose of the agent for triangulating the 3D poses of the mobile objects; (2) The concurrent movements of both the agent and the dynamic objects pose challenges for locating and object

tracking; (3) Errors in possible object detection algorithms and sensor noise also contribute to inaccuracies in localizing surrounding objects; (4) Factors such as occlusions and high-speed movement also introduce difficulty in object matching, leading to locating and tracking errors.

Existing works in 3D multiple object tracking (3D MOT)[27, 37, 49] often assume that the ego-motion is known and free from noise or the sensor is fixed in the world frame. However, in real-world applications, it is general that both the agent and surrounding objects are in motion, requiring concurrent SLAM and 3D MOT. Furthermore, due to motion, the agent may produce inconsistent observations of the same object in different frames, leading to inaccuracies of the bounding boxes (Figure 3) and the difficulty in object matching across frames. Object-Oriented SLAM (OOSLAM) [3, 30, 34, 51] is another closely related area, but OOSLAM primarily focuses on estimating the ego-motion trajectory. It does not explicitly address the optimization of the detailed object 3D trajectories. But many applications do require real-time tracking of the 3D object poses[11, 20, 23].

To address the above challenges, this paper presents GSLAMOT, a graph matching and graph optimization based system for conducting SLAM and 3D MOT simultaneously, which takes stereo images and LiDAR point cloud sequences as input. To the best of our knowledge, it is the first work that outputs ego trajectory and object trajectories concurrently and accurately, even in challenging scenarios with crowded and highly dynamic objects. In particular, GSLAMOT represents the dynamic scene by a combination of *(1) a semantic map representing the environment, (2) the ego-agent trajectory, and (3) an online maintained Tracklet Graph representing the 3D MOT trajectories.* TG tracks the 3D poses of multiple objects and can predict their poses at time t. A Query Graph (QG) is constructed in each frame based on object detection to query the predicted object poses. For accurate object association, a novel Multi-criteria Star Graph Association (MSGA) method is proposed for robust association to deal with the matching challenges in dynamic, congested, and noisy environments. MSGA evaluates neighborhood consistency, spatial consistency, and shape consistency between TG and QG, significantly improving multi-object tracking compared to using only spatial features.

After finding associations, an Object-centric Graph Optimization (OGO) method is proposed to simultaneously optimize the TG, the map, and the agent trajectory. We divide the optimization into two parts: (1) a real-time Object-Centric Optimization Window (OCOW) and (2) a long-term Object-Ego Fusion Window (OEFW). In OCOW, a two-stage optimization strategy is adopted. Then, the well-optimized ego-motion, environmental points, and tracklets will be fused in the OEFW in a tightly coupled manner. Experiments show the OGO achieves better convergence speed (Figure 4) and accuracy than the ego-centric methods.

In system implementation, to enable the localization, 3D MOT, and semantic mapping to run concurrently and efficiently, We employ multi-thread parallel computing, executing modules such as visual front-end, mapping, detection, feature extraction, and tracking concurrently across threads. The main contributions of this work are as follows:

- We present GSLAMOT, which utilizes multimodel information, including stereo images and LiDAR point clouds, as inputs, and uses graph matching and graph optimization to conduct SLAM and 3D MOT simultaneously even in challenging scenarios.
- Multi-criteria Star Graph Association (MSGA) is proposed to match QG and the predicted TG.
- Object-Centric Graph Optimization (OGO) is proposed to estimate tracklets' poses. Object-Ego Fusion Window (OEFW) is proposed to jointly optimize the object poses and the ego-motion poses in a long-term sliding window.
- We exploit parallel threads to address the concurrency requirements in implementing the GSLAMOT system. Experiments demonstrate that GSLAMOT exhibits better real-time performance than other OOSLAM systems.
- We have also created a Traffic Congestion dataset for object-level scene understanding using the Carla simulator[10]. The dataset encompasses a wide range of maps and has a varying number of dynamic and static objects, providing a valuable resource for research and development in this field.

2 Related Work

The most closely related works are reviewed, which are categorized into two areas: Object-Oriented SLAM (OOSLAM) and 3D Multiple Object Tracking (3D MOT).

2.1 Object-oriented SLAM

Object-oriented SLAM (OOSLAM), as an important branch in the research of object-level scene understanding, aims to extract, model, and track the static and dynamic objects in the environment[1]. Early OOSLAM systems[30, 33] typically maintain a database of objects. These systems utilize RGB-D cameras as inputs and employ ICP losses along with pose graph optimization to solve for the camera and object poses. However, this model-based approach requires prepared object models in advance and cannot exhibit robust scalability to unknown scenes. Recent advancements in OOSLAM have increasingly focused on improving object feature extraction and modeling. In QuadricSLAM[26], objects are modeled as ellipsoids or quadric geometry, enabling more effective optimization of shape and scale. CubeSLAM[39] aligns the detection bounding boxes with the image edges to enhance object proposals. DSP-SLAM[34] leverages the signed distance function (SDF) to reconstruct the objects after detection. The learned shape embedding serve as priors during the reconstruction, and the shapes of the objects are jointly optimized in the bundle adjustment process. In MOTSLAM[44], Zhang et al. apply three neural networks to extract 2D bounding boxes, 3D bounding boxes, and semantic segmentation, respectively. While the multi-network approach can extract accurate information about an object from different scales, it does demand large computational resources. DynaSLAM II[2] employs instance semantic segmentation and tracks ORB features on dynamic objects. Similarly, VDO-SLAM[45] leverages instance semantic segmentation, utilizing dense optical flow to maximize the number of tracked points on moving objects.

However, most of OOSLAM systems only utilize objects as landmarks for observation, lacking a complete multi-object tracking
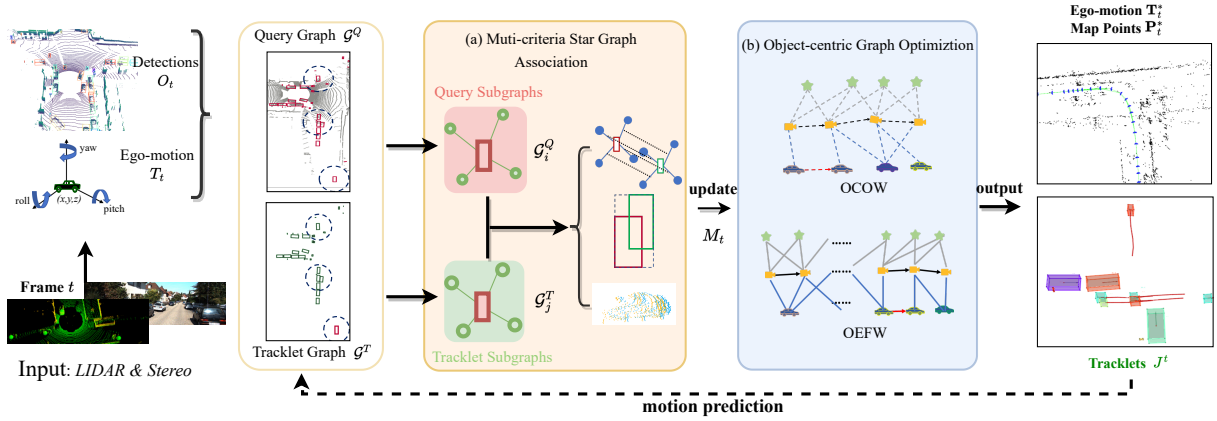
Figure 1: Our system processes LiDAR point clouds and stereo images as inputs. The 3D detection algorithm extracts detection boxes from the point cloud, while the visual odometry front-end obtains the initial ego-motion pose. In the world frame, we construct query and tracklet graphs for detections and tracklets, respectively, and use the MSGA algorithm for association and tracking. Ego-motion, map points, and tracklets are optimized in OGO. Tracklet states for the next frame are estimated using a motion model and participate in the subsequent tracking.

process and failing to output trajectories for surrounding objects. For the noises in object detection and camera pose tracking, object tracking remains a challenging problem in SLAM settings, especially in object-crowded environments.

2.2 3D Multiple Object Tracking

2D multi-object tracking, which involves tracking object bounding boxes in the image plane, has been extensively studied [43, 46–48]. 3D multi-object tracking remains a challenging problem due to the involvement of spatial motion and the 3D appearance of the objects [28, 31].

The framework of 3D MOT is mainly divided into tracking by detection and learning-based algorithms. The former mainly uses Kalman filtering to estimate objects and then relies on specific indicators such as Intersection of Union (IoU)[37], Mahalabnobis distance[8], or Generalized IoU[27] for association.

For learning-based algorithms, many works model 3D MOT as GNN, representing the association information of objects by predicting edges[4, 38, 42]. More relevant to this paper is the approach based on geometric information. PolarMOT[18] only uses 3D boxes as inputs to GNN to learn the geometric features of objects. BOTT[49] relies on self-attention to represent global context information and associate boxes.

Existing 3D MOT approaches generally work either in the ego-motion frame or under the assumption of known ego-motion. However, SLAM and 3D MOT are specially required in unknown environments or in the presence of ego-motion noise. Therefore, our GSLAMOT presents a practical approach.

3 Approach Overview

The proposed GSLAMOT framework is shown in Figure 1. We consider the agent is conducting self-locating, environment mapping, and multiple object 3D pose tracking in an unfamiliar environment. The agent is equipped with a stereo camera that captures RGB

image pairs, and a 3D LiDAR. The stereo images are processed by the visual odometry (VO) system and the LiDAR point clouds are used for 3D object detection[21, 22].

For the t-th frame, the sensor input is denoted as Ω_t. The 3D detector provides the object detection results $\mathbf{O}_t$. The VO front-end outputs the current ego-motion pose T_t. The detections $\mathbf{O}_t$ are transformed into the world coordinate frame by $\mathbf{Q}_t = T_t \mathbf{O}_t$, and the Query Graph $\mathcal{G}^Q$ is constructed.

$$\mathbf{O_t} = f_{detector}(\Omega_t) \tag{1}$$

$$T_t = f_{odometry}(\Omega_t, T_{t-1,...}) \tag{2}$$

$$\mathcal{G}^Q = f_{qconstruct}(\mathbf{Q}_t) \tag{3}$$

In processing this new frame, the active tracklets $\mathcal{J}_{t-1}$ are predicted to time t based on motion model, denoted as $\mathcal{J}_t^{pred}$. Then a tracklet graph $\mathcal{G}^T$ is constructed based on $\mathcal{J}_t^{pred}$. Subsequently, we employ Multi-criteria Star Graph Association (MSGA) to match between $\mathcal{G}^Q$ and $\mathcal{G}^T$ to obtain the matching relationship $\mathcal{M}_t$.

$$\mathcal{J}_t^{pred} = f_{motion}(\mathcal{J}_{t-1}) \tag{4}$$

$$\mathcal{G}^T = f_{tconstruct}(\mathcal{J}_t^{pred}) \tag{5}$$

$$\mathcal{M}_t = f_{MSGA}(\mathcal{G}^Q, \mathcal{G}^T) \tag{6}$$

Then, We employ Object-centric Graph Optimization (OGO) to optimize the ego-motion T_t, tracklets $\mathcal{J}_t$, and the map points P_t. OGO operates within a window size w, incorporating Object-Centric Optimization Window (OCOW) and Object-Ego Fusion Window (OEFW). The optimized results are used to update the Tracklet Graphs.

$$T_t, \mathcal{J}_t, P_t = f_{OCOW}(T_t, \mathcal{M}_t, \mathcal{J}_t^{pred}, \mathbf{Q}_t) \tag{7}$$

$$T_{t-w:t}^*, \mathcal{J}_{t-w:t}^*, P_{t-w:t}^* = f_{OEFW}(T_{t-w:t}, \mathcal{M}_{t-w:t}, \mathcal{J}_{t-w:t}, \mathbf{Q}_{t-w:t}) \tag{8}$$

We use Pointpillars[21] for LiDAR-based 3D object detection, and use ORB-SLAM3[6] for visual odometry front-end. Their results, i.e., $\mathbf{O}_t$ and T_t are inputs of following steps.

4 Multi-criteria Star Graph Association

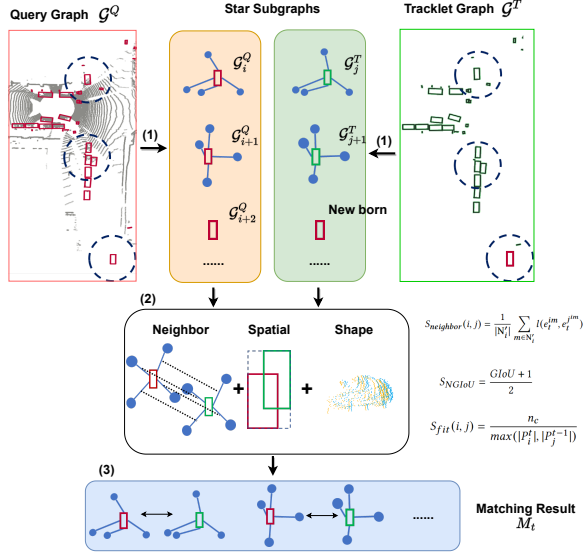We firstly introduction the construction of TG and QG, and the Multi-criteria Star Graph Association (MSGA).

Figure 2: Multi-criteria Star Graph Association.

4.1 Query Graph and Tracklet Graph

At frame t, we select the active tracklets from frame $t - w$ to frame $t - 1$ as $\mathcal{J}_{t-1}$, where w is the OGO window size (detailed in Section 5). The critical task at this stage is to correctly associate the objects in $\mathbf{O}_t$ with those in $\mathcal{J}_{t-1}$. The association is challenging for the following reasons: (1) the agent and the objects are moving; (2) the objects may be crowded; and (3) the detections may be noisy.

Thus, we firstly predict the poses of the tracklets, i.e., $\mathcal{J}_t^{pred} = f_{motion}(\mathcal{J}_{t-1})$, where f_{motion} is based on Kalman Filter[27] using the objects' estimated motion velocities and historical trajectories. Then we construct a graph for both the detections $\mathbf{O}_t$ and the predicted poses of the tracklets, named Query Graph (QG) and Tracklet Graph (TG) respectively.

In particular, in constructing QG, we firstly transform the poses of the detected 3D boxes to the world frame, obtaining $\mathbf{Q}_t$. Then for each detection i in $\mathbf{Q}_t$, the nearest K detections within distance L are chosen as i's neighbor set $\mathbf{N}_i$. We build edges between each detection and its neighbors. This converts the detection set $\mathbf{Q}_t$ to a graph $\mathcal{G}^Q$. Each detection i also forms a local star-graph $\mathcal{G}_i^Q = (\mathcal{V}_i, \mathcal{E}_i)$, where $\mathcal{V}_i = \{i\} \cup \mathbf{N}_i$, and $\mathcal{E}_i$ includes the edges from i to each element in $\mathbf{N}_i$.

In constructing TG, for each j in $\mathcal{J}_t^{pred}$, the nearest K predicted object poses within distance L are chosen as j's neighbor set $\mathbf{N}_j$. We also build edges between each tracklet and its neighbors, and converts $\mathcal{J}_t^{pred}$ to a graph $\mathcal{G}^T$. Each predicted object j also forms

a local star-graph $\mathcal{G}_j^T = (\mathcal{V}_j, \mathcal{E}_j)$, where $\mathcal{V}_j = \{j\} \cup \mathbf{N}_j$, and $\mathcal{E}_j$ includes the edges from j to its neighbors.

4.2 Multi-criteria Consistency

The purpose of association is to determine whether a detection $i \in Q^t$ and a tracklet $j \in \mathcal{J}_t^{pred}$ are corresponding to the same object in the environment. We access this by multi-criteria of $\mathcal{G}_i^Q$ and $\mathcal{G}_j^T$. Two subgraphs with similar structures and features should exhibit a high graph feature consistency[25]. We particularly consider consistency evaluation from three aspects: (1) *neighborhood consistency*, which can be considered as edge similarity; (2) *spatial consistency*, evaluated by Normalized Generalized Intersection over Union (NGIoU)[29] and (3) *shape consistency*, evaluated by Iterative Closest Point (ICP)[15] between the point clouds in the object bounding boxes.

4.2.1 Neighborhood Consistency. In detection i's local star graph, the edge e^{im} between vertex i and $m \in \mathbf{N}_i$ can be represented by a relative pose transformation from i to the neighbor m, denoted by $\mathbf{T}_t^{im}$. If $i \in \mathcal{G}_i^Q$ and $j \in \mathcal{G}_j^T$ are consistent, the local neighborhood edges of i and j should be highly consistent. For example, if vertex m in $\mathcal{G}_i^Q$ is corresponding to vertex n in $\mathcal{G}_j^T$, $\mathbf{T}_t^{jn}$ should be consistent with $\mathbf{T}_t^{im}$. So we evaluate the neighborhood consistency between e_t^{im} and e_t^{jn} by:

$$l(e_t^{im}, e_t^{jn}) = exp(-\|\mathbf{T}_t^{im}(\mathbf{T}_t^{jn})^{-1} - \mathbf{I}\|_F) \qquad (9)$$

where $\mathbf{T}_t^{im}$ is the transformation of e_t^{im} and $(\mathbf{T}_t^{jn})^{-1}$ is the inverse transformation of e_t^{jn}. $\| \cdot \|_F$ represents the Frobenius norm.

However, at this stage, we have not yet completed the object association, meaning that for the query and tracklet graphs, the correspondence of leaf vertices (m and n in Equation (9)) is unknown. This presents a classic chicken-and-egg problem. We propose a **Greedy Neighbor Strategy** to address this issue. For query local star graph $\mathcal{G}_i^Q$ and tracklet local star graph $\mathcal{G}_j^T$. We firstly assume their central vertices represent the same object, then if the edge e^{im} in $\mathcal{G}_i^Q$ exhibits the highest consistency (Equation 9) with the edge e^{jn} in $\mathcal{G}_j^T$, e^{im} and e^{jn} are considered the corresponding edges.

If the highest consistency edge pairs $\mathcal{G}_i^Q$ and $\mathcal{G}_j^T$ do not actually correspond to each other, we still select the edge in $\mathcal{G}_j^T$ with the highest consistency for each edge in $\mathcal{G}_i^Q$, even if their consistency is low. This low consistency can contribute to discrimination in multi-criteria star graph matching (Section 4.3). The overall neighborhood consistency score between $\mathcal{G}_i^Q$ and $\mathcal{G}_j^T$ is therefore:

$$S_{neighbor}(i, j) = \frac{1}{|\mathbf{N}_i'|} \sum_{m \in \mathbf{N}_i'} l(e_t^{im}, e_t^{j^{im}}) \qquad (10)$$

where $\mathbf{N}_i'$ contains the leaf vertices in $\mathcal{G}_i^Q$ that have corresponding vertices in $\mathcal{G}_j^T$. $e_t^{j^{im}}$ represents the corresponding edge of e_t^{im} in $\mathcal{G}_j^T$, which is obtained from the Greedy Neighbor Strategy.

4.2.2 Spatial Consistency. Traditional 3D object tracking typically relies on spatial metrics based on Intersection over Union (IoU)[37]

or inter-distance[8]. However, traditional methods based on inter-distance or IoU between the current detection and tracklets may fail when the distance or IoU values exceed thresholds. This is generally when the agent or objects move at high speed or when there are detection errors. Furthermore, IoU cannot handle cases where two bounding boxes do not overlap, leading to redundant objects in the tracking results.

To address the issues above, we utilize Normalized Generalized Intersection over Union (NGIoU) as our spatial consistency metric. For two centering vertices i and j corresponding to two star graphs (for instance, detection i and tracklet j), we suppose their 3D bounding boxes are B_i and B_j respectively. Then the value of GIoU is calculated as follows:

$$GIoU(i, j) = \frac{|B_i \cap B_j|}{|B_i \cup B_j|} - \frac{|B_i \cup B_j| - |B_i \cap B_j|}{|B_i \cup B_j|} \quad (11)$$

We further normalize GIoU to ensure its range lies within $[0, 1]$:

$$S_{NGIoU} = \frac{GIoU + 1}{2} \quad (12)$$

Experimental results in Section 7 demonstrate that this metric enhances object tracking performances in challenging scenarios.

4.2.3 Shape Consistency. Traditional 3D object tracking methods [8, 37] rarely leverage semantic information within the bounding boxes, while deep learning-based approaches rely on implicit features. We find that the point cloud within the bounding box inherently represents the shape of the object, which can be considered as semantic information and is an important clue for evaluating object consistency. Although the shapes of the same object in consecutive frames are similar, the view angles maybe different. So we propose a point cloud registration based methods to evaluate the point cloud shape similarity.

For detection i in $\mathcal{G}^Q$ and tracklet j in $\mathcal{G}^T$, the point clouds within their respective bounding boxes are denoted as P_i^Q and P_j^T. The ICP algorithm with RANSA[50] is used to compute the point-to-point correspondence between P_i^Q and P_j^T. The fitness score calculated for the two point clouds is given by:

$$S_{fit}(i, j) = \frac{n_c}{max(|P_i^Q|, |P_j^T|)} \quad (13)$$

where n_c represents the number of successfully registered points.

4.3 Subgraph Matching

For each $\mathcal{G}_i^Q$ and its central vertex i, we first collect the candidata set $\mathbf{C}_i$. For each tracklet star-graph in $\mathbf{C}_i$, the distance between its central vertex and i is less than L. For a query star-graph $\mathcal{G}_i^Q$ and a tracklet star-graph $\mathcal{G}_j^T$ in $\mathbf{C}_i$, the multi-criteria feature consistency is given by:

$$S(i, j) = \lambda_1 S_{neighbor}(i, j) + \lambda_2 S_{NGIoU}(i, j) + \lambda_3 S_{fit}(i, j) \quad (14)$$

If $S(i, j)$ falls below a threshold τ, then we consider i and j should not be associated, leading to the removal of j from the candidate set $\mathbf{C}_i$. If the candidate set $\mathbf{C}_i$ is empty, this indicates that i is a newly detected object does not originally exist in the map. We create a new tracklet, which is added as the only element in $\mathbf{C}_i$. Finally, we utilize the Kuhn-Munkres algorithm to compute the final

one-to-one detection-to-tracklet matching by seeking maximum consistency:

$$\mathcal{M}_t^* = \arg\max_{\mathcal{M}_t \in \Psi_t} \sum_{i \in Q_t, j = \mathcal{M}_t^i} S(i, j) \quad (15)$$

where Ψ_t represents the set of all possible matches and $j = \mathcal{M}_t^i$ represents the association of detection i with tracklet j under the matching relationship $\mathcal{M}_t$.
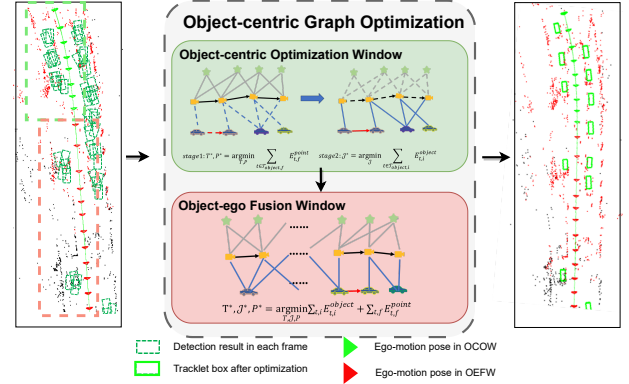

Figure 3: Object-centric Graph Optimization (OGO). Residual edges in solid lines participate in graph optimization, while those in dashed lines do not.

5 Object-centric Graph Optimization

Graph optimization is the widely employed back-end in SLAM (Simultaneous Localization and Mapping)[9, 13, 35, 36], representing ego-motion states and sensor measurements as nodes and edges in a graph, utilizing optimization algorithms to estimate the agent's trajectory and the environment map. However, this approach performs well mainly in static scenes.

For tracking dynamic objects, the ego-motion errors and the object pose errors coexist, affecting the convergence speed and accuracy of the graph optimization. To enable optimization tailored for 3D tracklets, we propose a graph optimization framework named Object-centric Graph Optimization (OGO). We divide the sliding window into two parts: Object-centric Optimization Window (OCOW) and Object-Ego Fusion Window (OEFW), and two windows adopt different optimization strategies.
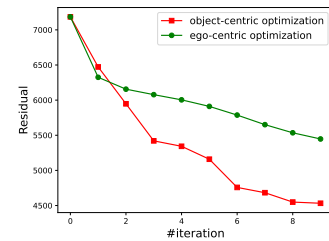

Figure 4: The residual curves of our proposed object-centric optimization and classic ego-centric optimization. The residual refers to the summary of all mapping-based errors and object detection errors in Equation (19).

5.1 Object-centric Optimization Window

Before graph optimization, we obtain initial ego-motion poses and map points for each frame from the visual odometry. Additionally, initial tracklets and the matching result $\mathcal{M}_t$ for each frame are also available after tracking and association.

In the object-centric optimization window, we adopt a two-stage optimization strategy (Figure 3). In the first stage, we solely utilize residuals from static environment landmarks in SLAM and ego-motion poses to estimate ego-motion. At this point, the ego motion serves as a relatively reliable initial value. The objective function of the first stage is:

$$stage\ 1 : \mathbf{T}^*, \mathbf{P}^* = \arg\min_{\mathbf{T},\mathbf{P}} \sum_{t \in \mathcal{T}_{OCOW}} E_t^{map} \qquad (16)$$

where $\mathcal{T}_{OCOW}$ represents the set of frames in OCOW and E_t^{map} represents the mapping residuals in ORBSLAM3[6]. $\mathcal{T}_{OCOW}$ represents the set of ego-motion poses, and $\mathbf{P}$ represents the set of map point positions.

After the first stage, the estimated ego-motion is reliable due to the elimination of dynamic objects in estimation. Then, we fix the ego-motion and solely optimize the object poses using residuals from object detection. The object detection residual at frame t is:

$$E_t^{object} = \sum_{i \in O_t} ||\mathbf{T}_t^{cw}\mathbf{T}_t^{wj}(\mathbf{T}_t^{ci})^{-1}||_F \qquad (17)$$

where $j = \mathcal{M}_t^i$ and $\mathbf{T}_t^{ab}$ represents the transformation from a to b at frame t. The objective function for the sliding window is:

$$stage\ 2 : \mathcal{J}^* = \arg\min_{\mathcal{J}} \sum_{t \in \mathcal{T}_{OCOW}} E_t^{object} \qquad (18)$$

5.2 Object-ego Fusion Window

We combine reliable tracklets and ego-motion poses for joint optimization in OEFW, thereby enhancing the accuracy of both.

When the number of frames where a static object is observed exceeds a threshold w, we move it from the object-centric optimization window (OCOW) to the object-ego fusion window (OEFW). Once all detections in a frame have been moved to the OEFW, we transfer that frame's ego-motion pose and map points to the OEFW. In OEFW, objects and tracklets have undergone sufficient multi-frame observations, possessing good initial values and low system error. Reliable observation and joint optimization can help correct cumulative errors and improve the accuracy of locating and tracking. We employ a tightly coupled optimization strategy within the window, jointly optimizing ego-motion poses, map points, and tracklet poses. The objective function of OEFW is:

$$\mathbf{T}^*, \mathcal{J}^*, \mathbf{P}^* = \arg\min_{\mathbf{T},\mathcal{J},\mathbf{P}} \sum_{t \in \mathcal{T}_{OEFW}} E_t^{object} + E_t^{map} \qquad (19)$$

where $\mathbf{T}, \mathcal{J}, \mathbf{P}$ represent the set of ego-motion poses, the set of object poses, and the set of landmark positions, respectively. $\mathcal{T}_{OEFW}$ is the set of frames in OEFW.

6 System Implementation Details

6.1 System Acceleration

GSLAMOT demands high real-time performance for localization and tracking. We leverage parallel operations, effectively reducing the processing time for each frame of data. Specifically, tasks including visual front-end localization, environment map construction, object detection, and tracking are processed concurrently by different threads. The VO front-end determines whether a frame is a keyframe[6]. Keyframes are processed by MSGA and OGO. Localization and mapping for non-keyframes proceed in other parallel threads, thus eliminating the need to wait for time-consuming keyframe operations. Specifically, the odometry, mapping, detection, and submodules of OGO and MSGA all run on separate threads (Figure 5). In addition, we utilize parallel acceleration for point cloud operations[50] during the computation of shape features. Experimental results demonstrate that our system achieves real-time performance and incurs less time overhead compared to other OOSLAM systems (Section 7.6).
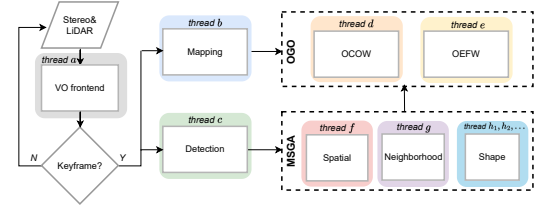

Figure 5: The system architecture and parallel threads.

6.2 Traffic Congestion Dataset

Highly dynamic and congested traffic environments are ubiquitous in the real world, but few data sets emphasize these scenarios. To address this challenge, we use the Carla simulator[10] to simulate dynamic and crowded scenes, and generate the **Traffic Congestion Dataset (TCD)**. We collected several sequences from four maps and the sensors on the vehicle include stereo cameras and 64-channel LiDAR data. In addition to the ego vehicle, hundreds of other dynamic or static vehicles are on the map to simulate dynamic and crowded environments in reality. In each segment sequence of the Waymo Open Dataset[32], the average number of vehicles is less than one hundred, while in our Traffic Congestion dataset, the number of vehicles per segment sequence ranges from 200 to 300. The dense and dynamic flow of vehicles pose challenges to accurate self-localization and MOT.

7 Experiments

7.1 Experimental Settings

We perform evaluations of the proposed algorithm and compare it with the state-of-the-art algorithms, including the SLAM-based systems: ORBSLAM3[6], DynaSLAM[3], VDO-SLAM[45], DSP-SLAM[34] and MOT algorithms: AB3DMOT[37], ProbTrack[8], CenterPoint[40], BOTT[49], SimpleTrack[27], TrajectoryFormer[7]. All experiments are conducted on a computer with a CPU of i7-12700, GPU of RTX 3060Ti, and 16G RAM. We adopt ORB-SLAM3 as our visual odometry front-end. We use Pointpillars[21] for 3D object detection. We

evaluated our system on the KITTI benchmark[12], Waymo Open Dataset[32] and our emulated Traffic Congestion dataset.

Figure 6: The examples of the maps in Traffic Congestion dataset and the outputs of GSLAMOT. Blue points represent ego-motion trajectories, green boxes represent tracklets, and black points represent map points.

7.2 3D Object Tracking Evaluation

We evaluate the results of 3D MOT on both the Traffic Congestion Dataset (TCD) (Table 2) and the widely used Waymo Open Dataset (WOD) (Table 1). For the WOD, we report the mismatch metrics and the MOTA metrics for each of the three categories (vehicle, pedestrian, and cyclist).

Researches on 3D MOT typically assume known ego-motion poses. However, our system computes the ego-motion in real-time, making the comparisons unfair. Therefore, we provide two system implementations: one calculates ego-motion poses simultaneously (GSLAMOT) and another utilizes ground truth ego-motion poses as inputs (GSLAMOT*). Results show that our methods outperform other methods in both known ego-motion and unknown ego-motion scenarios.

Table 1: 3D MOT Evaluation on Waymo Open Dataset. The top one is in bold and the second is underlined.

Method	MOTA(L1)↑	MOTA(L2)↑	Mismatch↓	MOTA(L2)↑ vehicle	MOTA(L2)↑ pedestrian	MOTA(L2)↑ cyclist
AB3DMOT[37]	-	-	-	40.1	33.7	50.39
ProbTrack[8]	48.26	45.25	1.05	54.06	48.10	22.98
CenterPoint[40]	58.35	55.81	0.74	59.38	56.64	60.0
SimpleTrack[27]	59.44	56.92	0.36	56.12	57.76	56.88
BOTT[49]	59.67	<u>57.14</u>	<u>0.35</u>	59.49	58.82	60.41
TrajectoryF[7]	-	-	-	59.7	**61.0**	**60.6**
GSLAMOT	<u>59.69</u>	57.10	**0.33**	<u>60.45</u>	60.02	60.33
GSLAMOT*	**59.75**	**57.20**	**0.33**	**60.47**	<u>60.23</u>	<u>60.45</u>

GSLAMOT: The ego-motion poses are estimated by odometry front-end.
GSLAMOT*: The groundtruth ego-motion poses are given as other MOT algorithms.

Table 2: 3D MOT Evaluation on Traffic Congestion Dataset. The top one is in bold and the second is underlined.

	MOTA↑	MOTP↑	IDS↓	Recall↑	Precision↑
AB3DMOT[37]	36.95	39.42	99	36.49	37.05
ProbTrack[8]	39.07	45.56	77	47.23	43.87
SimpleTrack[27]	<u>43.12</u>	<u>59.91</u>	<u>60</u>	<u>56.35</u>	<u>55.27</u>
GSLAMOT*	**54.36**	**70.92**	**15**	**68.05**	**66.89**
DSP-SLAM[34]	20.95	51.58	51	30.01	<u>57.21</u>
VDO-SLAM[45]	<u>33.24</u>	<u>53.1</u>	<u>29</u>	<u>40.68</u>	55.12
GSLAMOT	**49.10**	**69.71**	**22**	**67.55**	**65.52**

GSLAMOT: The ego-motion poses are estimated by odometry front-end.
GSLAMOT*: The groundtruth ego-motion poses are given as other MOT algorithms.

7.3 Robustness Evaluation

The environment conditions, like rainy, snowy, and foggy days, may interfere with hardware devices and algorithms and then may produce noises. However, few studies have focused on these noises in 3D object tracking. We add Gaussian noise to the ego-motion and 3D detection boxes to test the algorithm's robustness to noise on the Traffic Congestion dataset.

Specifically, gaussian noise with zero mean and standard deviation σ is added to the sampled detections. We gradually increase the value of σ. Experimental results (Figure 8) show that our system maintains superior tracking results compared to other algorithms even as noise increases, demonstrating strong robustness.

7.4 Trajectory Evaluation

We comprehensively evaluate the trajectory accuracy on the KITTI dataset and the Traffic Congestion dataset. We employ standard metrics and compare them against the state-of-the-art SLAM systems. Trajectory results are processed using the EVO tool[14]. Specifically, we consider both the Absolute Pose Error (APE) and Relative Pose Error (RPE) for each trajectory and choose the Root Mean Square Error (RMSE) for a more robust evaluation.

The evaluation results are detailed in Table 3 for the TCD and Table 4 for the KITTI dataset. The results show the superior performance of our approach across different datasets. Although the vehicles in the scenes of the KITTI dataset are not crowded, our proposed association method still achieves a modest improvement. It performs particularly better in sequences with a larger number of objects, such as in KITTI-07. In the TCD dataset, our algorithm has demonstrated significantly superior performance due to the presence of crowded objects in the scenes.

7.5 Ablation Experiments

We test the efficiency of the proposed modules in ablation experiments, including the Multi-criteria Star Graph Association (MSGA) and Object-centric Graph Optimization (OGO). The experiments are conducted on the proposed TCD dataset to better facilitate the evaluation of both 3D MOT and SLAM. In the ablation experiments of OGO (Table 6), the tightly coupled OEFW significantly reduces APE, while the two-stage real-time optimization of OCOW reduces RPE. For the MSGA (Table 5), spatial, neighborhood, and shape consistency all contribute to the improvement of MOT.

7.6 Time Evaluation

To evaluate the system's efficiency and real-time applicability, we measure the average processing time per frame, encompassing all stages, including ego-motion pose estimation, object analysis, association, optimization, and map integration. We conduct experiments over sequences from the KITTI and TCD. The average computation time is reported. The evaluation results in Table 7 affirm GSLAMOT's real-time capability and computational efficiency.

We evaluate the running time of each main module in the system. Note that some modules, such as detection, tracking, and optimization, run only on keyframes selected by the VO front-end. To facilitate a more intuitive time comparison, we calculate the average time expenditure of all modules across all frames. The results are shown in Table 8. Due to the parallel computing strategy, the
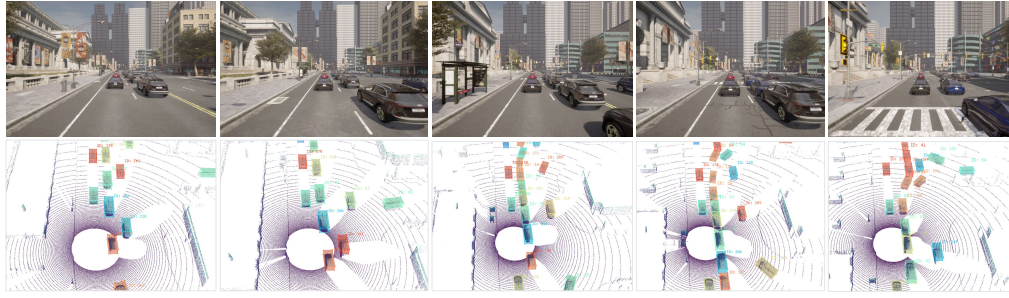
Figure 7: Examples of 3D MOT on Traffic Congestion dataset.

Table 3: Trajectory Accuracy Evaluation on Traffic Congestion Dataset. The top one is in bold and the second is underlined.

TC Seq.	0		1		2		3		4		Average	
Metrics(m)	RPE	APE	RPE	APE	RPE	APE	RPE	APE	RPE	APE	RPE	APE
ORBSLAM3[6]	5.10	0.95	4.88	<u>0.55</u>	2.97	0.68	5.22	0.94	5.14	0.74	4.66	0.77
DSP-SLAM[34]	5.07	0.74	4.73	0.59	2.97	<u>0.56</u>	<u>4.01</u>	0.68	5.11	0.59	4.38	0.63
DynaSLAM[3]	4.86	0.73	4.83	0.61	2.98	0.57	5.2	**0.62**	5.09	0.73	4.59	0.62
VDO-SLAM[45]	<u>4.79</u>	<u>0.69</u>	**4.14**	0.63	<u>2.61</u>	0.59	4.12	0.65	<u>4.81</u>	<u>0.50</u>	<u>4.09</u>	<u>0.61</u>
GSLAMOT(Ours)	**4.73**	**0.58**	<u>4.18</u>	**0.46**	**2.52**	**0.51**	**3.28**	<u>0.63</u>	**3.31**	**0.40**	**3.60**	**0.52**

Table 4: Trajectory Accuracy Evaluation on KITTI Dataset. The top one is in bold and the second is underlined.

KITTI Seq.	00		01		02		03		04		05		06		07		08		Average	
Metrics(m)	RPE	APE	RPE	APE	RPE	APE	RPE	APE	RPE	APE	RPE	APE	RPE	APE	RPE	APE	RPE	APE	RPE	APE
ORBSLAM3[6]	2.09	1.46	7.52	<u>12.70</u>	2.3	3.5	0.84	1.44	<u>0.6</u>	<u>0.25</u>	0.91	0.93	0.92	0.99	0.49	0.49	<u>3.06</u>	**3.06**	2.08	<u>2.76</u>
DSP-SLAM[34]	1.09	<u>1.10</u>	3.87	**12.06**	<u>0.94</u>	**0.89**	1.28	**0.47**	0.64	0.73	**0.53**	<u>0.46</u>	0.81	**0.42**	0.5	0.48	3.17	11.99	1.40	3.18
DynaSLAM[3]	1.05	1.28	<u>3.75</u>	21.13	1.1	<u>0.91</u>	**0.68**	1.43	0.73	0.82	0.64	1.52	0.8	1.35	0.51	0.78	<u>3.06</u>	10.41	<u>1.36</u>	4.40
VDO-SLAM[45]	<u>1.02</u>	1.44	3.80	13.79	0.98	0.99	<u>0.79</u>	0.83	0.61	<u>0.25</u>	<u>0.59</u>	0.49	<u>0.75</u>	0.63	0.49	0.52	3.34	9.76	1.50	3.19
GSLAMOT(Ours)	**1.01**	**1.02**	**3.69**	13.1	**0.92**	**0.91**	**0.68**	<u>0.57</u>	**0.56**	**0.23**	**0.53**	**0.41**	**0.70**	<u>0.44</u>	**0.48**	**0.43**	**3.05**	<u>3.16</u>	**1.29**	**2.25**

(a) MOTA   **(b) MOTP**
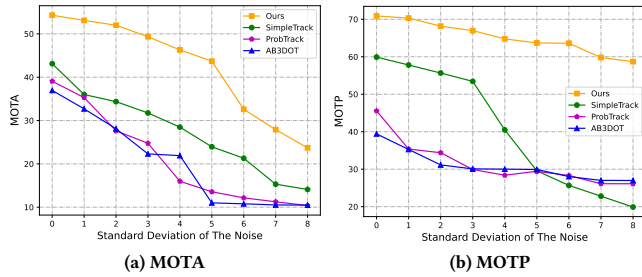**Figure 8: Results in robustness evaluation experiments.**

Table 5: Ablation Experiments of MSGA on TCD.

MSSA	MOTA↑	MOTP↑	APE↓	RPE↓
Spatial	43.25	61.01	0.59	3.82
Spatial+Neighborhood	<u>46.79</u>	<u>67.88</u>	<u>0.57</u>	<u>3.77</u>
Spatial+Neighborhood+Shape	**49.10**	**69.71**	**0.52**	**3.60**

Table 6: Ablation Experiments of OGO on TCD.

OGO	MOTA↑	MOTP↑	APE↓	RPE↓
OEFW	43.66	59.91	<u>0.63</u>	4.21
OCOW	<u>48.93</u>	<u>63.40</u>	1.30	<u>3.81</u>
OCOW+OEFW	**49.10**	**69.71**	**0.52**	**3.60**

total time expenditure per frame is significantly less than the time consumed by serial computation of individual modules.

Table 7: Time Evaluation.

Method	Time Per Frame(s)↓
ORBSLAM3[6]	0.025
DSP-SLAM[34]	0.12
DynaSLAM[3]	0.086
VDO-SLAM[45]	0.14
GSLAMOT (ours)	0.082

Table 8: Running Time of the Main Modules in Our System.

Module	Time Per Frame(s)↓
Ego-motion Tracking	0.031
Mapping	0.027
3D Detection†	0.022
MSSA†	0.024
OGO†	0.037
Total	0.082

†: only for keyframes.

8 Conclusion

This paper investigates the problem of Simultaneous Locating, Mapping, and 3D Multiple Object Tracking. We propose the MSGA algorithm, which computes features from query graphs and tracklet graphs to achieve detection and tracklet association. We design the OGO strategy to optimize ego-motion and tracklet poses in multiple stages. Additionally, we create a simulation dataset to explore the challenges of 3D MOT in highly congested, dynamic scenes. In the future, we plan to extend GSLAMOT to multi-agent collaborative scenarios to leverage the information from multiple agents and viewpoints to enhance system performance.

9 Acknowledgments

References

[1] Ayman Beghdadi and Malik Mallem. 2022. A comprehensive overview of dynamic visual SLAM and deep learning: concepts, methods and challenges. *Machine Vision and Applications* 33, 4 (2022), 54.

[2] Berta Bescos, Carlos Campos, Juan D Tardós, and José Neira. 2021. DynaSLAM II: Tightly-coupled multi-object tracking and SLAM. *IEEE robotics and automation letters* 6, 3 (2021), 5191–5198.

[3] Berta Bescos, José M Fácil, Javier Civera, and José Neira. 2018. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters* 3, 4 (2018), 4076–4083.

[4] Martin Büchner and Abhinav Valada. 2022. 3d multi-object tracking using graph neural networks with cross-edge modality attention. *IEEE Robotics and Automation Letters* 7, 4 (2022), 9707–9714.

[5] Xudong Cai, Yongcai Wang, Zhe Huang, Yu Shao, and Deying Li. 2024. VOLoc: Visual Place Recognition by Querying Compressed Lidar Map. *arXiv preprint arXiv:2402.15961* (2024).

[6] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. 2021. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics* 37, 6 (2021), 1874–1890.

[7] Xuesong Chen, Shaoshuai Shi, Chao Zhang, Benjin Zhu, Qiang Wang, Ka Chun Cheung, Simon See, and Hongsheng Li. 2023. TrajectoryFormer: 3D Object Tracking Transformer with Predictive Trajectory Hypotheses. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 18527–18536.

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

[9] Frank Dellaert. 2012. Factor graphs and GTSAM: A hands-on introduction. *Georgia Institute of Technology, Tech. Rep* 2 (2012), 4.

[10] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR, 1–16.

[11] Sourav Garg, Niko Sünderhauf, Feras Dayoub, Douglas Morrison, Akansel Cosgun, Gustavo Carneiro, Qi Wu, Tat-Jun Chin, Ian Reid, Stephen Gould, et al. 2020. Semantics for robotic mapping, perception and interaction: A survey. *Foundations and Trends® in Robotics* 8, 1–2 (2020), 1–224.

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 3354–3361.

[13] Giorgio Grisetti, Rainer Kümmerle, Hauke Strasdat, and Kurt Konolige. 2011. g2o: A general framework for (hyper) graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 9–13.

[14] Michael Grupp. 2017. evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo.

[15] Xiaoshui Huang, Guofeng Mei, Jian Zhang, and Rana Abbas. 2021. A comprehensive survey on point cloud registration. *arXiv preprint arXiv:2103.02690* (2021).

[16] Zhe Huang, Yongcai Wang, Xingui Tang, and Hongyu Sun. 2023. Boundary-aware set abstraction for 3D object detection. In *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 01–07.

[17] Zhe Huang, Yongcai Wang, Jie Wen, Peng Wang, and Xudong Cai. 2023. An object detection algorithm combining semantic and geometric information of the 3D point cloud. *Advanced Engineering Informatics* 56 (2023), 101971.

[18] Aleksandr Kim, Guillem Brasó, Aljoša Ošep, and Laura Leal-Taixé. 2022. PolarMOT: How far can geometric relations take us in 3D multi-object tracking?. In *European Conference on Computer Vision*. Springer, 41–58.

[19] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* 23, 6 (2021), 4909–4926.

[20] Lars Kunze, Nick Hawes, Tom Duckett, Marc Hanheide, and Tomáš Krajník. 2018. Artificial intelligence for long-term robot autonomy: A survey. *IEEE Robotics and Automation Letters* 3, 4 (2018), 4023–4030.

[21] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12697–12705.

[22] Zechen Liu, Zizhang Wu, and Roland Tóth. 2020. Smoke: Single-stage monocular 3d object detection via keypoint estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 996–997.

[23] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. 2021. Multiple object tracking: A literature review. *Artificial intelligence* 293 (2021), 103448.

[24] Aashi Manglik, Xinshuo Weng, Eshed Ohn-Bar, and Kris M Kitanil. 2019. Forecasting time-to-collision from monocular video: Feasibility, dataset, and challenges. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 8081–8088.

[25] Brendan D McKay et al. 1981. Practical graph isomorphism. (1981).

[26] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. 2018. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters* 4, 1 (2018), 1–8.

[27] Ziqi Pang, Zhichao Li, and Naiyan Wang. 2022. Simpletrack: Understanding and rethinking 3d multi-object tracking. In *European Conference on Computer Vision*. Springer, 680–696.

[28] Yesul Park, L Minh Dang, Sujin Lee, Dongil Han, and Hyeonjoon Moon. 2021. Multiple object tracking in deep learning approaches: A survey. *Electronics* 10, 19 (2021), 2406.

[29] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 658–666.

[30] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. 2013. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1352–1359.

[31] Zolbayar Shagdar, Mohib Ullah, Habib Ullah, and Faouzi Alaya Cheikh. 2021. Geometric deep learning for multi-object tracking: A brief review. In *2021 9th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 1–6.

[32] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. 2020. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2446–2454.

[33] Keisuke Tateno, Federico Tombari, and Nassir Navab. 2016. When 2.5 D is not enough: Simultaneous reconstruction, segmentation and recognition on dense SLAM. In *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2295–2302.

[34] Jingwen Wang, Martin Rünz, and Lourdes Agapito. 2021. DSP-SLAM: Object oriented SLAM with deep shape priors. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 1362–1371.

[35] Shuo Wang, Yongcai Wang, Xuewei Bai, and Deying Li. 2023. Communication Efficient, Distributed Relative State Estimation in UAV Networks. *IEEE Journal on Selected Areas in Communications* 41, 4 (2023), 1151–1166. https://doi.org/10.1109/JSAC.2023.3242708

[36] Shuo Wang, Yongcai Wang, Deying Li, and Qianchuan Zhao. 2023. Distributed Relative Localization Algorithms for Multi-Robot Networks: A Survey. *Sensors* 23, 5 (2023). https://doi.org/10.3390/s23052399

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

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

[39] Shichao Yang and Sebastian Scherer. 2019. Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics* 35, 4 (2019), 925–938.

[40] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. 2021. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11784–11793.

[41] Angeliki Zacharaki, Ioannis Kostavelis, Antonios Gasteratos, and Ioannis Dokas. 2020. Safety bounds in human robot interaction: A survey. *Safety science* 127 (2020), 104667.

[42] Jan-Nico Zaech, Alexander Liniger, Dengxin Dai, Martin Danelljan, and Luc Van Gool. 2022. Learnable online graph representations for 3D multi-object tracking. *IEEE Robotics and Automation Letters* 7, 2 (2022), 5103–5110.

[43] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. 2022. Motr: End-to-end multiple-object tracking with transformer. In *European Conference on Computer Vision*. Springer, 659–675.

[44] Hanwei Zhang, Hideaki Uchiyama, Shintaro Ono, and Hiroshi Kawasaki. 2022. MOTSLAM: MOT-assisted monocular dynamic SLAM using single-view depth estimation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and*

Systems (IROS). IEEE, 4865–4872.

[45] Jun Zhang, Mina Henein, Robert Mahony, and Viorela Ila. 2020. VDO-SLAM: a visual dynamic object-aware SLAM system. *arXiv preprint arXiv:2005.11052* (2020).

[46] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2022. Bytetrack: Multi-object tracking by associating every detection box. In *European Conference on Computer Vision*. Springer, 1–21.

[47] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. 2021. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision* 129 (2021), 3069–3087.

[48] Yuang Zhang, Tiancai Wang, and Xiangyu Zhang. 2023. Motrv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 22056–22065.

[49] Lubing Zhou, Xiaoli Meng, Yiluan Guo, and Jiong Yang. 2023. BOTT: Box Only Transformer Tracker for 3D Object Tracking. *arXiv preprint arXiv:2308.08753* (2023).

[50] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847* (2018).

[51] Hu Zhu. 2023. Awesome Dynamic SLAM. https://github.com/zhuhu00/Awesome_Dynamic_SLAM.