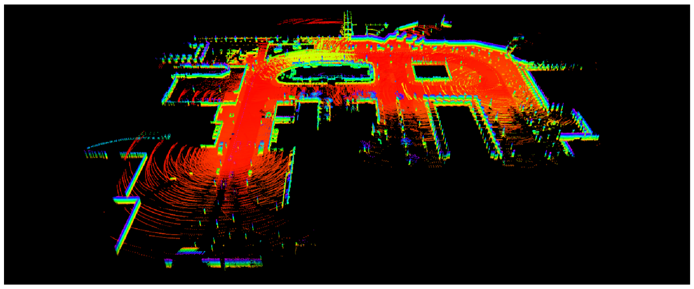
IMM-SLAMMOT : Tightly-Coupled SLAM and IMM-based Multi-Object Tracking

Zhuoye Ying[1], and Hao Li*[1,2]

***Abstract*—In the context of autonomous driving systems, SLAM and dynamic object tracking represent pivotal challenges. Autonomous driving scenarios frequently demand the simultaneous acquisition of ego-pose and comprehensive motion information from the surrounding environment to enhance decision-making and scene comprehension.Given the inherent interdependence between these two challenges, a viable approach is to integrate SLAM and object tracking into an interconnected system referred to as SLAMMOT. However, many conventional SLAMMOT solutions rely on a single motion model for object tracking, which may inadequately capture complicated dynamics of real-world motions. In practice, object motion patterns can change from time to time, not conforming neatly to a single model. To handle existing challenges, this paper proposes the IMM-SLAMMOT, a tightly-coupled LiDAR-based SLAMMOT system that utilizes instance semantic segmentation and IMM modelling for dynamic object tracking. Ego-pose and dynamic object states are jointly optimized in an innovative graph optimization framework intimately integrated with the IMM algorithm. Comparative analysis against our baseline, which employs a single motion model for object tracking, demonstrates that the IMM-SLAMMOT outperforms at motion-pattern-transition moments and consistently achieves competitive results in SLAM and multi-object tracking tasks throughout the entire trajectory.**

***Index Terms*—Simultaneous localization and mapping (SLAM), multi-object tracking, interacting multiple model (IMM), graph optimization**


(a) LiDAR odometry and mapping.

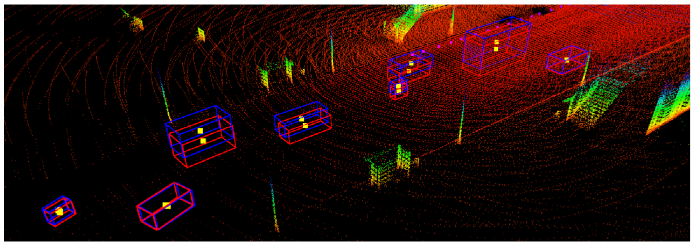

(b) Object tracking: blue bounding boxes denote the ground truth of dynamic objects in the scenario, and red bounding boxes denote corresponding estimates output by the MOT Module

Fig. 1. Output of the IMM-SLAMMOT: a tightly-coupled SLAM and IMM-based MOT system

I. INTRODUCTION

SIMULTANEOUS localization and mapping (SLAM) and multi-object tracking (MOT) represent two interrelated challenges in the domain of autonomous driving. SLAM focuses on the task of constructing a map of an unknown environment while simultaneously estimating the ego-pose within this map. In contrast, MOT is geared towards analysing the states of surrounding moving objects, and has a wide range of applications within the context of autonomous driving. Presently, extensive research has been conducted on these two problems separately, often operating under certain prerequisite assumptions.

The majority of SLAM solutions operate under the assumption of a static environment. While these methods primarily utilize cameras [1] [2] [3] [4] [5] and LiDAR sensors [6] [7], they can also incorporate other auxiliary sensors [8] to enhance their performance. Nonetheless, real-world environments often contain a substantial number of dynamic objects, and the accuracy of traditional SLAM approaches tends to degrade in such scenarios.

Work supported by the SJTU Young Talent Funding (WF220426002)

[1]École d'Ingénieurs SJTU-ParisTech (SPEIT), Shanghai, 200240, China.
[2]Department of Automation, Shanghai Jiao Tong University (SJTU), Shanghai, 200240, China.
*Corresponding author: Hao Li (email: haoli@sjtu.edu.cn).

For MOT tasks, it is a common practice to consider the ego-pose of the observer vehicle as a known prior, with little focus on its estimation. Consequently, the precision of conventional multi-target tracking methodologies is heavily reliant on the accuracy of ego-pose estimation. However, in complex dynamic environments where the existence of dependable static structures cannot be guaranteed, the reliability of ego-pose estimation tends to deteriorate.

SLAM and MOT are closely intertwined, because of their inherent interactive relationship. In SLAM tasks, it is imperative to consider the impact of moving object states to enhance ego-pose estimation accuracy. Conversely, in MOT, an accurate ego-pose estimation from the SLAM module significantly improves object state estimation. Consequently, some efforts have been directed toward the development of coupled MOT and SLAM systems. The concept of SLAMMOT [9] [10] [11] [12], originally introduced in [13], has seen substantial progress in recent years. However, it's noteworthy that most of these endeavours have primarily focused on ensuring accuracy throughout the entire trajectory, relying on a single motion model for tracking. In contrast, it's crucial to recognize and address motion-pattern-transition moments that are not consistent with a single motion model. Take driving scenarios as an example; significant traffic incidents often occur when vehicles abruptly stop or change their motion patterns.

To handle existing challenges, we propose the IMM-SLAMMOT, an example output of which is illustrated in Fig. 1. Our proposed method begins with instance semantic segmentation, enabling to distinguish between static and moving points and to extract 3D bounding boxes from LiDAR scans. Subsequently, we establish parallelism between the LiDAR-based SLAM and MOT modules as the front-end components. For MOT, we adopt the Interacting Multiple Model (IMM) technique [14] [15], which proves to be more adaptable to complex real-world dynamic scenarios. In the final step, we integrate the IMM-based MOT with the LiDAR-based SLAM through a tightly coupled graph optimization process that aligns seamlessly with IMM algorithms.

The highlights of the presented works are summarized as follows:

- the holistic cooperative SLAMMOT system considering multiple motion models (via the IMM).
- The graph optimization back-end that seamlessly integrates with IMM algorithms.
- Demonstrated competitiveness of our proposed method in terms of ego-pose estimation and multi-object tracking throughout the entire trajectory.
- Significant competitiveness of our proposed method in handling motion-pattern-transition moments of moving objects.

II. Related Works

A. Simultaneous Localization and Mapping

Traditional SLAM methods operate under the assumption of a static world, which work well in most scenarios where the majority of objects remain stationary. In terms of sensor usage, the bulk of SLAM approaches can be categorized into visual-based SLAM and LiDAR-based SLAM.

The advent of visual sensors has given birth to Visual SLAM techniques, leveraging camera data for the construction of maps and the estimation of poses. Remarkable exemplars in this category include LSD-SLAM [1] and ORB-SLAM [3] [4] [5], both offering robust real-time solutions for visual SLAM tasks. The LSD-SLAM adopts a direct-method-based approach, estimating camera motion and scene structure by exploiting image brightness information rather than relying on feature points. Its proficiency extends to scenarios encompassing both large-scale and small-scale environments. In contrast, the ORB-SLAM relies on extraction of feature points and subsequent descriptor matching to simultaneously perform localization and map generation. It distinguishes itself as a highly acclaimed visual SLAM system with various versions and iterations.

The advent of LiDAR-based SLAM has garnered considerable attention, primarily owing to the widespread availability of LiDAR sensors. The LOAM [6] is a prominent representative of real-time localization and map construction. It relies on discrete scan matching and point cloud registration techniques. Building upon LOAM, LeGO-LOAM [7] emerges as an enhanced version, strategically optimized for ground-based point cloud data to enhance localization precision. Recently, Guo et al. [16] proposed a principle component analysis (PCA)-based feature extraction method for keyframe-based 3D LiDAR SLAM, which achieves globally consistent estimation performance.

Moreover, certain approaches leverage auxiliary sensors or employ a combination of both camera and LiDAR sensors to enhance the robustness of SLAM results. LIO-SAM [8] adopts a distinctive strategy by incorporating LiDAR and Inertial Measurement Unit (IMU) data. This integration enhances the localization and mapping robustness, rendering it suitable for diverse environments and dynamic scenarios. Shin et al. [17] employ the direct method, which is more robust under sparse depth with a narrow field of view, and combine camera and LiDAR measurements. In a different vein, Huang et al. [18] introduce a LiDAR-monocular visual odometry approach that incorporates point and line features.

B. Multi-Object Tracking

Researches in the realm of multi-object tracking typically assume knowledge of the ego-pose. The AB3DMOT [19] introduces a fundamental framework for LiDAR-based Multi-Object Tracking. It initially acquires 3D detections from LiDAR point cloud data, followed by a straightforward combination of 3D Kalman Filter and the Hungarian algorithm for state estimation and data association. Due to its lightweight architecture and low computational demands, the AB3DMOT achieves real-time performance. Huang et al. [20] have developed an end-to-end deep neural network that simultaneously handles object detection and correlation using data from both cameras and LiDAR sensors. They also employ a mixed-integer programming algorithm for data association, optimizing among detection confidences, affinities, and start-end probabilities. More recently, the DeepFusionMOT [21] has proposed a camera-LiDAR fusion-based Multi-Object Tracking method with a deep association mechanism. It tracks an object in a 2D domain when the object is far away and only visible to the camera. It then updates the 2D trajectory with 3D information obtained when the object enters the LiDAR field of view, achieving a seamless fusion of 2D and 3D trajectories. Additionally, Wang et al. [22] have introduced a combined appearance-motion optimization technique that utilizes both camera and LiDAR data to significantly reduce tracking failures caused by occlusions and false detections.

C. Coupled Multi-Object Tracking and SLAM

Due to the close correlation between SLAM and Multi-Object Tracking (MOT), the coupled system SLAMMOT originated from [13]. They established a mathematical framework to integrate a filtering-based SLAM and moving object tracking, demonstrating its ability to meet navigation and safety requirements in autonomous driving. The ClusterSLAM [9], on the other hand, constructs a noise-aware motion affinity matrix based on landmarks and employs agglomerative clustering to distinguish rigid bodies. As the ClusterSLAM serves as a back end rather than a complete system, its performance is heavily reliant on the quality of landmark extraction and association from the front end.

The same authors have developed the full system ClusterVO [10], which is capable of online processing for multiple motion estimations. To achieve this, the system employs a multi-level probabilistic association mechanism and a heterogeneous Conditional Random Field (CRF) clustering approach, combining semantic, spatial, and motion information to jointly infer cluster segmentations online for every frame.

The VDO-SLAM [11] utilizes dense optical flow to maximize the number of tracked points on moving objects. They implement a bundle adjustment with cameras, objects, and points, yielding promising results but introducing computational complexity. To ensure accurate camera pose estimation over extended time spans, Du et al. [23] provide a dynamic 3D landmark detection method, followed by the use of long-term consistency via conditional random fields. The DynaSLAM2 [12] utilizes instance semantic segmentation and ORB features to track dynamic objects. It optimizes the structures of the static scene and dynamic objects jointly with the trajectories of both the camera and the moving agents within a bundle adjustment proposal. Recently, MOTSLAM [24] has incorporated neural network-based monocular depth estimation to generate 3D positions of dynamic objects. It jointly optimizes camera poses, object poses, and both static and dynamic map points using a bundle adjustment method.

A primary challenge for SLAMMOT is the inherent non-deterministic nature of moving objects (more specifically, the inherent non-deterministic nature of surrounding moving objects' motion patterns) in practical applications. For example, we can never know *a priori* whether a (potentially) moving object is actually stationary, moving at constant speed, or turning at constant yaw rate, etc. Such inherent non-deterministic nature has crucial influence on SLAMMOT. In fact, the difficulty of SLAMMOT consists right in the inherent non-deterministic nature of moving objects. If complete *a priori* knowledge of moving objects could be available, then SLAMMOT would just be a trivial extension of traditional SLAM.

How moving objects (including their non-deterministic nature) are treated in SLAMMOT, or in other words, how MOT is treated for SLAM in SLAMMOT, distinguishes existing SLAMMOT methods. From the methodology perspective, existing SLAMMOT methods (be them vision-based or LiDAR-based) may be roughly categorized into four methodology levels:

Methodology Level 0: no cooperation between SLAM and MOT (namely to treat all objects as stationary in SLAM, be them actually so or not). Traditional SLAM methods belong to this category.

Methodology Level 1: slight cooperation between SLAM and MOT (namely to use certain semantic segmentation method to determine potential moving objects and then to heuristically sift out potential moving objects in SLAM, be them actually so or not). Typical works include [9] [10] [25].

Methodology Level 2: holistic cooperation between SLAM and MOT considering a single motion model (namely to holistically fuse state estimation of moving objects and SLAM while fitting moving objects rigidly with a single motion model). Typical works include [11] [12] [24].

Methodology Level 3: holistic cooperation between SLAM and MOT considering multiple motion models (via the IMM), i.e. our proposed one (namely to holistically fuse state estimation of moving objects and SLAM while fitting moving objects flexibly with multiple motion models). Considering multiple motion models, which enables the SLAMMOT to effectively handle the inherent non-deterministic nature of moving objects, is the essential point of the proposed methodology.

III. PREREQUISITE

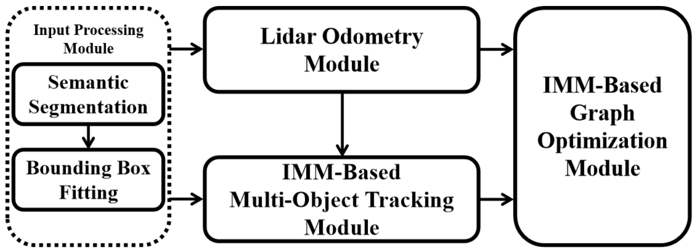


Fig. 2. Overall Architecture of the IMM-SLAMMOT.

A. Overall Architecture of the IMM-SLAMMOT

The IMM-SLAMMOT extends from the foundation of LiDAR-based SLAM, specifically LeGO-LOAM. It takes LiDAR scans as its input and provides both the LiDAR and dynamic object poses for each frame, along with a spatial/temporal map containing the dynamic objects. The architectural structure of our method is depicted in Fig. 2.

Benefiting from advancements in 3D object detection, we have access to high-quality detections [26] [27]. Within the Input Processing Module, we employ a 3D object detection method to distinguish between static points and dynamic objects. Subsequently, the LiDAR Odometry Module, utilizes the static points to compute the ego-pose of the LiDAR. Concurrently, taking the ego-pose and dynamic objects as inputs, the IMM-based MOT Module estimates the object states in the current frame. Finally, both the ego-pose and all dynamic states are collectively optimized within a Graph Optimization Module, closely integrated with the IMM algorithm. Notably, for real-time applications, the processing rate of the Graph Optimization Module is lower compared to the LiDAR Odometry Module and the IMM-based MOT Module.

The distinct contributions and components of the IMM-SLAMMOT are further elaborated in the following sections.

B. Input Processing Module

The details of the Input Processing Module are presented in the left segment of the overall architecture depicted in Fig. 2. This module takes LiDAR scans as input and yields static point clouds and 3D bounding boxes of dynamic objects as output. In this context, we consider segmented points with ”vehicle” labels and those with ”pedestrian” labels as potential moving objects, while categorizing the remaining points as static. Notably, in our practical implementations of this component, we benefit from advancements in instance semantic segmentation in 3D point clouds. This allows us to seamlessly integrate Semantic Segmentation and Bounding Box Fitting into a single deep neural network, such as PointRCNN [26] or CenterPoint [27]. Consequently, we employ one of these deep learning-based methods directly as our Input Processing Module.

C. LiDAR Odometry Module

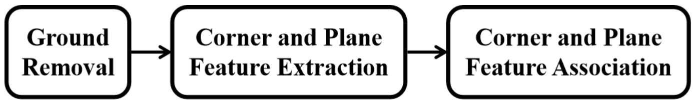

Ground Removal
Corner and Plane Feature Extraction
Corner and Plane Feature Association

Fig. 3. Details of LiDAR Odometry Module.

Our LiDAR Odometry Module relies on a single LiDAR sensor, for which we have selected the LeGO-LOAM as the foundational framework. The detailed process of this component is illustrated in Fig. 3. The input comprises static points, which notably include a substantial number of ground points. As the LeGO-LOAM, we commence by removing the ground points to extract corner features and plane features from the non-ground cloud. Subsequently, we perform separate associations of corner feature points and plane feature points (including the downsampled ground points) between consecutive frames to calculate the transformations between these frames. Consequently, the current ego-pose is estimated through the accumulation of all frame-to-frame transformations and is then provided as output to both the IMM-based MOT Module and the IMM-based Graph Optimization Module.

IV. IMM-BASED MODULES

A. Notation

To handle real-world motion patterns that are changeable and unpredictable, we employ three fundamental motion models: the Constant Position (CP) model, the Constant Velocity (CV) model, and the Constant Turning Velocity (CTRV) model. Let $\boldsymbol{x}_t^m$ denote the object state at the time t under the model m. Denote the states associated with multiple motion patterns respectively as

1) Constant Position: $\boldsymbol{x}_t^p = [x_t, y_t, \theta_t]^T$
2) Constant Velocity: $\boldsymbol{x}_t^v = [x_t, y_t, \theta_t, v_t]^T$
3) Constant Turning Velocity: $\boldsymbol{x}_t^r = [x_t, y_t, \theta_t, v_t, \omega_t]^T$

In the aforementioned notations, x_t and y_t denote the object position in the horizontal plane, θ_t denotes the rotation angle along the vertical axis, v_t and ω_t denote the linear velocity and angular velocity, respectively.

According to the motion patterns, IMM-based state estimation is performed as follow: Following physical laws of Constant Turning Velocity Model, we have

$$\boldsymbol{x}_t^r = g^r(\boldsymbol{x}_{t-1}^r) \tag{1}$$

Where

$$g^r(\boldsymbol{x}_t^r) = \begin{bmatrix} x_t + v_t\Delta t\cos(\theta_t + \frac{\omega_t\Delta t}{2}) \\ y_t + v_t\Delta t\sin(\theta_t + \frac{\omega_t\Delta t}{2}) \\ \theta_t + \omega_t\Delta t \\ v_t \\ \omega_t \end{bmatrix} \tag{2}$$

The transition equation $\boldsymbol{x}_t^r = g^r(\boldsymbol{x}_{t-1}^r)$ can be discretely approximated as

$$\boldsymbol{x}_t^r \simeq \frac{\partial g^r}{\partial \boldsymbol{x}_t^r}\boldsymbol{x}_{t-1}^r \tag{3}$$

Let

$$\boldsymbol{A}_t^r = \frac{\partial g^r}{\partial \boldsymbol{x}_t^r} \tag{4}$$

Finally, we have

$$\boldsymbol{x}_t^r \simeq \boldsymbol{A}_{t-1}^r\boldsymbol{x}_{t-1}^r \tag{5}$$

For the Constant Position Model and Constant Velocity Model, we simply set the angular velocity and linear velocity to zero. Therefore, the transition equations can be formulated as follows:

$$\boldsymbol{x}_t^p = \boldsymbol{A}_{t-1}^p\boldsymbol{x}_{t-1}^p \tag{6}$$

With $\boldsymbol{A}_t^p = I_3$.

$$\boldsymbol{x}_t^v \simeq \boldsymbol{A}_{t-1}^v\boldsymbol{x}_{t-1}^v \tag{7}$$

With

$$\boldsymbol{A}_t^v = \begin{bmatrix} 1 & 0 & -v_t\Delta t\sin(\theta_t) & \Delta t\cos(\theta_t) \\ 0 & 1 & v_t\Delta t\cos(\theta_t) & \Delta t\sin(\theta_t) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

B. IMM-based MOT Module

For each dynamic object in the current frame, we employ three system models to characterize its motion. At the beginning of each recursive cycle, we initialize with three model weights w_{t-1}^c, three state means $\hat{\boldsymbol{x}}_{t-1}^c$, and three associated covariances $\hat{\boldsymbol{P}}_{t-1}^c$, with $c \in \{p, v, r\}$, where p, v, r denote Constant Position Model, Constant Velocity Model and Constant Turning Velocity Model respectively. The transition probabilities are denoted by $\boldsymbol{C} \in \mathbb{R}^{3\times3}$. The IMM-based MOT Module consists of the following five steps:

1) Merging of the CP-CV-CTRV system model: For the d estimation track, $d \in \{p, v, r\}$, its initial estimate $\boldsymbol{x}_t^{d,M}$ is obtained by a weighted merging of all three states $\hat{\boldsymbol{x}}_{t-1}^c$ that contribute to $\boldsymbol{x}_t^{d,M}$ according to their model weights w_{t-1}^c and transition probabilities $\boldsymbol{C}_{cd}$. For model weight merging we have

$$w_t^{d,M} = \sum_{c\in\{p,v,r\}} \boldsymbol{C}_{cd} w_{t-1}^c \tag{9}$$

For state mean merging we have

$$\boldsymbol{x}_t^{d,M} = \frac{1}{w_t^{d,M}} \sum_{c\in\{p,v,r\}} \hat{\boldsymbol{x}}_{t-1}^c \boldsymbol{C}_{cd} w_{t-1}^c \tag{10}$$

For state covariance merging we have

$$\boldsymbol{P}_t^{d,M} = \frac{1}{w_t^{d,M}} \sum_{c\in\{p,v,r\}} \boldsymbol{C}_{cd} w_{t-1}^c [\hat{\boldsymbol{P}}_{t-1}^c + \Delta\boldsymbol{x}_{t,cd}^2] \tag{11}$$

Where

$$\Delta\boldsymbol{x}_{t,cd}^2 = (\hat{\boldsymbol{x}}_{t-1}^c - \boldsymbol{x}_t^{d,M})(\hat{\boldsymbol{x}}_{t-1}^c - \boldsymbol{x}_t^{d,M})^T \tag{12}$$

2) Distributed prediction tracks using the CP-CV-CTRV system models: For the d estimation track, $d \in \{p, v, r\}$, its initial estimate $\{\boldsymbol{x}_t^{d,M}, \boldsymbol{P}_t^{d,M}\}$ is used to predict a priori estimate $\{\bar{\boldsymbol{x}}_t^d, \bar{\boldsymbol{P}}_t^d\}$ via its corresponding representative system model. A predicted state, resulting from merging all three predictions, is then used to search for associated measurement using Kuhn–Munkres algorithm [28] [29].

Prediction:

$$\bar{\boldsymbol{x}}_t^d = \boldsymbol{A}_{t-1}^d \boldsymbol{x}_{t-1}^d \quad , \quad \bar{\boldsymbol{P}}_t^d = \boldsymbol{A}_{t-1}^d \hat{\boldsymbol{P}}_{t-1}^d \boldsymbol{A}_{t-1}^{d\ T} + \boldsymbol{Q}_d \tag{13}$$

with $\boldsymbol{Q}_d$ represents a diagonal matrix denoting the system noise of model d.

$$\begin{cases} \boldsymbol{Q}_p \in \mathbb{R}^{3\times3} \\ \boldsymbol{Q}_v \in \mathbb{R}^{4\times4} \text{ and } \forall 1 \le i \le 3 \ \boldsymbol{Q}_{v,ii} = 0 \\ \boldsymbol{Q}_r \in \mathbb{R}^{5\times5} \text{ and } \forall 1 \le i \le 3 \ \boldsymbol{Q}_{r,ii} = 0 \end{cases}$$

Therefore, the merging predicted state is

$$\bar{\boldsymbol{x}}_t = w_t^{p,M}\bar{\boldsymbol{x}}_t^p + w_t^{v,M}\bar{\boldsymbol{x}}_t^v + w_t^{r,M}\bar{\boldsymbol{x}}_t^r \tag{14}$$

and by using the Kuhn–Munkres algorithm, the associated measurement is denoted by $\boldsymbol{z}_t = [x_t, y_t, \theta_t]^T$.

3) Distributed update tracks using the CP-CV-CTRV system models: The distributed update step is same as Extended Kalman Filter [30], for the d estimation track, $d \in \{p, v, r\}$, we have

$$\begin{aligned} \hat{\boldsymbol{x}}_t^d &= \bar{\boldsymbol{x}}_t^d + \boldsymbol{K}(\boldsymbol{z}_t - \boldsymbol{I}_d\bar{\boldsymbol{x}}_t^d) \\ \hat{\boldsymbol{P}}_t^d &= (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{I}_d)\bar{\boldsymbol{P}}_t^d \\ \boldsymbol{K} &= \bar{\boldsymbol{P}}_t^d(\boldsymbol{I}_d\bar{\boldsymbol{x}}_t^d)^T[(\boldsymbol{I}_d\bar{\boldsymbol{x}}_t^d)\bar{\boldsymbol{P}}_t^d(\boldsymbol{I}_d\bar{\boldsymbol{x}}_t^d)^T + \boldsymbol{R}]^{-1} \end{aligned} \tag{15}$$

With $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ the measurement noise and $\boldsymbol{I}_d$ the identity matrix associated with dimension of $\boldsymbol{x}_t^d$, i.e. $\boldsymbol{I}_d = \boldsymbol{I}_{3,n_d}$ with n_d dimension of $\boldsymbol{x}_t^d$.

4) CP-CV-CTRV model weight update: For the d estimation track, $d \in \{p, v, r\}$, its weight w_t^d is updated from its initial weight $w_t^{d,M}$ according to its measurement innovation:

$$w_t^d = \eta \frac{w_t^{d,M}}{\sqrt{\det \boldsymbol{S}_t^d}} e^{-\frac{1}{2}[\boldsymbol{z}_t - \boldsymbol{I}_d\hat{\boldsymbol{x}}_t^d]^T(\boldsymbol{S}_t^d)^{-1}[\boldsymbol{z}_t - \boldsymbol{I}_d\hat{\boldsymbol{x}}_t^d]} \tag{16}$$

Where η id a normalization constant and

$$\boldsymbol{S}_t^d = \boldsymbol{I}_d^T\hat{\boldsymbol{P}}_t^d\boldsymbol{I}_d + \boldsymbol{R} \tag{17}$$

5) Output synthesis: The final output of IMM-based MOT Module is synthesized from the three distributed state estimates $\{\hat{\boldsymbol{x}}_t^p, \hat{\boldsymbol{P}}_t^p\}$, $\{\hat{\boldsymbol{x}}_t^v, \hat{\boldsymbol{P}}_t^v\}$ and $\{\hat{\boldsymbol{x}}_t^r, \hat{\boldsymbol{P}}_t^r\}$:

$$\begin{aligned} \hat{\boldsymbol{x}}_t &= w_t^p\hat{\boldsymbol{x}}_t^p + w_t^v\hat{\boldsymbol{x}}_t^v + w_t^r\hat{\boldsymbol{x}}_t^r \\ \hat{\boldsymbol{P}}_t &= \sum_{d\in\{p,v,r\}} w_t^d[\hat{\boldsymbol{P}}_t^d + (\hat{\boldsymbol{x}}_t^d - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^d - \hat{\boldsymbol{x}}_t)^T] \end{aligned} \tag{18}$$

C. IMM-based Graph Optimization Module

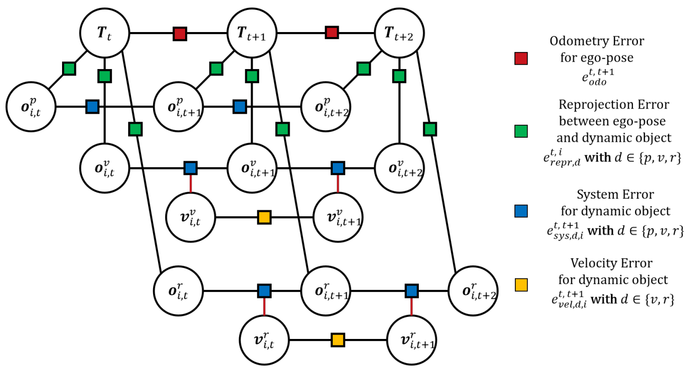

T_t
T_{t+1}
T_{t+2}
$o_{i,t}^p$
$o_{i,t+1}^p$
$o_{i,t+2}^p$
$o_{i,t}^v$
$o_{i,t+1}^v$
$o_{i,t+2}^v$
$v_{i,t}^v$
$v_{i,t+1}^v$
$o_{i,t}^r$
$o_{i,t+1}^r$
$o_{i,t+2}^r$
$v_{i,t}^r$
$v_{i,t+1}^r$
🟥 Odometry Error for ego-pose $e_{odo}^{t,t+1}$
🟩 Reprojection Error between ego-pose and dynamic object $e_{repr,d}^{t,i}$ with $d \in \{p,v,r\}$
🟦 System Error for dynamic object $e_{sys,d,i}^{t,t+1}$ with $d \in \{p,v,r\}$
🟨 Velocity Error for dynamic object $e_{vel,d,i}^{t,t+1}$ with $d \in \{v,r\}$

Fig. 4. IMM-based Factor Graph Representation

We utilize a factor graph to collectively optimize ego-pose and object states. To integrate the IMM algorithm with graph optimization, we introduce nodes for the three models into the factor graph. The error associated with each model edge is then multiplied by a model weight obtained from the IMM-based MOT Module. The overall structure of the IMM-based Graph Optimization Module is illustrated in Fig. 4.

The edge between two consecutive ego-poses represents the odometry error:

$$\boldsymbol{e}_{odo}^{t,t+1} = (\boldsymbol{T}_t^{t+1}\boldsymbol{T}_t)^{-1}\boldsymbol{T}_{t+1} \tag{19}$$

With $\boldsymbol{T}_t$ the ego-pose at the time t and $\boldsymbol{T}_t^{t+1}$ the LiDAR odometry result from the time t to the time $t+1$.

Using $\boldsymbol{p}_{i,t}^d = [x_{i,t}^d, y_{i,t}^d, z_{i,t}^d] \in \mathbb{R}^3$ to represent the position of dynamic object i and $\theta_{i,t}^d$ to denote its yaw rate, $\boldsymbol{o}_{i,t}^d = [x_{i,t}^d, y_{i,t}^d, z_{i,t}^d, \theta_{i,t}^d]^T$, $[\boldsymbol{p}_{i,t}^z, \theta_{i,t}^z]$ represents the corresponding measurement associated with dynamic object i at frame t, and ϕ_t signifies the yaw rate of the ego-pose. The reprojection error of dynamic object i for motion model d, with $d \in \{p, v, r\}$ is:

$$\boldsymbol{e}_{repr,d}^{t,i} = [\boldsymbol{p}_{i,t}^d, \theta_{i,t}^d]^T - [T_t\boldsymbol{p}_{i,t}^z, \phi_t + \theta_{i,t}^z]^T \tag{20}$$

Based on the motion pattern we use, the system error between two consecutive poses of dynamic object i can be expressed as:

$$\boldsymbol{e}_{sys,p,i}^{t,t+1} = [\boldsymbol{p}_{t+1,i}^p, \theta_{t+1,i}^p]^T - [\boldsymbol{p}_{t,i}^p, \theta_{t,i}^p]^T \tag{21}$$

$$\boldsymbol{e}_{sys,v,i}^{t,t+1} = [\boldsymbol{p}_{t+1,i}^v, \theta_{t+1,i}^v]^T - g_v([\boldsymbol{p}_{t,i}^v, \theta_{t,i}^v]^T, \boldsymbol{v}_{t,i}^v) \tag{22}$$

$$\boldsymbol{e}_{sys,r,i}^{t,t+1} = [\boldsymbol{p}_{t+1,i}^r, \theta_{t+1,i}^r]^T - g_r([\boldsymbol{p}_{t,i}^r, \theta_{t,i}^r]^T, \boldsymbol{v}_{t,i}^r) \tag{23}$$

where $\boldsymbol{v}_{t,i}^v = v_{t,i}^v$ and $\boldsymbol{v}_{t,i}^r = [v_{t,i}^r, \omega_{t,i}^r]^T$ denote the linear velocity and angular velocity respectively, g_v and g_r are the transition functions for the CV Model and the CTRV Model respectively:

$$g_v([\boldsymbol{p}_{t,i}^v, \theta_{t,i}^v]^T, \boldsymbol{v}_{t,i}^v) = \begin{bmatrix} x_{t,i}^v + v_{t,i}^v\Delta t\cos(\theta_{t,i}^v) \\ y_{t,i}^v + v_{t,i}^v\Delta t\sin(\theta_{t,i}^v) \\ z_{t,i}^v \\ \theta_{t,i}^v \end{bmatrix} \tag{24}$$

$$g_r([\boldsymbol{p}^r_{t,i}, \theta^r_{t,i}]^T, \boldsymbol{v}^r_{t,i}) = \begin{bmatrix} x^r_{t,i} + v^r_{t,i}\Delta t \cos(\theta^r_{t,i} + \frac{\omega^r_{t,i}\Delta t}{2}) \\ y^r_{t,i} + v^r_{t,i}\Delta t \sin(\theta^r_{t,i} + \frac{\omega^r_{t,i}\Delta t}{2}) \\ z^r_{t,i} \\ \theta^r_{t,i} + \omega^r_{t,i}\Delta t \end{bmatrix} \quad (25)$$

For the CV Model and CTRV Model, velocity errors are also included in the factor graph, leading to:

$$\boldsymbol{e}^{t,t+1}_{vel,v,i} = v^v_{t+1,i} - v^v_{t,i} \quad (26)$$

$$\boldsymbol{e}^{t,t+1}_{vel,r,i} = [v^r_{t+1,i}, \omega^r_{t+1,i}]^T - [v^r_{t,i}, \omega^r_{t,i}]^T \quad (27)$$

Finally, considering the associated model weights, the optimization function can be expressed as:

$$\begin{aligned} \min \sum_t ||\boldsymbol{e}^{t,t+1}_{odo}||^2 + (\sum_i w^p_{t,i}(||\boldsymbol{e}^{t,i}_{repr,p}||^2 + ||\boldsymbol{e}^{t,t+1}_{sys,p,i}||^2) \\ + w^v_{t,i}(||\boldsymbol{e}^{t,i}_{repr,v}||^2 + ||\boldsymbol{e}^{t,t+1}_{sys,v,i}||^2 + ||\boldsymbol{e}^{t,t+1}_{vel,v,i}||^2) \\ + w^r_{t,i}(||\boldsymbol{e}^{t,i}_{repr,r}||^2 + ||\boldsymbol{e}^{t,t+1}_{sys,r,i}||^2 + ||\boldsymbol{e}^{t,t+1}_{vel,r,i}||^2)) \end{aligned} \quad (28)$$

D. The Baseline Method at Methodology Level 2

In our presented works, we would like to propose Methodology Level 3 and demonstrate its methodology advantage over Methodology Level 0 to 2. For fairness of comparative study, we intentionally instantiate the four methodology levels in the way that they share the same composing modules, so that their performance comparison will not be biased by any ad hoc composing module. Therefore, the methodology of the DynaSLAM2 [12], a representative state-of-the-art methodology of SLAMMOT, is adapted for Lidar based realization and then is adopted as our baseline method at Methodology Level 2. It is worth noting that the original DynaSLAM2 is based on vision and cannot be directly applied in our context, yet the core merit of the DynaSLAM2 consists right in its methodology, an advantageous methodology which can be potentially adapted for whatever kinds of sensor configurations. Consequently, in our LiDAR-based scenario, we utilize the same SLAMMOT architecture as our proposed method but implement an EKF-based MOT for object tracking and a factor graph that includes dynamic states for Graph Optimization. The details of these modifications will be described in the following sections.

1) EKF-based MOT Module: We consider an EKF-based Multi-Object Tracking approach that employs the Constant Turning Velocity Model as the system model. The detailed procedure can be outlined in the following two steps:

Prediction:

$$\bar{\boldsymbol{x}}^r_t = \boldsymbol{A}^r_{t-1}\boldsymbol{x}^r_{t-1} \quad , \quad \bar{\boldsymbol{P}}^r_t = \boldsymbol{A}^r_{t-1}\hat{\boldsymbol{P}}^d_{t-1}\boldsymbol{A}^{r\ T}_{t-1} + \boldsymbol{Q}_r \quad (29)$$

According to the predicted state $\bar{\boldsymbol{x}}^r_t$ and predicted covariance $\bar{\boldsymbol{P}}^r_t$, the corresponding measurement $\boldsymbol{z}_t$ can be found by Kuhn–Munkres algorithm.

Update:

$$\begin{aligned} \hat{\boldsymbol{x}}^r_t &= \bar{\boldsymbol{x}}^r_t + \boldsymbol{K}(\boldsymbol{z}_t - \boldsymbol{I}_r\bar{\boldsymbol{x}}^r_t) \\ \hat{\boldsymbol{P}}^r_t &= (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{I}_r)\bar{\boldsymbol{P}}^r_t \\ \boldsymbol{K} &= \bar{\boldsymbol{P}}^r_t(\boldsymbol{I}_r\bar{\boldsymbol{x}}^r_t)^T[(\boldsymbol{I}_r\bar{\boldsymbol{x}}^r_t)\bar{\boldsymbol{P}}^r_t(\boldsymbol{I}_r\bar{\boldsymbol{x}}^r_t)^T + \boldsymbol{R}]^{-1} \end{aligned} \quad (30)$$

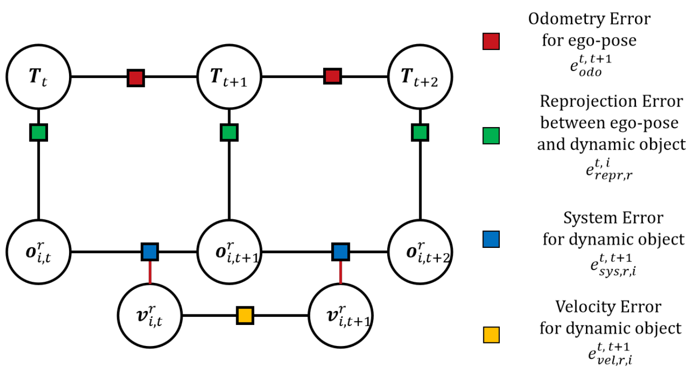

T_t
T_{t+1}
T_{t+2}
$o^r_{i,t}$
$o^r_{i,t+1}$
$o^r_{i,t+2}$
$v^r_{i,t}$
$v^r_{i,t+1}$
🟥 Odometry Error for ego-pose $e^{t,t+1}_{odo}$
🟩 Reprojection Error between ego-pose and dynamic object $e^{t,i}_{repr,r}$
🟦 System Error for dynamic object $e^{t,t+1}_{sys,r,i}$
🟨 Velocity Error for dynamic object $e^{t,t+1}_{vel,r,i}$

Fig. 5. Baseline Factor Graph Representation

2) Baseline Graph Optimization Module: To jointly optimize the dynamic object state and ego-pose estimation, analogous to the IMM-based Factor Graph, we incorporate position nodes and velocity nodes into the factor graph. The representation of the Baseline Graph Optimization is illustrated in Fig. 5. The optimization function for this baseline method is defined as:

$$\min \sum_t ||\boldsymbol{e}^{t,t+1}_{odo}||^2 + (\sum_i ||\boldsymbol{e}^{t,i}_{repr,r}||^2 + ||\boldsymbol{e}^{t,t+1}_{sys,r,i}||^2 + ||\boldsymbol{e}^{t,t+1}_{vel,r,i}||^2) \quad (31)$$

V. EXPERIMENTS

In this section, we provide a comprehensive overview of the experiments conducted to evaluate the IMM-SLAMMOT and the baseline methods. To demonstrate the methodological advantages of IMM-SLAMMOT over Methodology Levels 0 to 2, we incorporate three baseline methods, each corresponding to one of these three methodology levels. For the baseline method at Methodology Level 0, we simply employ LeGO-LOAM and assume that all points in the environment are static. In contrast, for Methodology Level 1, we introduce a preprocessing module capable of filtering out potential moving points in the scene, and the remaining points are then passed to the LiDAR odometry module (LeGO-LOAM). The baseline at Methodology Level 2 is as introduced in Section IV-D. We assess the methods from two perspectives: ego-pose estimation accuracy and MOT performance. Regarding the latter, we analyse performance over the entire tracking duration and over motion-pattern-transition periods, considering that motion-pattern-transition moments are especially critical in real traffics.

Experiments are conducted using synthetic data generated by the CARLA [31] which has been widely adopted for research in the field of intelligent vehicles, and we also tested our method on the KITTI Tracking Dataset. When generating scenes on CARLA, we configure an ego vehicle equipped with a LiDAR sensor mounted on its roof, and introduce a variety of random vehicles and pedestrians into the scene. Simulation scenarios like real traffics are established. In order to facilitate an unbiased comparison between IMM-SLAMMOT and the baseline method, we generated random sequences with varying trajectories in different scenes provided by CARLA. In our experiments, we specifically designed four trajectories across four different scenes. For each trajectory, we generated two sequences with different quantities of dynamic objects. Sequences ending with 'a' included a high quantity of dynamic objects such as vehicles and pedestrians, while sequences ending with 'b' had a much lower quantity of dynamic objects in the scene.

IMM-SLAMMOT and the baseline methods at Methodology Level 1, 2 require instance semantic priors, which can be well provided by state-of-the-art semantic segmentation methods such as the PointRCNN [26] and the CenterPoint [27]. Any further marginal improvement on state-of-the-art semantic segmentation methods would also bring some marginal benefit to SLAMMOT methods. On the other hand, to save experimentation time spent on semantic segmentation yet without influencing fairness of the comparative study, we intentionally use noise-corrupted ground truth as semantic priors for the methods under comparison. After all, any concrete semantic segmentation method itself is neither essential nor indispensable to the SLAMMOT methods.

A. LiDAR Odometry

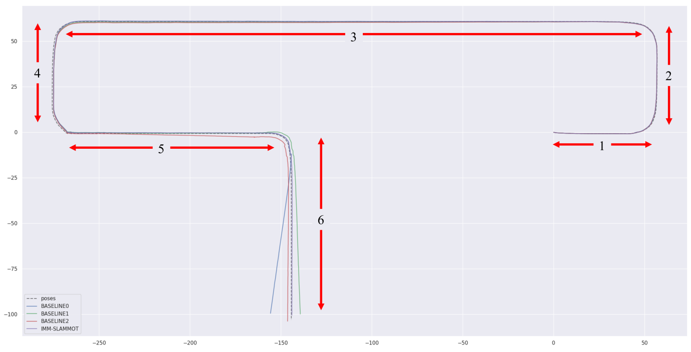


Fig. 6. Comparison of ego-pose estimation for sequence 01a on CARLA Simulated Data

In the LiDAR Odometry experiments, we compare the performance of ego-pose accuracy among methods at four methodology levels. We evaluate the absolute pose error (APE), the relative translation error ($\mathrm{RPE_t}$), and the relative rotation error ($\mathrm{RPE_r}$). These three metrics serve as reliable indicators of ego-pose estimation accuracy throughout the entire tracking period.

1) Results on CARLA Simulated Data: Table I provides a comprehensive performance comparison among our proposed IMM-SLAMMOT and other baseline methods on CARLA Simulated Data and Fig. 6 illustrates trajectory reconstruction comparisons for sequence 01a as an example for visualization. In experimental scenarios, dynamic points constitute smaller part of LiDAR scans compared with static points [1], so the baseline methods at three methodology levels and the IMM-SLAMMOT all achieve competitive results in ego-pose estimation, yet the IMM-SLAMMOT demonstrates marginal advantage.

As mentioned at the beginning of this section, the sequences 'a' and sequences 'b' share the same simulated ego-vehicle trajectory. However, in sequences 'a', we introduce a greater number of dynamic objects, leading to higher relative improvements for IMM-SLAMMOT and the baseline methods at Methodology Level 1, 2 compared to the baseline methods at Methodology Level 0. Therefore, in scenarios with numerous dynamic objects in the scene, IMM-SLAMMOT and the baseline methods at Methodology Level 1, 2 outperform the baseline methods at Methodology Level 0 to a greater extent.

Furthermore, in our generated scenes, dynamic objects predominantly follow stationary motion patterns throughout the entire trajectory. Consequently, both the baseline method at Methodology Level 2 and IMM-SLAMMOT achieve competitive results in the Multi-Object Tracking Module for most of the time. Therefore, the final SLAM results for both the baseline method and IMM-SLAMMOT are also competitive throughout the entire trajectory.

Some extra explanations hover over details of the comparative study example visualized in Fig. 6. The scenario consists of six road segments, as indicated by numbers 1 to 6 in Fig. 6. The road segments 1 to 4 are rather static environments, almost exempt from moving objects. The road segment 5 is with many moving objects and still with considerable amount of static objects as well, whereas the road segment 6 is with even more moving objects but with less static objects. Moving objects on the road segments 5 and 6 have changeable motion patterns. Since the road segments 1 to 4 are almost static, all the four methods perform similarly well on these road segments. On the road segment 5, Methodology Levels 0 and 1 continue to perform desirably, thanks to existence of still considerable amount of static objects. In contrast, Methodology Level 2 starts to suffer from moderate errors due to inconsistency between single motion modelling and actual motion patterns of moving objects that change frequently. On the road segment 6, although Methodology Level 2 still suffers from moderate errors due to its inherent modelling inconsistency, it can somewhat take advantage of state estimation of moving objects to maintain its performance. In contrast, Performances of Methodology Levels 0 and 1 deteriorate apparently, due to insufficiency of static objects on the road segment 6. As we can see, IMM-SLAMMOT outperforms Methodology Levels 0, 1, and 2 in the sense that it can maintain desirable performance on all the road segments from 0 to 6, regardless of environmental changes.

2) Results on KITTI Tracking Dataset: Table II presents a comparative analysis of IMM-SLAMMOT against three baseline methods using the KITTI Tracking Dataset. The performance of IMM-SLAMMOT and the three baseline methods is consistent with their performance on CARLA Simulated Data. Although some scenes in the KITTI Tracking Dataset are relatively simple, allowing all methods to achieve reasonable results, we observed that in scenes with a high quantity of dynamic objects, IMM-SLAMMOT outperforms the methods

[1]In real traffics, dynamic points also tend to constitute smaller part of perceptive data whereas static points constitute larger part. Even when the vehicle is occasionally surrounded by cars in crowded traffics, it may still perceive over surrounding cars considerable part of the static environment. It is in the rare case when the vehicle is incidentally surrounded by big buses and trucks that the occlusion would indeed cause a fatal challenge to all existing single-vehicle SLAM and SLAMMOT methods. Perhaps we need to resort to cooperative perception [32] [33] to handle such kind of fatal challenge.

TABLE I
EGOMOTION COMPARISON ON CARLA SIMULATED DATA

sequences		seq01a	seq01b	seq02a	seq02b	seq03a	seq03b	seq04a	seq04b	Mean
BASELINE0 (Methodology Level 0)	APE (m)	1.762	1.029	1.006	0.815	3.254	1.064	8.772	6.798	3.062
	$\mathrm{RPE_t}$ (m/f)	0.072	0.049	0.063	0.051	0.060	0.075	0.089	0.065	0.065
	$\mathrm{RPE_r}$ (rad/f)	0.004	0.004	0.006	0.005	0.006	0.009	0.009	0.008	0.006
BASELINE1 (Methodology Level 1)	APE (m)	1.570	1.427	0.869	0.801	**2.460**	1.077	6.303	4.756	2.408
	$\mathrm{RPE_t}$ (m/f)	0.055	0.053	**0.060**	0.046	**0.050**	0.074	0.071	0.068	0.060
	$\mathrm{RPE_r}$ (rad/f)	0.004	0.005	**0.006**	0.005	**0.006**	0.010	0.009	0.009	0.007
BASELINE2 (Methodology Level 2)	APE (m)	1.478	0.930	0.855	0.778	2.474	1.044	**6.176**	**4.536**	2.284
	$\mathrm{RPE_t}$ (m/f)	0.050	**0.044**	**0.060**	**0.045**	0.052	0.074	0.073	0.061	0.057
	$\mathrm{RPE_r}$ (rad/f)	**0.003**	**0.004**	**0.006**	0.005	**0.006**	0.009	0.010	**0.008**	**0.006**
IMM-SLAMMOT (Methodology Level 3)	APE (m)	**1.297**	**0.929**	**0.834**	**0.656**	2.517	**0.591**	6.631	4.549	**2.251**
	$\mathrm{RPE_t}$ (m/f)	**0.042**	0.053	0.061	0.047	0.051	**0.057**	**0.062**	**0.060**	**0.054**
	$\mathrm{RPE_r}$ (rad/f)	**0.003**	0.005	0.007	**0.004**	**0.006**	**0.008**	**0.008**	**0.008**	**0.006**

at Methodology Levels 0 to 2.

B. Multi-Object Tracking

Regarding the Multi-Object Tracking results, we compare the IMM-SLAMMOT and baseline method at Methodology Level 2. We evaluate the Euclidean distance based multi-object tracking precision ($\mathrm{MOTP_{dis}}$) and the 3D bounding box intersection over union ($\mathrm{MOTP_{3d}}$). Additionally, we consider the mean absolute pose error (APE) and the mean absolute rotation error ($\mathrm{APE_r}$) for all dynamic objects throughout the entire trajectory or during motion-pattern-transition periods.

Table III presents the MOTP results over the entire trajectory. As previously mentioned, despite the relatively limited occurrence of motion-pattern-transition periods, the baseline method also attains reasonably good results. However, it is worth noting that IMM-SLAMMOT consistently exhibits superior precision throughout the entire trajectory compared to the baseline method.

Table IV further supports this conclusion by detailing the mean pose estimation results for all objects throughout the entire track. For each trajectory, we analyze the behavior of all dynamic objects to assess the mean absolute position error and mean absolute rotation error throughout the entire track. As shown in Table IV, when compared with the baseline method, the IMM-SLAMMOT demonstrates better tracking accuracy, though the improvements are moderate.

In contrast, Table V and Table VI focus on the motion-pattern-transition period, emphasizing the mean accuracy during this phase when the state undergoes transitions between stationary, uniform linear motion, and uniform rotation. Specifically, we extract the motion-pattern-transition period for all objects in all sequences and analyze these periods, including transitions between the Constant Position Model and Constant Velocity Model and the transition between the Constant Velocity Model and Constant Turning Velocity Model. Consequently, during these periods, the positional accuracy and rotation accuracy in IMM-SLAMMOT are significantly higher than those in the baseline method. Therefore, the results demonstrate that IMM-SLAMMOT significantly outperforms the baseline method in position estimation and yaw rate estimation during the motion-pattern-transition period.

C. Computation Overhead

We calculated the average time consumption of the main functional modules, excluding object detection, which can be accelerated by a GPU. The implementation was performed in C++ on a computer equipped with a 3.6GHz i7-12700KF CPU. The LiDAR odometry, object tracking, and graph optimization modules took approximately 8 ms, 0.5 ms, and 112 ms, respectively. In our experiments, we executed the graph optimization at a low frequency (3Hz); therefore, the main SLAMMOT modules (excluding the Input Processing Module) exhibited real-time performance.

D. Discussion

In our experiments, we observe that, on average, i.e., considering the overall trajectory, the IMM-SLAMMOT exhibits competitive performance compared with the baseline methods. Moreover, during motion-pattern-transition periods, the IMM-SLAMMOT significantly outperforms the baseline method at Methodology Level 2. In reality, most traffic accidents tend to occur when the state of dynamic objects changes suddenly. For instance, if the car in front suddenly brakes for some unexpected reason, the vehicle behind often doesn't have sufficient time to react, resulting in a collision. Similarly, when a stationary vehicle at an intersection suddenly starts to cross the road, it can lead to accidents due to the sudden change in the traffic situation. Therefore, our results highlighting the advantages of the IMM-SLAMMOT during motion-pattern-transition periods hold significant implications for real-world applications.

Furthermore, we have observed that in the Graph Optimization Module, when we substitute the IMM-based Factor Graph with the Baseline Factor Graph as depicted in Fig. 5, the performance does not deteriorate. This is because, in the Graph Optimization Module, we treat all ego-pose and dynamic

TABLE II
EGOMOTION COMPARISON ON KITTI TRACKING DATASET

sequences		0	3	6	9	20	Mean
BASELINE0 (Methodology Level 0)	APE (m)	1.727	1.858	0.459	**3.135**	23.429	6.122
	$\mathrm{RPE_t}$ (m/f)	0.047	0.086	0.049	**0.089**	0.143	0.083
	$\mathrm{RPE_r}$ (rad/f)	0.005	**0.003**	**0.002**	0.004	**0.003**	**0.003**
BASELINE1 (Methodology Level 1)	APE (m)	1.717	1.752	**0.403**	**3.790**	8.182	3.169
	$\mathrm{RPE_t}$ (m/f)	0.047	**0.080**	0.046	**0.089**	0.138	0.080
	$\mathrm{RPE_r}$ (rad/f)	0.005	0.003	**0.002**	0.004	**0.003**	0.004
BASELINE2 (Methodology Level 2)	APE (m)	1.715	1.746	0.404	3.584	7.611	3.012
	$\mathrm{RPE_t}$ (m/f)	0.047	0.082	0.045	0.090	**0.139**	0.080
	$\mathrm{RPE_r}$ (rad/f)	0.005	**0.003**	**0.002**	**0.004**	0.003	**0.003**
IMM-SLAMMOT (Methodology Level 3)	APE (m)	**1.713**	**1.745**	0.404	3.151	**7.291**	**2.861**
	$\mathrm{RPE_t}$ (m/f)	**0.045**	**0.080**	**0.044**	**0.089**	**0.139**	**0.079**
	$\mathrm{RPE_r}$ (rad/f)	**0.005**	**0.003**	**0.002**	**0.004**	**0.003**	**0.003**

TABLE III
OBJECT MOTION COMPARISON THROUGHOUT ENTIRE TRAJECTORY

sequences	**BASELINE2** (Methodology Level 2)		**IMM-SLAMMOT** (Methodology Level 3)	
	$\mathrm{MOTP_{dis}}$	$\mathrm{MOTP_{IoU}}$	$\mathrm{MOTP_{dis}}$	$\mathrm{MOTP_{IoU}}$
seq01a	0.219	88.7%	**0.181**	**90.4%**
seq01b	0.187	91.2%	**0.177**	**91.5%**
seq02a	0.208	88.4%	**0.162**	**91.0%**
seq02b	0.260	87.4%	**0.231**	**88.1%**
seq03a	0.282	84.9%	**0.261**	**85.8%**
seq03b	0.362	83.0%	**0.322**	**84.2%**
seq04a	0.274	85.7%	**0.244**	**86.7%**
seq04b	0.363	82.0%	**0.360**	**81.7%**
Mean	0.269	86.4%	**0.242**	**87.4%**

TABLE IV
MEAN OBJECT MOTION COMPARISON THROUGHOUT ENTIRE TRAJECTORY

sequences	**BASELINE2** (Methodology Level 2)		**IMM-SLAMMOT** (Methodology Level 3)	
	APE (m)	$\mathrm{APE_r}$ (rad)	APE (m)	$\mathrm{APE_r}$ (rad)
seq01a	0.210	0.078	**0.180**	**0.032**
seq01b	0.189	0.108	**0.184**	**0.058**
seq02a	**0.187**	0.059	0.199	**0.038**
seq02b	0.221	0.100	**0.189**	**0.037**
seq03a	0.315	0.171	**0.288**	**0.055**
seq03b	0.443	0.264	**0.387**	**0.079**
seq04a	0.282	0.111	**0.252**	**0.043**
seq04b	0.347	0.206	**0.343**	**0.086**
Mean	0.274	0.137	**0.253**	**0.053**

object states as nodes and optimize them collectively using the iSAM2 algorithm whenever new nodes are incorporated into the graph. Consequently, the highest-level motion model (in our case, the Constant Turning Velocity Model) can, to some extent, account for the other two lower-level motion models. So for real-world applications, if execution efficiency is of the most important concern, then readers may just utilize the baseline factor graph in the graph optimization module, though the IMM-based graph optimization module can achieve real-time performance as well.

The most important contribution of our presented works is at SLAMMOT methodology level, namely holistic cooperation between SLAM and MOT considering multiple motion models (via the IMM), i.e. the proposed Methodology Level 3. Considering multiple motion models, which enables the SLAMMOT to effectively handle inherent non-deterministic nature of surrounding moving objects' motion patterns, is the essential point of the proposed methodology.

Concerning the IMM itself, we just adopt it (which is already robust and effective enough in practice) but have no intention to scrutinize any improved variant of it, because this deviates from the focus of the presented works. It is true that any improved variant of the IMM (such as that proposed in [34]) may bring marginal improvement when instantiating the proposed Methodology Level 3, yet the improved variant of the IMM brings no methodology difference concerning the essential point of considering multiple motion models.

VI. CONCLUSION AND FUTURE WORKS

We have introduced an IMM-based SLAMMOT system accompanied by a novel Graph Optimization method tightly integrated with the IMM algorithm. This enables us to track dynamic objects using multiple motion models and optimize the trajectories of both the ego vehicle and the surrounding objects, leading to improved estimation performance and enhanced recognition of motion-pattern-transition periods. Comparing the IMM-SLAMMOT with our three baseline methods, our experiments demonstrate that the IMM-SLAMMOT achieves competitive accuracy for ego-pose estimation and multi-object state estimation. Furthermore, our experiments highlight the IMM-SLAMMOT's superior performance in

TABLE V
OBJECT MOTION COMPARISON DURING MOTION-PATTERN-TRANSITION (CP & CV) PERIOD

sequences	**BASELINE2** (Methodology Level 2)				**IMM-SLAMMOT** (Methodology Level 3)			
	CP to CV		CV to CP		CP to CV		CV to CP	
	APE (m)	$\mathrm{APE_r}$ (rad)	APE	$\mathrm{APE_r}$ (rad)	APE (m)	$\mathrm{APE_r}$ (rad)	APE (m)	$\mathrm{APE_r}$ (rad)
seq01a	0.470	0.005	0.556	0.007	**0.271**	**0.004**	**0.312**	**0.003**
seq01b	0.423	0.007	0.501	0.010	**0.244**	**0.007**	**0.318**	**0.006**
seq02a	0.419	0.004	0.496	0.005	**0.242**	**0.004**	**0.344**	**0.004**
seq02b	0.495	0.007	0.586	0.009	**0.286**	**0.004**	**0.327**	**0.004**
seq03a	0.705	0.012	0.834	0.015	**0.407**	**0.006**	**0.499**	**0.006**
seq03b	0.991	0.018	1.172	0.023	**0.572**	**0.009**	**0.670**	**0.008**
seq04a	0.631	0.008	0.746	0.010	**0.364**	**0.005**	**0.436**	**0.005**
seq04b	0.777	0.014	0.919	0.018	**0.448**	**0.010**	**0.594**	**0.009**
Mean	0.614	0.009	0.726	0.012	**0.354**	**0.006**	**0.438**	**0.006**

TABLE VI
OBJECT MOTION COMPARISON DURING MOTION-PATTERN-TRANSITION (CV & CTRV) PERIOD

sequences	**BASELINE2** (Methodology Level 2)				**IMM-SLAMMOT** (Methodology Level 3)			
	CV to CTRV		CTRV to CV		CV to CTRV		CTRV to CV	
	APE (m)	$\mathrm{APE_r}$ (rad)	APE (m)	$\mathrm{APE_r}$ (rad)	APE (m)	$\mathrm{APE_r}$ (rad)	APE (m)	$\mathrm{APE_r}$ (rad)
seq01a	0.493	0.206	0.475	0.201	**0.291**	**0.064**	**0.249**	**0.062**
seq01b	0.444	0.286	0.427	0.280	**0.296**	**0.117**	**0.253**	**0.113**
seq02a	0.439	0.157	0.423	0.153	**0.321**	**0.078**	**0.274**	**0.075**
seq02b	0.519	0.264	0.500	0.258	**0.305**	**0.075**	**0.261**	**0.072**
seq03a	0.739	0.454	0.712	0.444	**0.464**	**0.112**	**0.397**	**0.108**
seq03b	1.039	0.700	1.000	0.685	**0.623**	**0.159**	**0.533**	**0.153**
seq04a	0.661	0.295	0.637	0.289	**0.406**	**0.087**	**0.347**	**0.084**
seq04b	0.814	0.546	0.784	0.534	**0.553**	**0.174**	**0.473**	**0.167**
Mean	0.644	0.363	0.620	0.356	**0.407**	**0.108**	**0.348**	**0.104**

tracking dynamic objects during motion-pattern-transition periods, a crucial capability for real-world environments.

Nevertheless, the point-based model utilized in the MOT module for multi-object tracking imposes constraints on our ability to accurately model the shapes of real dynamic objects. Additionally, we have yet to address scenarios involving partial observations, where the detected bounding box may be incomplete, leading to measurement position errors relative to the true object position. Fully leveraging the available detection points would expand our capabilities in this regard. Furthermore, we intend to incorporate other sensors such as cameras and IMUs due to their provision of richer environmental information.

REFERENCES

[1] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*, pp. 834–849, Springer, 2014.

[2] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[5] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[6] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and systems*, pp. 1–9, Berkeley, CA, 2014.

[7] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, IEEE, 2018.

[8] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 5135–5142, IEEE, 2020.

[9] J. Huang, S. Yang, Z. Zhao, Y.-K. Lai, and S.-M. Hu, "Clusterslam: A slam backend for simultaneous rigid body clustering and motion estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5875–5884, 2019.

[10] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu, "Clustervo: Clustering moving instances and estimating visual odometry for self and surroundings," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2168–2177, 2020.

[11] J. Zhang, M. Henein, R. Mahony, and V. Ila, "Vdo-slam: a visual dynamic object-aware slam system," *arXiv preprint arXiv:2005.11052*, 2020.

[12] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "Dynaslam ii: Tightly-coupled multi-object tracking and slam," *IEEE robotics and automation letters*, vol. 6, no. 3, pp. 5191–5198, 2021.

[13] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.

[14] H. A. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[15] H. Li, *Fundamentals and applications of recursive estimation theory*. Shanghai Jiao Tong University Press, 2022.

[16] S. Guo, Z. Rong, S. Wang, and Y. Wu, "A lidar slam with pca-based feature extraction and two-stage matching," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

[17] Y.-S. Shin, Y. S. Park, and A. Kim, "Dvl-slam: Sparse depth enhanced direct visual-lidar slam," *Autonomous Robots*, vol. 44, no. 2, pp. 115–130, 2020.

[18] S.-S. Huang, Z.-Y. Ma, T.-J. Mu, H. Fu, and S.-M. Hu, "Lidar-monocular visual odometry using point and line features," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1091–1097, IEEE, 2020.

[19] X. Weng, J. Wang, D. Held, and K. Kitani, "Ab3dmot: A baseline for 3d multi-object tracking and new evaluation metrics," *arXiv preprint arXiv:2008.08063*, 2020.

[20] K. Huang and Q. Hao, "Joint multi-object detection and tracking with camera-lidar fusion for autonomous driving," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6983–6989, IEEE, 2021.

[21] X. Wang, C. Fu, Z. Li, Y. Lai, and J. He, "Deepfusionmot: A 3d multi-object tracking framework based on camera-lidar fusion with deep association," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8260–8267, 2022.

[22] L. Wang, X. Zhang, W. Qin, X. Li, J. Gao, L. Yang, Z. Li, J. Li, L. Zhu, H. Wang, *et al.*, "Camo-mot: Combined appearance-motion optimization for 3d multi-object tracking with camera-lidar fusion," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[23] Z.-J. Du, S.-S. Huang, T.-J. Mu, Q. Zhao, R. R. Martin, and K. Xu, "Accurate dynamic slam using crf-based long-term consistency," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 4, pp. 1745–1757, 2020.

[24] H. Zhang, H. Uchiyama, S. Ono, and H. Kawasaki, "Motslam: Mot-assisted monocular dynamic slam using single-view depth estimation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4865–4872, IEEE, 2022.

[25] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[26] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 770–779, 2019.

[27] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11784–11793, 2021.

[28] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[29] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

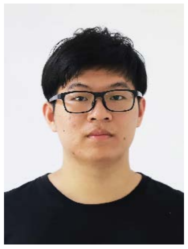
[30] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, no. 46, pp. 3736–3741, 2004.

[31] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[32] H. Li, M. Tsukada, F. Nashashibi, and M. Parent, "Multivehicle cooperative local mapping: a methodology based on occupancy grid map merging," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2089 – 2100, 2014.

[33] S. Fang, H. Li, and M. Yang, "Lidar slam based multivehicle cooperative localization using iterated split cif," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21137 – 21147, 2022.

[34] P. Liu and Z. Duan, "An imm-enabled adaptive 3d multi-object tracker for autonomous driving," in *IEEE International Conference on Information Fusion*, pp. 1–8, 2021.



Zhuoye Ying received the B.Enf. from École d'Ingénieurs SJTU-ParisTech (SPEIT), Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2021. He has been working toward the master degree with École d'Ingénieurs SJTU-ParisTech (SPEIT), Shanghai Jiao Tong University (SJTU), Shanghai, China, since 2021. During 2020 to 2022, he participated in the double-diploma program between SPEIT and École Nationale Supérieure de Techniques Avancées (ENSTA), France, and received an Engineering degree from ENSTA. His research interests include computer vision and autonomous driving.

Hao Li is currently an Assoc. Prof. with the Department of Automation, Shanghai Jiao Tong University (SJTU), China. He obtained the Ph.D. degree from the Robotics Center of MINES ParisTech and INRIA in 2012, the M.Eng. degree and B.Eng. degree from the Department of Automation of SJTU in 2009 and 2006 respectively. His current research interests are in automation, computer vision, data fusion, and cooperative intelligent systems.