IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

DynaMeshSLAM: A Mesh-Based Dynamic Visual SLAMMOT Method

Yang Liu, Chi Guo, Yarong Luo, and Yingli Wang

***Abstract*—In order to estimate both camera poses and dynamic object poses, the visual SLAMMOT method combines visual Simultaneous Localization and Mapping (SLAM) with Multiple Object Tracking (MOT). Many visual SLAMMOT methods represent dynamic objects as bounding boxes and point cloud clusters, which ignores the geometric properties of the object surfaces that can provide additional constraints. In this letter, we propose DynaMeshSLAM, a visual SLAMMOT method, which represents dynamic objects as mesh models to leverage intrinsic geometric properties. Firstly, DynaMeshSLAM fuses the mesh projection and the optical flow to achieve multi-level object data association. Secondly, a constrained mesh smoothing method is embedded into the visual SLAMMOT framework to adjust dynamic landmarks depending on both the smoothness of object mesh models and the projection error of mesh vertices. Thirdly, a bundle adjustment solution incorporating the deformation graph optimizes the states of dynamic objects, while ensuring the local rigidity of the smoothed mesh models. Experiments on the KITTI-Tracking dataset demonstrate that our method achieves state-of-the-art performance in both object tracking and object pose estimation.**

***Index Terms*—SLAM, Visual tracking, semantic scene understanding.**

I. Introduction

VISUAL SLAM is a critical technology in autonomous driving, robotics, and augmented reality/virtual reality (AR/VR). Traditional visual SLAM [1], [2] often relies on the static assumption that only the camera is in motion and the environment is static. However, dynamic objects commonly existing in real-world scenes not only hinder accurate localization but also lead to ambiguous environment reconstruction [3].

Some visual SLAM methods [4], [5] for dynamic scenes consider dynamic features as outliers and remove them, thus effectively improving localization accuracy. However, the motion information of dynamic objects is also important in some scenes. For example, autonomous vehicles control their direction and speed according to the movements of surrounding objects such as vehicles and pedestrians. The motion information of dynamic objects provides clues for dynamic obstacle avoidance and path planning.

Different from traditional visual SLAM methods, the visual SLAMMOT methods estimate the poses of the camera and dynamic objects, while constructing a map that contains the object models and the structure of the static scene. However, most visual SLAMMOT methods model the object as a cluster of point clouds [6], [7] or a cubic bounding box [8], [9], [10]. Since the geometric structure of the object is ignored, some intrinsic geometric constraints of the object cannot be leveraged in the localization and reconstruction. AirDOS [11] models dynamic objects as articulated joints and rods, exploiting the intrinsic rigidity constraints of the objects, but this method requires the extraction of predefined semantic key points.

Object modeling based on visual measurements mainly involves the reconstruction of object surfaces. The mesh model is a common lightweight surface representation that describes the metric geometry of the environment, enabling a visual SLAM system to exploit the structural information to promote localization and mapping. In this letter, we propose DynaMeshSLAM which represents dynamic objects as mesh models. Unlike previous visual SLAMMOT methods, DynaMeshSLAM uses intrinsic geometric properties of dynamic objects to facilitate object data association, reconstruction, and pose estimation. The multi-level object association uses mesh projection to acquire more precise object contours for the instance-level association. A constrained mesh smoothing method is embedded into the visual SLAMMOT framework to obtain a smoother mesh model as well as more accurate dynamic landmarks. In order to maintain the local rigidity of the smoothed mesh models while optimizing object poses and dynamic landmarks, the deformation graph is integrated into object bundle adjustment.

The main contributions of this letter are as follows:

- A robust multi-level object data association method fusing mesh projection and optical flow to achieve instance-level and pixel-level association.
- A constrained mesh smoothing method to obtain smooth mesh models with accurate dynamic landmarks.
- A novel bundle adjustment solution incorporating the deformation graph to facilitate the object pose estimation by leveraging the intrinsic geometry of objects.
- Experiments demonstrate that our proposed methods improve the quality of object tracking and the accuracy of both camera and object pose estimation.

Manuscript received 19 December 2023; accepted 22 April 2024. Date of publication 2 May 2024; date of current version 10 May 2024. This letter was recommended for publication by Associate Editor G. Dubbelman and Editor M. Vincze upon evaluation of the reviewers' comments. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3903801, in part by the Major Science and Technology Project of Hubei Province under Grant 2022AAA009, and in part by China Postdoctoral Science Foundation under Grant 2023TQ0248. *(Corresponding author: Chi Guo.)*

Yang Liu is with the Hubei Luojia Laboratory, Wuhan University, Wuhan 430072, China (e-mail: liuyangmail@whu.edu.cn).

Chi Guo is with the Hubei Luojia Laboratory, Wuhan University, Wuhan 430072, China, also with the GNSS Research Center, Wuhan University, Wuhan 430072, China, and also with the AI Institute, Wuhan University, Wuhan 430072, China (e-mail: chiguo@whu.edu.cn).

Yarong Luo and Yingli Wang are with the GNSS Research Center, Wuhan University, Wuhan 430072, China.

Digital Object Identifier 10.1109/LRA.2024.3396103

II. Related Work

A. Dynamic Visual SLAM

In recent years, many dynamic visual SLAM methods have been proposed, which can be mainly divided into two categories. The first category combines semantic extraction methods (e.g. [12], [13]) and motion consistency checking to reject dynamic feature points directly (e.g. [4], [5]). Although these methods improve localization accuracy, they do not acquire the motion information of dynamic objects.

The second category combines object tracking and visual SLAM to form visual SLAMMOT methods. Co-Fusion [14], Mask-Fusion [15], and Mid-Fusion [16] are all dense SLAM methods with dynamic object tracking. These methods estimate object poses by ICP (Iterative Closest Point) and minimizing photometric error, and reconstruct objects by semantic segmentation or motion segmentation. However, these methods rely on dense point clouds collected from RGB-D data. [8] combines 2D object detection and observation viewpoint classification to infer 3D bounding boxes to estimate object poses in driving scenes. ClusterVO [9] models dynamic objects as point cloud clusters and tracks objects by hierarchical data association. [17] estimates the motion of rigid objects via a formulation on relative object poses. VDO-SLAM [6] jointly optimizes the optical flow and the dynamic object relative poses. DOT [18] performs object pose estimation by minimizing the photometric error, and propagates semantic masks to deal with the tracking loss. DynaSLAM II [10] proposes an object-dominant optimization formulation, which effectively reduces the number of parameters in the joint optimization of the camera and objects. TwistSLAM [7] introduces mechanical constraints between objects, which reduces the degrees of freedom during object pose estimation. Few of the above methods use intrinsic geometric constraints. AirDOS [11] exploits the intrinsic rigidity constraints by decomposing the object into joint nodes and connected rods, and estimates the object motion under articulation constraints. However, the need for predefined semantic key points limits its generalizability.

B. Visual SLAM With Mesh Generation

Visual SLAM can use different types of maps for scene representation. The main types are sparse and dense point cloud maps. Sparse maps store landmarks for localization but lack precise metric information. Dense maps offer detailed scene reconstruction but are costly to store and process. Mesh representation is a compromise form that builds surfaces based on landmark connections to provide environment structure and support localization.

Many visual SLAM systems adopt the mesh representation to obtain structural constraints for localization and mapping. [19] performs Delaunay triangulation on the 2D key points to extract a mesh from the sparse VIO (Visual-Inertial-Odometry) point cloud and detects planes in the mesh to provide structural regularization constraints for VIO optimization. Kimera [20] includes a 3D mesh reconstruction module to maintain a multi-frame mesh in a sliding window. It also adds regularization factors to the VIO backend based on extracted planes from the mesh. [21] uses tracked landmarks obtained from visual odometry and depth measurements to reconstruct a smoother and more accurate mesh via non-smooth convex optimization. Kimera-Multi [22] performs local mesh optimization using a deformation graph to ensure the global consistency between camera poses and the constructed mesh. ElasticFusion [23], in its global optimization, optimizes the map using the deformation graph to maintain the geometric consistency of environment surfaces. Mesh representation is also applied in object-level SLAM. [24] uses semantic key points of objects observed in multiple frames to generate mesh models, and alternately optimize object poses and mesh models. DSP-SLAM [25] uses a priori shape coding to reconstruct an object model and renders the object surface via a differentiable SDF (Signed Distance Function) rendering approach. The aforementioned methods are mainly for static scenes. TwistSLAM++ [26] fuses multiple modalities to track dynamic objects and also fit their SDF models based on LiDAR scans. In this letter, our method emphasizes facilitating dynamic object tracking and pose estimation via mesh models generated from images.

III. Methods

A. System Overview

The framework of DynaMeshSLAM is shown in Fig. 1. The system is implemented based on the stereo mode of ORB-SLAM3 [2] and comprises two parts. The first part is camera pose estimation and static mapping, which provides accurate camera poses and a static map in dynamic scenes. This part follows the algorithm of ORB-SLAM3, except for the addition of dynamic feature elimination. The second part is dynamic object pose estimation and dynamic structure reconstruction, which provides motion information and reconstruction results of dynamic objects. We extract ORB [27] features and adopt an instance segmentation method [12] to extract instance-level semantic information of dynamic objects. The semantic instances are used to filter dynamic ORB features. The remaining static ORB features are associated for camera pose estimation and camera bundle adjustment. The semantic instances are also used to create 2D object observations, which are associated with existing 3D objects via our proposed multi-level object data association. The poses of associated objects are subsequently estimated. The associated objects are then updated and the unassociated 2D object observations are used to initialize new 3D objects. In both updating and initializing, new dynamic feature points are sampled with depth estimated by an efficient stereo matching method [28], and all the observed object feature points are used to generate object mesh models. We then perform a constrained mesh smoothing on each mesh model. After the smoothing, the object states including poses and dynamic landmark positions are jointly optimized in the object bundle adjustment incorporating the deformation graph.

B. Feature Extraction and Data Association

We use ORB features for camera pose estimation and static mapping, following the extraction and association scheme of ORB-SLAM3. In addition, we remove the points belonging to dynamic objects to mitigate their influence on camera pose estimation. Since some priori dynamic objects may be static (e.g., parked cars), we need to determine whether an observed object is dynamic or not before camera pose estimation. We acquire 2D bounding boxes from segmented masks and perform lightweight object tracking by traversing and comparing the intersection over union (IoU) of bounding boxes between two consecutive frames. The tracked objects with a 3D linear velocity close to zero (≤ 0.5 m/s) are recognized as actually static and the ORB

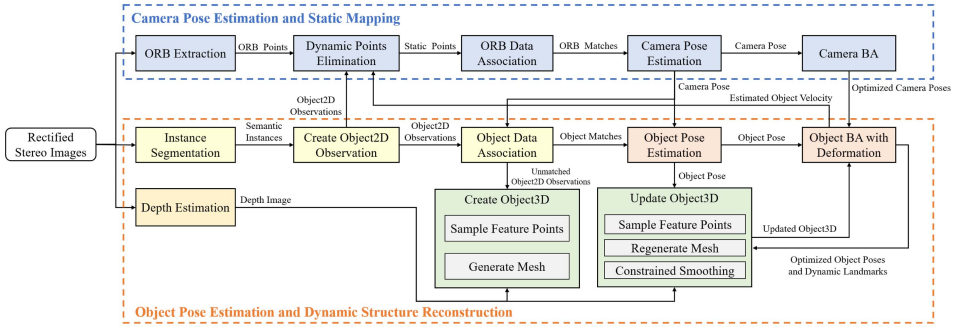


Fig. 1. The system takes rectified stereo images as input and comprises two parts. The part of camera pose estimation and static mapping includes feature extraction and matching, camera pose estimation, and bundle adjustment (BA). We add a dynamic feature elimination module for feature extraction. The part of object pose estimation and dynamic structure reconstruction tracks, reconstructs objects, and performs object pose estimation and object BA.

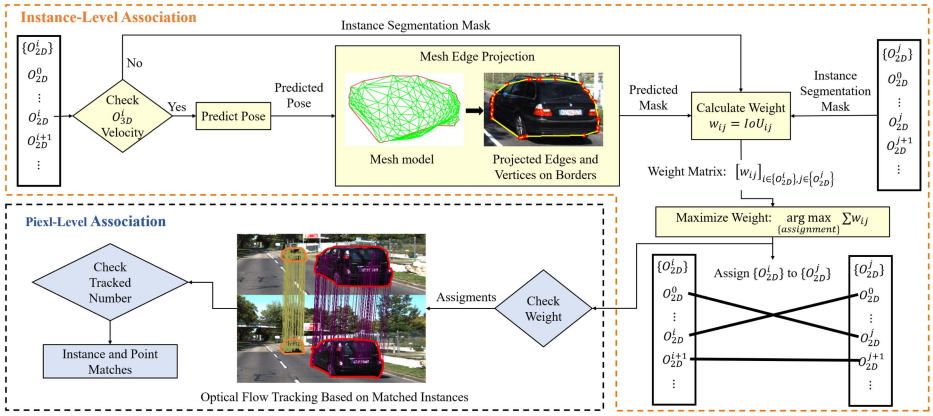


Fig. 2. The multi-level object data association scheme includes instance-level and pixel-level association. In the instance-level association, the red edges on the border of the mesh model can be projected into the current frame via a predicted pose to form a mask for association. In the pixel-level association, sampled points are tracked via optical flow between matched instances and used to check the final association results.

features within their current masks are reserved. Otherwise, we remove these features. This simple strategy ensures that a sufficient number of actual static landmarks contribute to the initial camera pose estimation and subsequent camera bundle adjustment.

Our method tracks common priori dynamic objects (e.g., vehicles), regardless of their motion. For data association, pose estimation, and reconstruction of these prior dynamic objects, we extract both instance-level semantic information and feature points. We use an instance segmentation method [12] to obtain semantic information including classes, bounding boxes, and masks, denoted as O_{2D}, which represents the object observation in image space. The object pose estimation needs pixel-level observations and correspondences over consecutive frames. Considering that ORB features are sparse on some texture-less objects, we directly sample feature points within the dynamic object mask and give each point its corresponding right observed point by a stereo matching method [28]. The point sampling is performed in both object initializing and updating for object structure initialization and supplement of available features, respectively.

Fig. 2 illustrates our multi-level object data association method. The first step is to associate instances over consecutive frames. Specifically, we associate the 2D object observations $\{O^i_{2D}\}$ associated with existing 3D objects $\{O^i_{3D}\}$ in the last frame with the 2D object observations $\{O^j_{2D}\}$ in the current frame. We first check the velocity of a 3D object O^i_{3D} to determine whether its possible variation in image space is large and choose the way to generate a candidate mask for association accordingly. When the linear velocity of a O^i_{3D} surpasses a predefined threshold (we set it to 3 m/s), we utilize a constant velocity model to predict the current object pose and project the edges and vertices on the border of the object mesh model into the current frame to outline the object. The contours are used to fit a polygon mask as the candidate mask. For objects with a velocity below the threshold, we directly use the mask of instance segmentation as the candidate mask. The predicted mask is more consistent with the projection of the high-speed object but requires extra computation, and the segmentation mask is sufficient for tracking low-speed and static objects. We adopt this adaptive way to balance efficiency. Then we calculate IoUs between the candidate masks of $\{O^i_{2D}\}$ and masks of $\{O^j_{2D}\}$ in the current frame to create a weight matrix describing the overlap of every pair of masks. We solve the weight maximization problem via the Kuhn–Munkres (KM) algorithm [29] to assign $\{O^i_{2D}\}$ to $\{O^j_{2D}\}$ and discard the assignments with a weight below a certain threshold (we set it to 0.1). Note that when occlusion causes an object to be lost in tracking and it reappears afterward, it is treated as a new one. The next step is to acquire

pixel-level correspondences based on the matched instances. For each assignment (O^i_{2D}, O^j_{2D}), we use Lucas-Kanade optical flow [30] to track feature points of O^i_{2D}. If the number of tracked points within the mask of O^j_{2D} is greater than a threshold, the assignment will be accepted and the point correspondences will be saved. In the above process, the IoU-based assignment ensures accurate instance-level association, while optical flow tracking provides pixel-level association and validates the final association results. Without using CNN-based dense optical flow, our method employs this multi-level scheme to achieve coarse-to-fine object data association.

C. Camera and Object Pose Estimation

For the camera, we perform pose estimation and bundle adjustment by minimizing the re-projection error. The estimated camera pose at time t_i is denoted as $\mathbf{T}^i_{cw} \in SE(3)$. Through data association, we obtain the l-th static landmark $\mathbf{P}^l_w \in \mathbb{R}^3$ in the world coordinates and its corresponding feature point $\mathbf{z}^{i,l} = [u^{i,l}_l, v^{i,l}_l, u^{i,l}_r]^T \in \mathbb{R}^3$, consisting of pixel coordinates in the left image and the horizontal coordinate in the right image. The camera re-projection error is as follows:

$$\mathbf{e}^{i,l}_{proj} = \mathbf{z}^{i,l} - \pi(\mathbf{T}^i_{cw}\overline{\mathbf{P}}^l_w) \tag{1}$$

where $\pi(\cdot)$ projects the homogeneous coordinates of a landmark in the camera coordinates onto the image plane using intrinsic parameters, with $\overline{(\cdot)} \in \mathbb{R}^4$ denoting the homogeneous coordinates of a landmark.

The optimization problem for camera pose estimation (2) and bundle adjustment (3) are as follows:

$$\mathbf{T}^i_{cw}{}^* = \underset{\mathbf{T}^i_{cw}}{\arg\min} \sum_{l\in\mathcal{M}_i} \|\mathbf{e}^{i,l}_{proj}\|^2_{\Sigma^{i,l}_p} \tag{2}$$

$$\{\mathbf{T}^i_{cw}, \mathbf{P}^l_w\}^* = \underset{\{\mathbf{T}^i_{cw}, \mathbf{P}^l_w\}^*}{\arg\min} \sum_{i\in\mathcal{W}} \sum_{l\in\mathcal{M}_i} \|\mathbf{e}^{i,l}_{proj}\|^2_{\Sigma^{i,l}_p} \tag{3}$$

where $\mathcal{M}_i$ is the set of association results of static landmarks and ORB features at time t_i, $\mathcal{W}$ is a local window established according to co-visible key frames, and $\Sigma^{i,l}_p$ is the covariance matrix related to the image pyramid level.

For objects, we establish the object coordinate system for their pose estimation. When initializing a new 3D dynamic object O_{3D} using a O_{2D}, the sampled feature points are back-projected into space to form an initial point cloud. The new O_{3D} has an initial pose $\mathbf{T}_{wo} \in SE(3)$, where the rotational part is the identity rotation, and the translation part is the centroid of the initial point cloud. This initial pose determines the initial object coordinate system of O_{3D}, with its origin at the point cloud centroid and its axes aligned with the world coordinate system. The object pose estimation is to estimate the relative poses of the object coordinate system with respect to the world coordinate system.

When a O_{2D} is associated with an existing O_{3D}, we obtain the correspondences between dynamic landmarks and feature points. Then we perform the object pose estimation by minimizing the re-projection error. The estimated pose of the j-th object O^j_{3D} at time t_i is denoted as $\mathbf{T}^{i,j}_{wo} \in SE(3)$. The k-th dynamic landmark of O^j_{3D} in the object coordinates is denoted as $\mathbf{P}^{j,k}_o \in \mathbb{R}^3$, and its image feature point is denoted as $\mathbf{z}^{i,j,k} \in \mathbb{R}^3$.

Algorithm 1: Constrained Mesh Smoothing.

Input: $\mathbf{V}^j_{origin} = \{\mathbf{P}^{j,k}_{old}\}$-The mesh vertices of the O^j_{3D}, $\mathbf{I}^j = \{(a^{j,n}, b^{j,n}, c^{j,n})\}$-The vertex indices of triangle surfaces of the mesh model, $\mathbf{Z}^j = \{\mathbf{z}^{i,j,k}\}$-The observed feature points of vertices in a temporal window
Output: $\mathbf{V}^j_{new} = \{\mathbf{P}^{j,k}_{new}\}$-The adjusted mesh vertices

$\mathbf{V}^j_{smooth} \leftarrow \emptyset$
for $\mathbf{P}^{j,k}_{old}$ in $\mathbf{V}^j_{origin}$ **do**
  if $\mathbf{P}^{j,k}_{old}$ is on the Border **then**
    continue
  else
    $\mathbf{V}^{j,k}_{neighbor} \leftarrow \emptyset$
    $\mathbf{V}^{j,k}_{neighbor} \leftarrow NeighborVertices(k, \mathbf{V}^j_{origin}, \mathbf{I}^j)$
    $\mathbf{P}^{j,k}_{smooth} = WeightedAverage(\mathbf{V}^{j,k}_{neighbor})$
    Add $\mathbf{P}^{j,k}_{smooth}$ to $\mathbf{V}^j_{smooth}$
  end if
end for
$\mathbf{G} = BuildFactorGraph(\mathbf{Z}^j, \mathbf{V}^j_{smooth}, \mathbf{V}^j_{origin})$
$\mathbf{V}^j_{new} \leftarrow Optimize(\mathbf{G})$

The object re-projection error is as follows:

$$\mathbf{e}^{i,j,k}_{proj} = \mathbf{z}^{i,j,k} - \pi(\mathbf{T}^i_{cw}\mathbf{T}^{i,j}_{wo}\overline{\mathbf{P}}^{j,k}_o) \tag{4}$$

The optimization problem for object pose estimation is as follows:

$$\mathbf{T}^{i,j}_{wo}{}^* = \underset{\mathbf{T}^{i,j}_{wo}}{\arg\min} \sum_{k\in\mathcal{M}_{i,j}} \|\mathbf{e}^{i,j,k}_{proj}\|^2_{\Sigma^{i,j,k}_p} \tag{5}$$

where $\mathcal{M}_{i,j}$ is the set of association results of dynamic landmarks and tracked feature points, and $\Sigma^{i,j,k}_p$ is the empirical covariance matrix of the re-projection error. The object bundle adjustment will be introduced in Section III-E.

D. Object Mesh Generation and Constrained Smoothing

Inspired by [19], we do not directly generate 3D object mesh models from point clouds. Instead, we perform 2D Delaunay triangulation on object feature points and then back-project the generated 2D mesh into 3D space to form the 3D mesh model. The dynamic landmarks corresponding to these feature points become the mesh vertices. Due to the motion of dynamic objects, it is difficult to track all mesh vertices and incrementally update a mesh model with reasonable topology and global consistency. Therefore, when a new frame is available, we regenerate the mesh model based on tracked and newly sampled feature points in object updating. This mesh model describes the latest observable geometry of a dynamic object and the mesh vertices are stable dynamic landmarks tracked in a local temporal window.

Considering that the object mesh model usually consists of smooth surfaces without sharp burrs [31], we correct dynamic landmarks by smoothing the object mesh model. The Laplacian smoothing [32] is widely used for mesh smoothing in 3D modeling, which uniformly adjusts each vertex to the average position of its neighbor vertices. However, in our visual SLAMMOT framework, the mesh vertices also serve as dynamic landmarks for object pose estimation, thus applying uniform smoothing to all vertices will negatively affect the projection consistency.

To take into account the projection consistency, we propose a constrained smoothing method described in Algorithm 1.

Algorithm 1 uses point positions generated by weighted Laplacian smoothing. For a mesh vertex $\mathbf{P}_{old}^{j,k}$, the $Weighted\ Average(\cdot)$ function is to calculate the smoothed point $\mathbf{P}_{smooth}^{j,k}$ which is the weighted average of the set of its neighbor vertices $\mathbf{V}_{neighbor}^{j,k}$ and given as follows:

$$\mathbf{P}_{smooth}^{j,k} = \frac{\sum_{\mathbf{P}_{old}^{n} \in \mathbf{V}_{neighbor}^{j,k}} w^n \mathbf{P}_{old}^{n}}{\sum_{\mathbf{P}_{old}^{n} \in \mathbf{V}_{neighbor}^{j,k}} w^n}$$

$$w^n = \frac{\|\mathbf{P}_{old}^{j,k} - \mathbf{P}_{old}^{n}\|^{-1}}{\sum_{\mathbf{P}_{old}^{m} \in \mathbf{V}_{neighbor}^{j,k}} \|\mathbf{P}_{old}^{j,k} - \mathbf{P}_{old}^{m}\|^{-1}} \tag{6}$$

where the weight w^n depends on the distance between the neighbor vertex and the smoothed vertex. Closer vertices contribute more to the smoothing operation because they are more likely to be in the same plane as the smoothed vertex.

The optimization in Algorithm 1 involves the re-projection error as (4) for the projection consistency and the smoothing term as follows for the smoothness of the mesh model:

$$\mathbf{e}_{smooth}^{j,k} = \mathbf{P}_{smooth}^{j,k} - \mathbf{P}_{o}^{j,k} \tag{7}$$

The whole optimization problem for the constrained smoothing is given as follows:

$$\{\mathbf{P}_{o}^{j,k}\}^* = \underset{\{\mathbf{P}_{o}^{j,k}\}}{\arg\min} \sum_{\mathbf{P}_{o}^{j,k} \in \mathbf{V}_{origin}^{j}} \sum_{i \in \mathcal{W}_{j,k}} \|\mathbf{e}_{proj}^{i,j,k}\|_{\Sigma_p^{i,j,k}}^2 + \sum_{\mathbf{P}_{smooth}^{j,k} \in \mathbf{V}_{smooth}^{j}} \|\mathbf{e}_{smooth}^{j,k}\|_{\Sigma_s^{j,k}}^2 \tag{8}$$

where $\mathbf{V}_{origin}^{j}$ is the set of mesh vertices of O_{3D}^{j}, $\mathcal{W}_{j,k}$ is a temporal window storing observed feature points, and $\mathbf{V}_{smooth}^{j}$ is the set of points generated from the weighted Laplacian smoothing algorithm. $\Sigma_p^{i,j,k}$ is the same as in (5) and $\Sigma_s^{j,k}$ is the diagonal weight matrix to balance the influence of smoothing (we set it to an identity matrix).

Note that Algorithm 1 is executed iteratively. If border vertices are iteratively adjusted according to (6), the mesh model will shrink incorrectly. To avoid this, we determine whether a vertex is on the border by checking if it belongs to an edge shared by two triangular surfaces. The border vertices are optimized only using the re-projection error.

E. Object Bundle Adjustment Incorporating the Deformation Graph

Object-related variables are estimated based on camera poses and do not introduce additional information to camera pose estimation unless an internal constraint, like the object constant velocity model, is applied. However, this model is mainly assumed to prevent abrupt changes in object motion estimation and does not necessarily aid camera pose optimization. Hence, we separate camera and object bundle adjustment. In our implementation of object state optimization, we fix camera poses to perform object-only bundle adjustment, which involves several error terms and can be represented as a factor graph shown in Fig. 3.

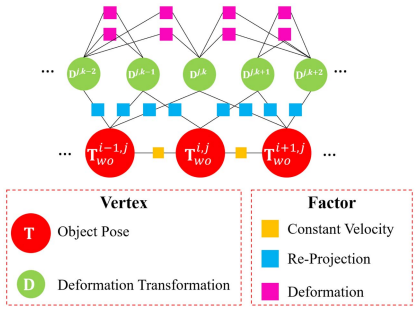


Fig. 3. The factor graph corresponding to the object bundle adjustment.

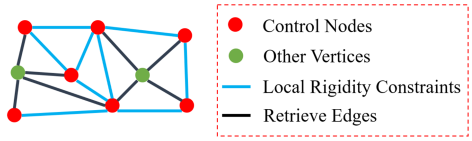


Fig. 4. The integrated deformation graph in object bundle adjustment. Blue edges denote local rigid constraints between control nodes (red), as in (11). Black edges denote the retrieve relations of control nodes to neighbor vertices (green), as in (14).

As object motion does not undergo abrupt changes, we use a constant velocity model to smooth the object trajectory. This model constrains the relative pose of two consecutive frames to be consistent with the relative pose between the previous two frames. The poses of O_{3D}^{j} at time t_{i-1}, t_i, and t_{i+1} are denoted as $\mathbf{T}_{wo}^{i-1,j}$, $\mathbf{T}_{wo}^{i,j}$, and $\mathbf{T}_{wo}^{i+1,j}$ respectively. The error term for the constant velocity model is as follows:

$$\mathbf{e}_{const}^{i,i+1,j} = Log[(\Delta\check{\mathbf{T}}_{wo}^{i-1,i,j})^{-1}\Delta\mathbf{T}_{wo}^{i,i+1,j}] \tag{9}$$

$$\Delta\check{\mathbf{T}}_{wo}^{i-1,i,j} = (\check{\mathbf{T}}_{wo}^{i-1,j})^{-1}\check{\mathbf{T}}_{wo}^{i,j}, \Delta\mathbf{T}_{wo}^{i,i+1,j} = (\mathbf{T}_{wo}^{i,j})^{-1}\mathbf{T}_{wo}^{i+1,j} \tag{10}$$

where $\check{(\cdot)}$ represents the fixed priori states and $Log(\cdot)$ represents the logmap from $SE(3)$ to $se(3)$.

We integrate the deformation graph [33] into object bundle adjustment to maintain the local rigidity of the smoothed mesh model while optimizing dynamic landmarks and object poses. The k-th dynamic landmark of O_{3D}^{j} in the object coordinates is denoted as $\mathbf{P}_{o}^{j,k}$, also a vertex of the mesh model. Considering that dynamic landmarks that have been tracked multiple times have more observations, we first select these landmarks as control nodes for the deformation graph. We then take vertices without connections to any control nodes as control nodes as well to ensure that all vertices can be optimized. Each control node $\mathbf{P}_{o}^{j,k}$ is assigned a transformation $\mathbf{D}^{j,k} = (\mathbf{R}^{j,k} \in SO(3), \mathbf{t}^{j,k} \in \mathbb{R}^3)$ with $\mathbf{R}^{j,k} \in SO(3)$ and $\mathbf{t}^{j,k}$ initialized to the identity and zero respectively, which defines a local coordinate system to map a vertex to the deformed position. The index set of the nodes connected to $\mathbf{P}_{o}^{j,k}$ is denoted as $\mathcal{N}(k)$. The relative position between a node $\mathbf{P}_{o}^{j,n}$ with $n \in \mathcal{N}(k)$ and $\mathbf{P}_{o}^{j,k}$ forms an intrinsic geometric constraint related to the local rigidity of the mesh model, as in Fig. 4. The corresponding error term is as follows:

$$\mathbf{e}_{def}^{j,k} = \mathbf{R}^{j,k}(\mathbf{P}_{o}^{j,n} - \mathbf{P}_{o}^{j,k}) - [(\mathbf{P}_{o}^{j,n} + \mathbf{t}^{j,n}) - (\mathbf{P}^{j,k} + \mathbf{t}^{j,k})] \tag{11}$$

For the nodes in the deformation graph, the re-projection error term is rewritten as follows:

$$\mathbf{e}_{proj'}^{i,j,k} = \mathbf{z}^{i,j,k} - \pi(\mathbf{T}_{cw}^{i}\mathbf{T}_{wo}^{i,j}\overline{(\mathbf{P}_{o}^{j,k} + \mathbf{t}^{j,k})}) \tag{12}$$

After the optimization, the position of a node vertex is updated as follows:

$$\tilde{\mathbf{P}}_{o}^{j,k} = \mathbf{P}_{o}^{j,k} + \mathbf{t}^{j,k} \tag{13}$$

As shown in Fig. 4, retrieving the position of a vertex which is not a node depends on its neighbor nodes and is as follows:

$$\tilde{\mathbf{P}}_{o}^{j,k} = \sum_{n\in\mathcal{N}(k)} w_k(n)[\mathbf{R}^{j,n}(\mathbf{P}_{o}^{j,n} - \mathbf{P}_{o}^{j,k}) + \mathbf{P}_{o}^{j,n} + \mathbf{t}^{j,n}] \tag{14}$$

where $w_k(n)$ is the weight corresponding to the distance between $\mathbf{P}_{o}^{j,n}$ with $n \in \mathcal{N}(k)$ and $\mathbf{P}_{o}^{j,k}$:

$$w_k(n) = (1 - \|\mathbf{P}_{o}^{j,k} - \mathbf{P}_{o}^{j,n}\|/d_{\max})^2 \tag{15}$$

where $d_{\max}$ is the maximum distance to the neighbor node and the weights are then normalized to sum to 1.

The optimization problem is as follows:

$$\begin{aligned}\{\mathbf{T}_{wo}^{i,j}, \mathbf{D}^{j,k}\}^* = \underset{\{\mathbf{T}_{wo}^{i,j}, \mathbf{D}^{j,k}\}}{\arg\min} &\sum_{j\in\mathcal{O}} \sum_{(i,i+1)\in\mathcal{W}_j} \|\mathbf{e}_{const}^{i,i+1,j}\|^2_{\Sigma_c^{i,i+1,j}} \\ &+ \sum_{j\in\mathcal{O}} \sum_{i\in\mathcal{W}_j} \sum_{k\in\mathcal{M}_{i,j}} \|\mathbf{e}_{proj'}^{i,j,k}\|^2_{\Sigma_p^{i,j,k}} + \sum_{j\in\mathcal{O}} \sum_{k\in\mathcal{D}_j} \|\mathbf{e}_{def}^{j,k}\|^2_{\Sigma_d^{j,k}}\end{aligned} \tag{16}$$

where $\mathcal{O}$ is the set of O_{3D} which are observed in the current frame. $\mathcal{W}_j$ is a local temporal window and we only consider the observations and states within the latest N (we set it to 15) frames, $\mathcal{M}_{i,j}$ is the set of point association results of O_{3D}^{j} at time t_i, $\mathcal{D}_j$ is the set of deformation constraints established according to the current mesh model. $\Sigma_p^{i,j,k}$ is the same as in (5). $\Sigma_c^{i,i+1,j}$ and $\Sigma_d^{j,k}$ are the diagonal weight matrices (we set both to identity matrices) to balance the errors. With multiple objects tracked, we put the variables of each object into one factor graph according to (16) and perform the bundle adjustment for all objects simultaneously. We use g2o [34] to solve the optimization problems in this letter.

IV. Experiments

A. Experiments Setup

Datasets: We use the KITTI-Tracking [35] dataset to evaluate our method. The dataset consists of rectified stereo images collected in driving scenarios where many dynamic objects exist. The dataset provides the ground truth for camera poses, 2D bounding boxes, and 3D bounding boxes of objects, which can be used to generate ground truth for object poses. Note that we combine 'truck', 'van', and 'car' in the dataset into one class 'cars', and assume cars to be the common priori dynamic objects for our experiments.

Metrics: For both camera and object pose estimation, we use absolute translation error (ATE) and relative pose error (RPE) for evaluation. RPE is divided into the translation part $\mathrm{RPE_t}$ and the rotation part $\mathrm{RPE_r}$. Since our method focuses on mesh-based object tracking in image space and does not generate 3D bounding boxes, we evaluate the quality of object tracking according to the 2D True Positive Rate (TP) and 2D Multi-Object Tracking Precision (MOTP), where the 2D bounding boxes for evaluation are fitted via object masks.

B. Camera Pose Estimation

We evaluate camera pose estimation on selected sequences in which the camera is moving. We first compare our method with the baseline method ORB-SLAM3. The results of ORB-SLAM3 are from our running in the stereo mode. The comparison results in Table I demonstrate that our method outperforms ORB-SLAM3 on several sequences, especially those with many dynamic objects (e.g., seq11, seq18, and seq20). Our method reduces the ATE by 17%, 14%, and 31% on these sequences respectively. On other sequences with fewer dynamic objects, the baseline method and our method achieve similar performance. We also compare our method with other dynamic visual SLAM methods, where DynaSLAM and DynaSLAM II report both ATE and RPE and the others only report RPE. The comparison results shown in Table I demonstrate that our method achieves higher accuracy on most sequences. DynaSLAM directly eliminates features within priori dynamic object masks, which results in accuracy degradation when many priori dynamic objects are actually static in the scene (e.g., seq01, seq07, seq09). Though other SLAMMOT methods treat dynamic landmarks with a speed close to zero as static in joint bundle adjustment, many actual static landmarks have been ignored in initial camera pose estimation, which causes an accuracy loss of ego-localization. As mentioned in Section III-B, our method eliminates dynamic points with object velocity checking before camera pose estimation, which reserves more actual static landmarks for the ego-localization framework. We argue that this scheme can reduce the performance degradation of a visual SLAMMOT system when facing many priori dynamic objects that are actually static.

C. Object Tracking and Pose Estimation

We follow DynaSLAM II [10] and TwistSLAM [7] to select several significant dynamic objects for object tracking and pose estimation evaluation. In Table II, we compare results from our method with and without the smoothing-deformation bundle adjustment. The results show that the tracking quality is close but our mesh-based method improves the accuracy of both the absolute and relative translation estimation. The improvement in the relative rotation is limited and even slightly degraded on a few objects. We argue that the manipulation of a distant mesh model affects the consistency of rotation estimation. The comparison results with DynaSLAM II and TwistSLAM are also shown in Table II. In terms of tracking quality, our 2D TPs are close to or better than other methods, while our 2D MOTPs are notably superior. That is because we predict masks via the mesh models, which recover accurate contours of tracked objects, as shown in Fig. 5. We also give a visible example of the effect of constrained smoothing. As shown in Fig. 6, some points of the burr shape are corrected in the smoothed mesh. As for object pose estimation, our method outperforms DynaSLAM II on most sequences. DynaSLAM II uses sparse ORB features to track objects but it is difficult to extract and match enough points of objects which are texture-less or far away from the camera (e.g. 05-31, 10-0), potentially leading to insufficient constraints on object pose estimation. Thanks to our multi-level object data association, we can track more available feature points for pose

TABLE I
CAMERA POSE ESTIMATION COMPARED WITH OTHER VISUAL SLAM METHODS ($\mathrm{ATE}:\mathrm{m}, \mathrm{RPE_t}:\mathrm{m/f}, \mathrm{RPE_R}:^\circ/\mathrm{f}$)

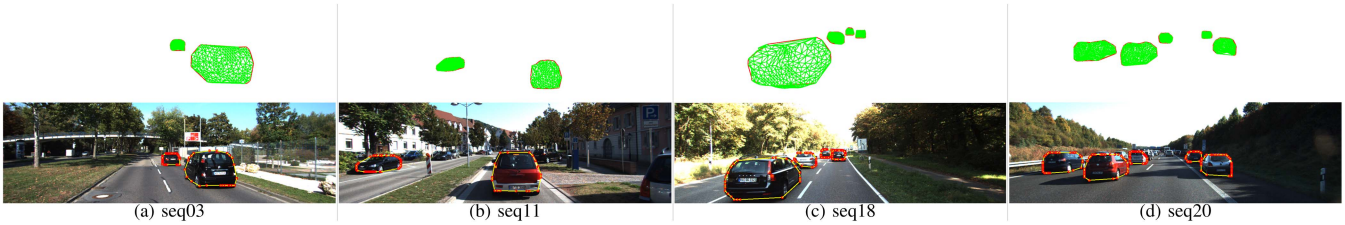
seq	ORB-SLAM3			DynaSLAM			VDO-SLAM			DynaSLAM II			TwistSLAM			Ours		
	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$
00	**1.27**	**0.04**	0.06	1.35	**0.04**	0.06	/	0.05	**0.05**	1.29	**0.04**	0.06	/	**0.04**	**0.05**	**1.27**	**0.04**	**0.05**
01	1.95	0.05	0.04	2.42	0.05	0.04	/	0.12	0.04	2.31	0.05	0.04	/	**0.04**	**0.03**	**1.90**	**0.04**	0.04
02	0.92	0.04	**0.02**	1.04	0.04	0.03	/	0.04	**0.02**	**0.91**	0.04	**0.02**	/	**0.03**	0.03	**0.91**	**0.03**	**0.02**
03	0.80	0.07	0.03	0.78	0.07	0.04	/	0.09	0.04	**0.69**	**0.06**	0.04	/	**0.06**	**0.02**	0.82	**0.06**	**0.02**
04	1.46	0.07	0.05	1.52	0.07	0.06	/	0.11	0.05	1.42	0.07	0.06	/	**0.06**	**0.04**	**1.40**	**0.06**	0.05
05	1.37	0.07	0.03	**1.22**	**0.06**	0.03	/	0.10	**0.02**	1.34	**0.06**	0.03	/	**0.06**	**0.02**	1.56	**0.06**	**0.02**
06	0.18	**0.02**	**0.04**	0.19	**0.02**	**0.04**	/	**0.02**	0.05	0.19	**0.02**	**0.04**	/	**0.02**	**0.04**	**0.15**	**0.02**	**0.04**
07	2.38	0.05	0.07	2.69	0.05	0.07	/	/	/	3.10	0.05	0.07	/	**0.04**	**0.04**	**2.32**	0.05	0.06
08	1.36	0.08	0.04	**1.29**	0.08	0.04	/	/	/	1.68	0.10	0.04	/	**0.07**	**0.03**	1.44	**0.07**	**0.03**
09	3.33	0.06	0.05	3.55	0.06	0.05	/	/	/	5.02	0.06	0.06	/	**0.05**	**0.04**	**3.32**	**0.05**	0.05
10	1.52	**0.07**	**0.03**	1.84	**0.07**	0.04	/	/	/	**1.30**	**0.07**	**0.03**	/	**0.07**	**0.03**	1.47	**0.07**	**0.03**
11	0.87	0.04	0.03	1.05	0.04	0.03	/	/	/	1.03	0.04	0.03	/	**0.03**	**0.02**	**0.72**	**0.03**	**0.02**
13	0.82	**0.03**	**0.04**	1.18	0.04	0.05	/	/	/	1.10	0.04	**0.04**	/	**0.03**	**0.04**	**0.78**	**0.03**	**0.04**
14	0.12	**0.03**	0.08	0.13	**0.03**	0.08	/	/	/	0.12	**0.03**	0.08	/	**0.03**	**0.06**	**0.11**	**0.03**	**0.06**
18	1.06	0.05	0.03	1.00	0.05	0.03	/	0.07	**0.02**	1.09	0.05	**0.02**	/	**0.04**	**0.02**	**0.91**	**0.04**	**0.02**
19	**2.08**	0.05	**0.03**	2.35	0.05	**0.03**	/	/	/	2.25	0.05	**0.03**	/	**0.03**	**0.03**	**2.08**	0.04	**0.03**
20	12.52	0.09	0.05	**1.10**	0.05	0.04	/	0.16	**0.03**	1.36	0.07	0.04	/	**0.04**	**0.03**	8.61	0.07	0.04

The bold values represent the best results in comparison.

TABLE II
OBJECT POSE ESTIMATION COMPARED WITH OTHER VISUAL SLAMMOT METHODS ($\mathrm{ATE}:\mathrm{m}, \mathrm{RPE_t}:\mathrm{m/f}, \mathrm{RPE_R}:^\circ/\mathrm{f}, \mathrm{TP}:\%, \mathrm{MOTP}:\%$)

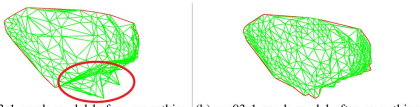
	DynaSLAM II					TwistSLAM					Ours (Motion-Only)					Ours (Smoothing-Deform)				
seq-obj. id	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$	2D-TP	2D-MOTP	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$	2D-TP	2D-MOTP	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$	2D-TP	2D-MOTP	ATE	$\mathrm{RPE_t}$	$\mathrm{RPE_R}$	2D-TP	2D-MOTP
03-1	0.69	0.34	1.84	50.00	71.79	0.31	0.10	**0.28**	**58.02**	60.00	0.52	0.09	0.71	42.62	**83.84**	**0.28**	**0.07**	0.39	42.62	**83.84**
05-31	0.51	0.26	13.50	28.96	60.30	0.35	0.19	0.58	30.84	35.00	0.31	0.07	**0.24**	**33.33**	**76.59**	**0.16**	**0.06**	0.48	**33.33**	**76.59**
10-0	0.95	0.40	2.84	**81.63**	73.51	0.77	0.21	1.98	7.20	3.70	0.75	0.25	0.35	75.17	**83.60**	**0.59**	**0.11**	**0.32**	75.17	**83.60**
11-0	1.05	0.43	12.51	**72.65**	74.78	**0.17**	0.23	**0.23**	29.61	32.50	0.82	0.15	0.38	59.79	**83.51**	0.54	**0.12**	1.66	59.79	**83.51**
11-35	1.25	0.89	16.64	53.17	65.25	**0.10**	**0.03**	**0.11**	**65.00**	67.50	0.36	0.05	0.15	62.50	**92.32**	0.12	0.04	0.25	62.50	**92.32**
18-2	1.10	0.30	9.27	86.36	74.81	**0.21**	0.27	0.66	84.67	87.50	0.53	**0.06**	0.40	**89.39**	**87.83**	0.30	**0.06**	**0.27**	**89.39**	**87.83**
18-3	1.13	0.55	20.05	53.33	70.94	0.15	0.21	0.56	28.19	30.00	0.56	0.11	**0.26**	**53.68**	**78.00**	**0.13**	**0.10**	0.63	**53.68**	**78.00**
19-63	0.86	1.45	48.80	35.26	63.50	**0.28**	2.17	1.08	**65.93**	70.00	0.45	0.09	0.50	64.16	**79.08**	0.33	**0.08**	**0.35**	65.32	79.07
19-72	0.99	1.12	3.36	**29.11**	62.59	**0.16**	0.05	0.34	16.92	20.00	0.32	**0.04**	**0.33**	23.73	**73.90**	0.28	**0.04**	0.36	24.05	73.88
20-0	0.56	0.45	1.30	63.68	78.54	**0.17**	0.20	0.72	**84.75**	87.50	2.97	0.11	**0.21**	63.18	**88.00**	2.58	**0.10**	0.28	63.18	**88.00**
20-12	1.18	0.40	6.19	**42.77**	76.77	**0.24**	**0.20**	**1.54**	14.24	17.50	3.15	1.53	6.58	11.19	**83.67**	1.91	1.07	3.76	11.19	**83.67**
20-122	0.87	0.72	5.75	34.90	78.76	**0.17**	**0.02**	**0.07**	**84.94**	**87.50**	2.34	0.38	0.38	44.70	66.02	2.61	0.75	0.67	44.70	66.02

The bold values represent the best results in comparison.



(a) seq03 (b) seq11 (c) seq18 (d) seq20

Fig. 5. Visible examples of object tracking: the object mesh models (top row) and the predicted masks from mesh models (bottom row).



(a) car03-1 mesh model before smoothing (b) car03-1 mesh model after smoothing

Fig. 6. Visible examples about constrained smoothing: (a) the mesh model with incorrect burr points before smoothing. (b) the mesh model with the burr points corrected after constrained smoothing.

TABLE III
COMPUTATIONAL TIME OF ADDED MODULES AVERAGED OVER FRAMES WITH OBJECTS TRACKED ON THE SELECTED SEQUENCES

Added Modules	Time Cost (ms)
Dynamic Feature Elimination	1.530
Object Association	2.241 (avg/object)
Mesh Generation	1.344 (avg/object)
Constrained Smoothing	54.820 (avg/object)
Object Pose Estimation	17.786 (avg/object)
Object Bundle Adjustment	84.230 (avg/object)

* (avg/object) denotes a further average over the number of objects.

estimation. Compared to TwistSLAM, our method generally has a smaller $\mathrm{RPE_t}$ and a smaller ATE on some distant objects (e.g. 05-31, 10-0). TwistSLAM outperforms other methods in ATE on other sequences because it uses the road plane to constrain the degrees of object motion freedom. However, it is challenging to fit the ground plane where distant vehicles are located.

D. Computational Analysis

We perform experiments on the dataset with given timestamps, so the authentic time information for each frame can be considered as known. However, the added modules cause extra computation that affects the real-time performance of the system. Here we provide the timing analysis of the runtime system. The extra computation is highly dependent on the number of objects and dynamic landmarks. For example, our system can achieve 4 fps on seq03 with only 2 objects and less than 1 fps on seq20 with over 10 objects. Compared to the original ORB-SLAM3 that can run in real-time, there is a degradation in the time performance of our system. To explore this, we count the average cost time of added modules in Table III. Note that

we do not include the time of instance segmentation, and in our experiments, the semantic masks are obtained offline and used as input. The instance segmentation [12] we use can run in real-time with a GPU. Table III shows that the main extra cost is in constrained smoothing and object bundle adjustment. For constrained smoothing, the dynamic landmarks are optimized separately, which increases the computational effort. For object bundle adjustment, the deformation graph establishes mutual constraints between pairs of dynamic landmarks, rendering it unfeasible to employ Schur's complement for marginalizing landmark variables to accelerate the optimization.

V. Conclusions and Future Work

In this letter, we propose DynaMeshSLAM, a mesh-based visual SLAMMOT method that leverages the intrinsic geometry of objects. The mesh projection facilitates object tracking while the constrained smoothing regularizes dynamic landmarks. The deformation graph integrated in object bundle adjustment maintains the local rigidity of the object structure during optimization. The experiments show that our method achieves accurate object tracking and pose estimation. In the future, we think it is worth exploring to fuse intrinsic and external constraints in a tight-coupled SLAMMOT framework.

References

[1] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular slam," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.

[2] 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 Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[3] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual slam and structure from motion in dynamic environments: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 1–36, 2018.

[4] C. Yu et al., "DS-SLAM: A semantic visual slam towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1168–1174.

[5] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018.

[6] J. Zhang, M. Henein, R. Mahony, and V. Ila, "VDO-SLAM: A visual dynamic object-aware SLAM system," 2021, *arXiv:2005.11052*.

[7] M. Gonzalez, E. Marchand, A. Kacete, and J. Royan, "TwistSLAM: Constrained SLAM in dynamic environment," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 6846–6853, Jul. 2022.

[8] P. Li et al., "Stereo vision-based semantic 3D object and ego-motion tracking for autonomous driving," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 646–661.

[9] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu, "ClusterVO: Clustering moving instances and estimating visual odometry for self and surroundings," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2168–2177.

[10] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 5191–5198, Jul. 2021.

[11] Y. Qiu, C. Wang, W. Wang, M. Henein, and S. Scherer, "AirDOS: Dynamic slam benefits from articulated objects," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 8047–8053.

[12] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9157–9166.

[13] J. Zhan, Y. Luo, C. Guo, Y. Wu, J. Meng, and J. Liu, "Yolopx: Anchor-free multi-task learning network for panoptic driving perception," *Pattern Recognit.*, vol. 148, 2023, Art. no. 110152.

[14] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4471–4478.

[15] M. Runz, M. Buffier, and L. Agapito, "MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2018, pp. 10–20.

[16] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "Mid-Fusion: Octree-based object-level multi-instance dynamic SLAM," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 5231–5237.

[17] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic SLAM: The need for speed," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2123–2129.

[18] I. Ballester, A. Fontán, J. Civera, K. H. Strobl, and R. Triebel, "DoT: Dynamic object tracking for visual SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11705–11711.

[19] A. Rosinol, T. Sattler, M. Pollefeys, and L. Carlone, "Incremental visual-inertial 3D mesh generation with structural regularities," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 8220–8226.

[20] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1689–1696.

[21] A. Rosinol and L. Carlone, "Smooth mesh estimation from depth data using non-smooth convex optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 6633–6640.

[22] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2022–2038, Aug. 2022.

[23] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, 2016.

[24] Q. Feng, Y. Meng, M. Shan, and N. Atanasov, "Localization and mapping using instance-specific mesh models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4985–4991.

[25] J. Wang, M. Rünz, and L. Agapito, "DSP-SLAM: Object oriented slam with deep shape priors," in *Proc. IEEE Int. Conf. 3D Vis.*, 2021, pp. 1362–1371.

[26] M. Gonzalez, E. Marchand, A. Kacete, and J. Royan, "TwistSLAM: Fusing multiple modalities for accurate dynamic semantic SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 9126–9132.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to sift or surf," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[28] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proc. Asian Conf. Comput. Vis.*, 2010, pp. 25–38.

[29] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.

[30] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.

[31] W. N. Greene and N. Roy, "FLaME: Fast lightweight mesh estimation using variational smoothing on delaunay graphs," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4696–4704.

[32] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Process.*, 2004, pp. 175–184.

[33] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," in *Proc. ACM Siggraph 2007 Papers*, 2007, p. 80–es. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1275808.1276478#

[34] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3607–3613.

[35] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.