VDO-SLAM: A Visual Dynamic Object-aware SLAM System

Jun Zhang[co], Mina Henein[co], Robert Mahony and Viorela Ila

***Abstract*—Combining Simultaneous Localisation and Mapping (SLAM) estimation and dynamic scene modelling can highly benefit robot autonomy in dynamic environments. Robot path planning and obstacle avoidance tasks rely on accurate estimations of the motion of dynamic objects in the scene. This paper presents VDO-SLAM, a robust visual dynamic object-aware SLAM system that exploits semantic information to enable accurate motion estimation and tracking of dynamic rigid objects in the scene without any prior knowledge of the objects' shape or geometric models. The proposed approach identifies and tracks the dynamic objects and the static structure in the environment and integrates this information into a unified SLAM framework. This results in highly accurate estimates of the robot's trajectory and the full $\mathrm{SE}(3)$ motion of the objects as well as a spatiotemporal map of the environment. The system is able to extract linear velocity estimates from objects' $\mathrm{SE}(3)$ motion providing an important functionality for navigation in complex dynamic environments. We demonstrate the performance of the proposed system on a number of real indoor and outdoor datasets and the results show consistent and substantial improvements over the state-of-the-art algorithms. An open-source version of the source code is available*.**

***Index Terms*—SLAM, dynamic scene, object motion estimation, multiple object tracking.**

I. Introduction

The ability of a robot to build a model of the environment, often called map, and to localise itself within this map is a key factor in enabling autonomous robots to operate in real world environments. Creating these maps is achieved by fusing multiple sensor measurements into a consistent representation using estimation techniques such as Simultaneous Localisation And Mapping (SLAM). SLAM is a mature research topic and have already revolutionised a wide range of applications from mobile robotics, inspection, entertainment and film production to exploration and monitoring of natural environments, amongst many others. However, most of the existing solutions to SLAM rely heavily on the assumption that the environment is predominantly static.

The conventional techniques to deal with dynamics in SLAM is to either treat any sensor data associated with moving objects as outliers and remove them from the estimation process ([1]–[5]), or detect moving objects and track them separately using traditional multi-target tracking approaches ([6]–[9]). The former technique excludes information about dynamic objects in the scene, and generates static only maps. The accuracy of the latter depends on the camera pose estimation, which is more susceptible to failure in complex dynamic environments. Increased presence of autonomous systems in dynamic environments is driving the community to challenge the static world assumption that underpins most existing open-source SLAM algorithms. In this paper, we redefine the term "mapping" in SLAM to be concerned with a *spatiotemporal representation of the world*, as opposed to the concept of a static map that has long been the emphasis of the classical SLAM algorithms. Our approach focuses on accurately estimate the motion of all dynamic entities in the environment including the robot and other moving objects in the scene, this information being highly relevant in the context of robot path planning and navigation in dynamic environments.

Existing scene motion estimation techniques mainly rely on optical flow estimation ([10]–[13]) and scene flow estimation ([14]–[17]). Optical flow records the scene motion by estimating the velocities associated with the movement

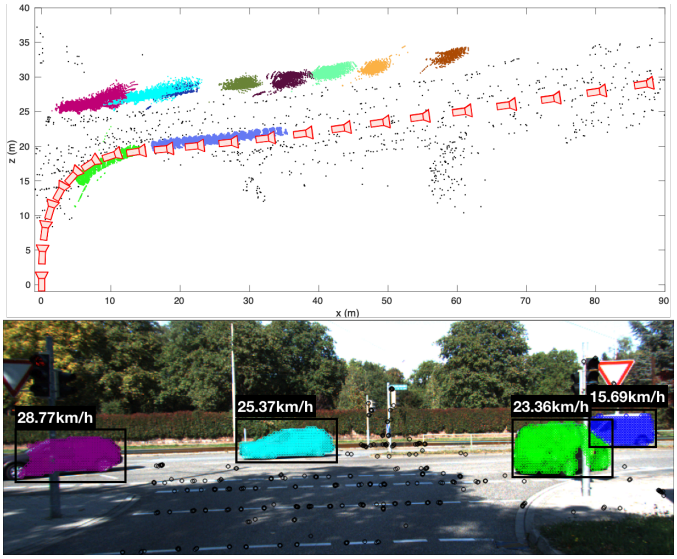

Fig. 1: **Results of our VDO-SLAM system.** (Top) A full map including camera trajectory in red, static background points in black and points on moving objects colour coded by their instance. (Bottom) Detected 3D points on the static background and the objects' body, and the estimated object speed. Black circles represents static points, and each object is shown with a different colour.

Jun Zhang, Mina Henein and Robert Mahony are with the Australian National University (ANU), 0020 Canberra, Australia. {jun.zhang2,mina.henein,robert.mahony}@anu.edu.au

Viorela Ila is with the University of Sydney (USyd), 2006 Sydney, Australia. viorela.ila@sydney.edu.au

[co]: The two authors contributed equally to this work.

*https://github.com/halajun/vdo_slam

of brightness patterns on an image plane. Scene flow, on the other hand, describes the 3D motion field of a scene observed at different instants of time. Those techniques only estimate linear translation of individual pixels or 3D points in the scene, and are not exploiting the collective behaviour points on rigid objects failing to describe the full $\mathrm{SE}(3)$ motion of objects in the scene. In this paper we explore this collective behaviour of points on individual objects to obtain accurate and robust motion estimation of the objects in the scene while simultaneously localising the robot and map the environment.

A typical SLAM system consists of a front-end module, that processes the raw data from the sensors and a back-end module, that integrates the obtained information (raw and higher-level information) into a probabilistic estimation framework. Simple primitives such as 3D locations of salient features are commonly used to represent the environment. This is largely a consequence of the fact that points are easy to detect, track and integrate within the SLAM estimation problem.

Feature tracking has been more reliable and robust with the advances in deep learning to provide algorithms that can reliably estimate the $2D$ optical flow associated with the apparent motion of every pixel on an image in a dense manner. A task that is particularly important for data association and that has been otherwise challenging in dynamic environments using classical feature tracking methods.
Other primitives such as lines and planes ([18]–[21]) or even objects ([22]–[24]) have been considered in order to provide richer map representations. To incorporate such information in existing geometric SLAM algorithms, either a dataset of $3D$-models of every object in the scene must be available a priori ([23], [25]) or the front end must explicitly provide object pose information in addition to detection and segmentation ([26]–[28]) adding a layer of complexity to the problem. The requirement for accurate $3D$-models severely limits the potential domains of application, while to the best of our knowledge, multiple object tracking and $3D$ pose estimation remain a challenge to learning techniques. There is a clear need for an algorithm that can exploit the powerful detection and segmentation capabilities of modern deep learning algorithms ([29], [30]) without relying on additional pose estimation or object model priors, an algorithm that operates at feature-level with the awareness of an object concept.

While the problems of SLAM and object motion tracking/estimation are long studied in isolation in the literature, recent approaches try to solve the two problems in a unified framework ([31], [32]). However, they both focus on the SLAM back-end instead of a full system, resulting in a severely limited performance in real world scenarios. In this paper, we carefully integrate our previous works ([31], [33]) and propose VDO-SLAM, a novel feature-based stereo/RGB-D dynamic SLAM system, that leverages image-based semantic information to simultaneously localise the robot, map the static and dynamic structure, and track motions of rigid objects in the scene. Different to [31], we rely on a denser object feature representation to ensure robust tracking, and propose new factors to smoothen the motion of rigid objects in urban driving scenarios. Different to [33], an improved robust feature and object tracking method is proposed, with the ability to handle indirect occlusions resulting from the failure of semantic object segmentation. In summary, the contributions of this work are:

- a novel formulation to model dynamic scenes in a unified estimation framework over robot poses, static and dynamic 3D points, and object motions.
- accurate estimation for $\mathrm{SE}(3)$ motion of dynamic objects that outperforms state-of-the-art algorithms, as well as a way to extract objects' velocity in the scene,
- a robust method for tracking moving objects exploiting semantic information with the ability to handle indirect occlusions resulting from the failure of semantic object segmentation,
- a demonstrable *full system* in complex and compelling real-world scenarios.

To the best of our knowledge, this is the first full dynamic SLAM system that is able to achieve motion segmentation, dynamic object tracking, and estimate the camera poses along with the static and dynamic structure, the full $\mathrm{SE}(3)$ pose change of every rigid object in the scene, extract velocity information, and be demonstrable in real-world outdoor scenarios (see Fig. 1). We demonstrate the performance of our algorithm on real datasets and show capability of the proposed system to resolve rigid object motion estimation and yield motion results that are comparable to the camera pose estimation in accuracy and that outperform state-of-the-art algorithms by an order of magnitude in urban driving scenarios.

The remainder of this paper is structured as follows, in the following Section II we discuss the related work. In Section III and IV we describe the proposed algorithm and system. We introduce the experimental setup, followed by the results and evaluations in Section V. We summarise and offer concluding remarks in Section VI.

II. Related Work

In the past two decades, the study of SLAM for dynamic environments has become more and more popular in the community, with a considerable amount of algorithms being proposed to solve the dynamic SLAM problem. Motivated by different goals to achieve, solutions in the literature can be mainly divided into three categories.

The first category aims to explore robust SLAM against dynamic environments. Early methods in this category ([2], [34], [35]) normally detect and remove the information drawn from dynamic foreground, which is seen as degrading the SLAM performance. More recent methods on this track tend to go further by not just removing the dynamic foreground, but also inpainting or reconstructing the static background that is occluded by moving targets. [5] present dynaSLAM that combines classic geometry and deep learning-based models to detect and remove dynamic objects, then inpaint the occluded background with multi-view information of the scene. Similarly, a Light Field SLAM front-end is proposed by [36] to reconstruct the occluded static scene via Synthetic Aperture Imaging (SAI) technics. Different from [5], features on the reconstructed static background are also tracked and used

to achieve better SLAM performance. The above state-of-the-art solutions achieve robust and accurate estimation by discarding the dynamic information. However, we argue that this information has potential benefits for SLAM if it is properly modelled. Furthermore, understanding dynamic scenes in addition to SLAM is crucial for many other robotics tasks such as planning, control and obstacle avoidance, to name a few.

Approaches of the second category performs SLAM and Moving Objects Tracking (MOT) separately, as an extension to conventional SLAM for dynamic scene understanding ([9], [37]–[39]). [37] developed a theory for performing SLAM with Moving Objects Tracking (SLAMMOT). In the latest version of their SLAM with detection and tracking of moving objects, the estimation problem is decomposed into two separate estimators (moving and stationary objects) to make it feasible to update both filters in real time. [9] tackle the SLAM problem with dynamic objects by solving the problems of Structure from Motion (SfM) and tracking of moving objects in parallel, and unifying the output of the system into a 3D dynamic map containing the static structure and the trajectories of moving objects. Later in [38], the authors propose to integrate semantic constraints to further improve the 3D reconstruction. The more recent work [39] present a stereo-based dense mapping algorithm in a SLAM framework, with the advantage of accurately and efficiently reconstructing both static background and moving objects in large scale dynamic environments. The listed algorithms above have proven that combining multiple objects tracking with SLAM is doable and applicable for dynamic scene exploration. To take a step further by proper exploiting and establishing the spatial and temporal relationships between the robot, static background, stationary and dynamic objects, we show in this paper that the problems of SLAM and multi-object tracking are mutually beneficial.

The last and most active category is object SLAM, which usually includes both static and dynamic objects. Algorithms in this class normally require specific modelling and representation of 3D object, such as 3D shape ([40]–[42]), surfel [43] or volumetric [44] model, geometric model such as ellipsoid ([45], [46]) or 3D bounding box ([24], [47]–[49]), etc., to extract high-level primitive (e.g., object pose) and integrate into a SLAM framework. [40] is one of the earliest works to introduce an object-oriented SLAM paradigm, which represents cluttered scene in object level and constructs an explicit graph between camera and object poses to achieve joint pose-graph optimisation. Later, [41] propose a novel 3D object recognition algorithm to ensure the system robustness and improve the accuracy of estimated object pose. The high-level scene representation enables real-time 3D recognition and significant compression of map storage for SLAM. Nevertheless, a database of pre-scanned or pre-trained object models has to be created in advance. To avoid prebuilt database, representing objects using surfel or voxel element in a dense manner starts to gain popularity, along with RGB-D cameras becoming widely used. [43] present MaskFusion that adopts surfel representation to model, track and reconstruct objects in the scene, while [44] apply an octree-based volumetric model to objects and build multi-object dynamic SLAM system. Both methods succeed to exploit object information in a dense RGB-D SLAM framework, without prior knowledge of object model. Their main interest, however, is the 3D object segmentation and consistent fusion of the dense map rather than the estimation of the motion of the objects.

Lately, the use of basic geometric models to represent objects becomes a popular solution due to the less complexity and easy integration into a SLAM framework. In Quadric-SLAM [46], detected objects are represented as ellipsoids to compactly parametrise the size and 3D pose of an object. In this way, the quadric parameters are directly constrained as geometric error and formulated together with camera poses in a factor graph SLAM for joint estimation. [24] propose to combine 2D and 3D object detection with SLAM for both static and dynamic environments. Objects are represented as high-quality cuboids and optimized together with points and cameras through multi-view bundle adjustment. While both methods prove the mutual benefit between detected object and SLAM, their main focus is on object detection and SLAM primarily for static scenarios. In this paper, we take this direction further to tackle the challenging problem of dynamic object tracking within a SLAM framework, and exploit the relationships between moving objects and agent robot, static and dynamic structures for potential advantages.

Apart from the dynamic SLAM categories, the literature of 6-DoF object motion estimation is also crucial for dynamic SLAM problem. Quite a few methods have been proposed in the literature to estimate $\mathrm{SE}(3)$ motion of objects in a visual odometry or SLAM framework ([50]–[52]). [50] present a model-free method for detecting and tracking moving objects in 3D LiDAR scans. The method sequentially estimates motion models using RANSAC [53], then segments and tracks multiple objects based on the models by a proposed Bayesian approach. In [51], the authors address the problem of simultaneous estimation of ego and third-party $\mathrm{SE}(3)$ motions in complex dynamic scenes using cameras. They apply multi-model fitting techniques into a visual odometry pipeline and estimate all rigid motions within a scene. In later work, [52] present ClusterVO that is able to perform online processing for multiple motion estimations. To achieve this, a multi-level probabilistic association mechanism is proposed to efficiently track features and detections, then a heterogeneous Conditional Random Field (CRF) clustering approach is applied to jointly infer cluster segmentations, with a sliding-window optimization for clusters in the end. While the above proposed methods represent an important step forward to the Multi-motion Visual Odometry (MVO) task, the study of spacial and temporal relationships is not fully explored but is arguably important. Therefore, by carefully considering the pros and cons in the literature of SLAM+MOT, object SLAM and MVO, this paper proposes a visual dynamic object-aware SLAM system that is able to achieve robust ego and object motion tracking, as well as consistent static and dynamic mapping in a novel SLAM formulation.

III. Methodology

Before discussing details of the proposed system pipeline, as shown in Fig. 4, this section covers the mathematical details

of the core components in the system. Variables and notations are first introduced, including the novel way of modelling the motion of a rigid-object in a model free manner. Then we show how the camera pose and object motion are estimated in the tracking component of the system. Finally, a factor graph optimisation is proposed and applied in the mapping component, to refine the camera poses and object motions, and build a global consistent map including static and dynamic structure.

A. Background and Notation

1) Coordinate Frames: Let ${}^{0}\mathbf{X}_k, {}^{0}\mathbf{L}_k \in \mathrm{SE}(3)$ be the robot/camera and the object 3D pose respectively, at time k in a global reference frame 0, with $k \in \mathscr{T}$ the set of time steps. Note that calligraphic capital letters are used in our notation to represent sets of indices. Fig. 2 shows these pose transformations as solid curves.

2) Points: Let ${}^{0}\mathbf{m}_k^i$ be the homogeneous coordinates of the i^{th} 3D point at time k, with ${}^{0}\mathbf{m}^i = \left[m_x^i, m_y^i, m_z^i, 1\right]^\top \in \mathbb{E}^3$ and $i \in \mathscr{M}$ the set of points. We write a point in robot/camera frame as ${}^{X_k}\mathbf{m}_k^i = {}^{0}\mathbf{X}_k^{-1}\,{}^{0}\mathbf{m}_k^i$.

Define $\mathbf{I}_k$ the reference frame associated with the image captured by the camera at time k chosen at the top left corner of the image, and let ${}^{I_k}\mathbf{p}_k^i = \left[u^i, v^i, 1\right] \in \mathbb{E}^2$ be the pixel location on frame $\mathbf{I}_k$ corresponding to the homogeneous 3D point ${}^{\mathbf{X}_k}\mathbf{m}_k^i$, which is obtained via the projection function $\pi(\cdot)$ as follows:

$$ {}^{I_k}\mathbf{p}_k^i = \pi({}^{X_k}\mathbf{m}_k^i) = \mathbf{K}\,{}^{X_k}\mathbf{m}_k^i\,, \tag{1} $$

where $\mathbf{K}$ is the camera intrinsics matrix.

The camera and/or object motions both produce an optical flow ${}^{I_k}\boldsymbol{\phi}^i \in \mathbb{R}^2$ that is the displacement vector indicating the motion of pixel ${}^{I_{k-1}}\mathbf{p}_{k-1}^i$ from image frame I_{k-1} to I_k, and is given by:

$$ {}^{I_k}\boldsymbol{\phi}^i = {}^{I_k}\tilde{\mathbf{p}}_k^i - {}^{I_{k-1}}\mathbf{p}_{k-1}^i\,. \tag{2} $$

Here ${}^{I_k}\tilde{\mathbf{p}}_k^i$ is the correspondence of ${}^{I_{k-1}}\mathbf{p}_{k-1}^i$ in I_k. Note that, we overload the same notation to represent the 2D pixel coordinates $\in \mathbb{R}^2$. In this work, we leverage optical flow to find correspondences between consecutive frames.

3) Object and 3D Point Motions: The object motion between times $k-1$ and k is described by the homogeneous transformation ${}^{L_{k-1}}_{k-1}\mathbf{H}_k \in \mathrm{SE}(3)$ according to:

$$ {}^{L_{k-1}}_{k-1}\mathbf{H}_k = {}^{0}\mathbf{L}_{k-1}^{-1}\,{}^{0}\mathbf{L}_k\,. \tag{3} $$

Fig. 2 shows these motion transformations as dashed curves. We write a point in its corresponding object frame as ${}^{L_k}\mathbf{m}_k^i = {}^{0}\mathbf{L}_k^{-1}\,{}^{0}\mathbf{m}_k^i$ (shown as a dashed vector from the object reference frame to the red dot in Fig. 2), substituting the object pose at time k from (3), this becomes:

$$ {}^{0}\mathbf{m}_k^i = {}^{0}\mathbf{L}_k\,{}^{L_k}\mathbf{m}_k^i = {}^{0}\mathbf{L}_{k-1}\,{}^{L_{k-1}}_{k-1}\mathbf{H}_k\,{}^{L_k}\mathbf{m}_k^i\,. \tag{4} $$

Note that for rigid body objects, ${}^{L_k}\mathbf{m}_k^i$ stays constant at ${}^{L}\mathbf{m}^i$, and ${}^{L}\mathbf{m}^i = {}^{0}\mathbf{L}_k^{-1}\,{}^{0}\mathbf{m}_k^i = {}^{0}\mathbf{L}_{k+n}^{-1}\,{}^{0}\mathbf{m}_{k+n}^i$ for any integer $n \in \mathbb{Z}$. Then, for rigid objects with $n=-1$, (4) becomes:

$$ {}^{0}\mathbf{m}_k^i = {}^{0}\mathbf{L}_{k-1}\,{}^{L_{k-1}}_{k-1}\mathbf{H}_k\,{}^{0}\mathbf{L}_{k-1}^{-1}\,{}^{0}\mathbf{m}_{k-1}^i\,. \tag{5} $$

(5) is crucially important as it relates the same 3D point on a rigid object in motion at consecutive time steps by a homogeneous transformation ${}^{0}_{k-1}\mathbf{H}_k := {}^{0}\mathbf{L}_{k-1}\,{}^{L_{k-1}}_{k-1}\mathbf{H}_k\,{}^{0}\mathbf{L}_{k-1}^{-1}$. This equation represents a *frame change of a pose transformation* [54], and shows how the body-fixed frame pose change ${}^{L_{k-1}}_{k-1}\mathbf{H}_k$ relates to the global reference frame pose change ${}^{0}_{k-1}\mathbf{H}_k$. The point motion in global reference frame is then expressed as:

$$ {}^{0}\mathbf{m}_k^i = {}^{0}_{k-1}\mathbf{H}_k\,{}^{0}\mathbf{m}_{k-1}^i\,. \tag{6} $$

Equation (6) is at the core of our motion estimation approach, as it expresses the rigid object pose change in terms of the points that reside on the object in a model-free manner without the need to include the object 3D pose as a random variable in the estimation. Section III-B2 details how this rigid object pose change is estimated based on the above equation. Here ${}^{0}_{k-1}\mathbf{H}_k \in \mathrm{SE}(3)$ represents the object point motion in global reference frame; for the remainder of this document, we refer to this quantity as the object pose change or the object motion for ease of reading.

B. Camera Pose and Object Motion Estimation

The cost function chosen to estimate the camera pose and object motion is associated with the 3D-2D re-projection error and is defined on the image plane. Since the noise is better characterised in image plane, this yields more accurate results for camera localisation [55]. Moreover, based on this error term, we propose a novel formulation to jointly optimise the optical flow along with the camera pose and the object motion, to ensure a robust tracking of points. In the mapping module, a 3D error cost function is used in global optimization to ensure best results of 3D structure and object motions estimation as later described in Section III-C.

1) Camera Pose Estimation: Given a set of static 3D points $\{{}^{0}\mathbf{m}_{k-1}^i \mid i \in \mathscr{M}, k \in \mathscr{T}\}$ observed at time $k-1$ in global reference frame, and the set of $2D$ correspondences $\{{}^{I_k}\tilde{\mathbf{p}}_k^i \mid i \in \mathscr{M}, k \in \mathscr{T}\}$ in image $\mathbf{I}_k$, the camera pose ${}^{0}\mathbf{X}_k$ is estimated via minimizing the re-projection error:

$$ \mathbf{e}_i({}^{0}\mathbf{X}_k) = {}^{I_k}\tilde{\mathbf{p}}_k^i - \pi({}^{0}\mathbf{X}_k^{-1}\,{}^{0}\mathbf{m}_{k-1}^i)\,. \tag{7} $$

We parameterise the $\mathrm{SE}(3)$ camera pose by elements of the Lie-algebra $\mathbf{x}_k \in \mathrm{se}(3)$:

$$ {}^{0}\mathbf{X}_k = \exp({}^{0}\mathbf{x}_k)\,, \tag{8} $$

and define ${}^{0}\mathbf{x}_k^\vee \in \mathbb{R}^6$ with the *vee* operator a mapping from $\mathrm{se}(3)$ to $\mathbb{R}^6$. Using the Lie-algebra parameterisation of $\mathrm{SE}(3)$ with the substitution of (8) into (7), the solution of the least squares cost is given by:

$$ {}^{0}\mathbf{x}_k^{*\vee} = \underset{{}^{0}\mathbf{x}_k^\vee}{\operatorname{argmin}} \sum_{i}^{n_b} \rho_h\left(\mathbf{e}_i^\top({}^{0}\mathbf{x}_k)\,\Sigma_p^{-1}\,\mathbf{e}_i({}^{0}\mathbf{x}_k)\right) \tag{9} $$

for all n_b visible 3D-2D static background point correspondences between consecutive frames. Here ρ_h is the Huber function [56], and Σ_p is the covariance matrix associated with the re-projection error. The estimated camera pose is given by ${}^{0}\mathbf{X}_k^* = \exp({}^{0}\mathbf{x}_k^*)$ and is found using the Levenberg-Marquardt algorithm to solve for (9).

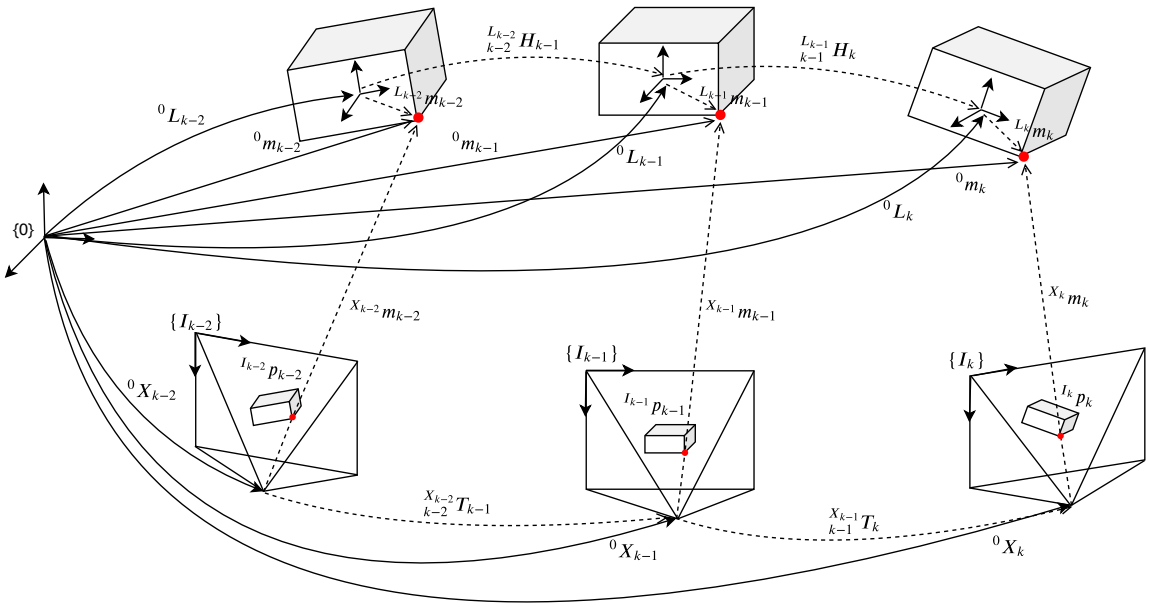

Fig. 2: **Notation and coordinate frames.** Solid curves represent camera and object poses in inertial frame; ${}^0\mathbf{X}$ and ${}^0\mathbf{L}$ respectively, and dashed curves their respective motions in body-fixed frame. Solid lines represent 3D points in inertial frame, and dashed lines represent 3D points in camera frames.

2) Object Motion Estimation: Analogous to the camera pose estimation, a cost function based on re-projection error is constructed to solve for the object motion ${}^0_{k-1}\mathbf{H}_k$. Using (6), the error term between the re-projection of an object 3D point and the corresponding $2D$ point in image $\mathbf{I}_k$ is:

$$\begin{aligned}\mathbf{e}_i({}^0_{k-1}\mathbf{H}_k) :=\;& {}^{I_k}\tilde{\mathbf{p}}^i_k - \pi({}^0\mathbf{X}_k^{-1}\,{}^0_{k-1}\mathbf{H}_k\,{}^0\mathbf{m}^i_{k-1})\\ =\;& {}^{I_k}\tilde{\mathbf{p}}^i_k - \pi({}^0_{k-1}\mathbf{G}_k\,{}^0\mathbf{m}^i_{k-1})\,,\end{aligned} \tag{10}$$

where ${}^0_{k-1}\mathbf{G}_k \in \mathrm{SE}(3)$. Parameterising ${}^0_{k-1}\mathbf{G}_k := \exp\left({}^0_{k-1}\mathbf{g}_k\right)$ with ${}^0_{k-1}\mathbf{g}_k \in \mathrm{se}(3)$, the optimal solution is found via minimising:

$${}^0_{k-1}\mathbf{g}^{*\vee}_k = \underset{{}^0_{k-1}\mathbf{g}^{\vee}_k}{\operatorname{argmin}} \sum_i^{n_d} \rho_h\left(\mathbf{e}_i^\top({}^0_{k-1}\mathbf{g}_k)\,\Sigma_p^{-1}\,\mathbf{e}_i({}^0_{k-1}\mathbf{g}_k)\right) \tag{11}$$

given all n_d visible 3D-2D dynamic point correspondences on an object between frames $k-1$ and k. The object motion, ${}^0_{k-1}\mathbf{H}_k = {}^0\mathbf{X}_k\,{}^0_{k-1}\mathbf{G}_k$ can be recovered afterwards.

3) Joint Estimation with Optical Flow: The camera pose and object motion estimation both rely on good image correspondences. Tracking of points on moving objects can be very challenging due to occlusions, large relative motions and large camera-object distances. In order to ensure a robust tracking of points, we follow our earlier work [33] to refine the estimation of the optical flow jointly with the motion estimation.

For camera pose estimation, the error term in (7) is reformulated considering (2) as:

$$\mathbf{e}_i({}^0\mathbf{X}_k, {}^{I_k}\boldsymbol{\phi}) = {}^{I_{k-1}}\mathbf{p}^i_{k-1} + {}^{I_k}\boldsymbol{\phi}^i - \pi({}^0\mathbf{X}_k^{-1}\,{}^0\mathbf{m}^i_{k-1})\,. \tag{12}$$

Applying the Lie-algebra parameterisation of SE(3) element, the optimal solution is obtained via minimising the cost function:

$$\begin{aligned}\{{}^0\mathbf{x}^{*\vee}_k, {}^k\boldsymbol{\Phi}^*_k\} = \underset{\{{}^0\mathbf{x}^{\vee}_k, {}^k\boldsymbol{\Phi}_k\}}{\operatorname{argmin}} \sum_i^{n_b} \Big\{ &\rho_h\left(\mathbf{e}_i^\top({}^{I_k}\boldsymbol{\phi}^i)\,\Sigma_\phi^{-1}\,\mathbf{e}_i({}^{I_k}\boldsymbol{\phi}^i)\right) +\\ &\rho_h\left(\mathbf{e}_i^\top({}^0\mathbf{x}_k, {}^{I_k}\boldsymbol{\phi}^i)\,\Sigma_p^{-1}\,\mathbf{e}_i({}^0\mathbf{x}_k, {}^{I_k}\boldsymbol{\phi}^i)\right)\Big\}\,,\end{aligned} \tag{13}$$

where $\rho_h(\mathbf{e}_i^\top({}^{I_k}\boldsymbol{\phi}^i)\,\Sigma_\phi^{-1}\,\mathbf{e}_i({}^{I_k}\boldsymbol{\phi}^i))$ is the regularization term with

$$\mathbf{e}_i({}^{I_k}\boldsymbol{\phi}^i) = {}^{I_k}\hat{\boldsymbol{\phi}}^i - {}^{I_k}\boldsymbol{\phi}^i\,. \tag{14}$$

Here ${}^{I_k}\hat{\boldsymbol{\Phi}}^i = \{{}^{I_k}\hat{\boldsymbol{\phi}}^i \mid i \in \mathscr{M}, k \in \mathscr{T}\}$ is the initial optic-flow obtained through classical or learning-based methods, and Σ_ϕ is the associated covariance matrix. Analogously, the cost function for object motion in (11) combining optical flow refinement is given by

$$\begin{aligned}\{{}^0_{k-1}\mathbf{g}^{*\vee}_k, {}^k\boldsymbol{\Phi}^*_k\} = \underset{\{{}^0_{k-1}\mathbf{g}^{\vee}_k, {}^k\boldsymbol{\Phi}_k\}}{\operatorname{argmin}} \sum_i^{n_d} \Big\{ &\rho_h\left(\mathbf{e}_i^\top({}^{I_k}\boldsymbol{\phi}^i)\,\Sigma_\phi^{-1}\,\mathbf{e}_i({}^{I_k}\boldsymbol{\phi}^i)\right) +\\ &\rho_h\left(\mathbf{e}_i^\top({}^0_{k-1}\mathbf{g}_k, {}^{I_k}\boldsymbol{\phi}^i)\,\Sigma_p^{-1}\,\mathbf{e}_i({}^0_{k-1}\mathbf{g}_k, {}^{I_k}\boldsymbol{\phi}^i)\right)\Big\}\,.\end{aligned} \tag{15}$$

C. Graph Optimisation

The proposed approach formulates the dynamic SLAM as a graph optimisation problem, to refine the camera poses and object motions, and build a global consistent map including static and dynamic structure. We model the dynamic SLAM problem as a factor graph as the one demonstrated in Fig. 3. The factor graph formulation is highly intuitive and has the advantage that it allows for efficient implementations of batch ([57], [58]) and incremental ([59]–[61]) solvers.

Four types of measurements/observations are integrated into a joint optimisation problem; the 3D point measurements, the

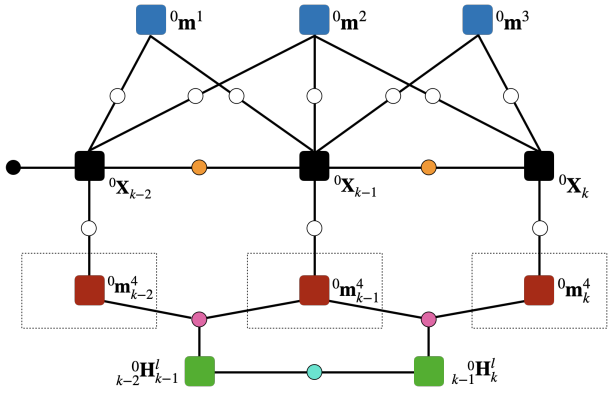

Fig. 3: **Factor graph representation of an object-aware SLAM with a moving object.** Black squares stand for the camera poses at different time steps, blue for static points, red for the same dynamic point on an object (dashed box) at different time steps and green for the object pose change between time steps. For ease of visualisation, only one dynamic point is drawn here. A prior factor is shown as a black circle, odometry factors are shown as orange, point measurement factors as white and point motion factors as magenta. A smooth motion factor is shown as cyan circle.

visual odometry measurements, the motion of points on a dynamic object and the object smooth motion observations.

The 3D point measurement model error $\mathbf{e}_{i,k}({}^0\mathbf{X}_k, {}^0\mathbf{m}^i_k)$ is defined as:

$$\mathbf{e}_{i,k}({}^0\mathbf{X}_k, {}^0\mathbf{m}^i_k) = {}^0\mathbf{X}_k^{-1}\,{}^0\mathbf{m}^i_k - \mathbf{z}^i_k\,. \tag{16}$$

Here $\mathbf{z} = \{\mathbf{z}^i_k \mid i \in \mathscr{M}, k \in \mathscr{T}\}$ is the set of all 3D point measurements at all time steps, with cardinality n_z and $\mathbf{z}^i_k \in \mathbb{R}^3$. The 3D point measurement factors are shown as white circles in Fig. 3.

The tracking component of the system provides a high-quality ego-motion via 3D-2D error minimization, which can be used as an odometry measurement to constrain camera poses in the graph. The visual odometry model error $\mathbf{e}_k({}^0\mathbf{X}_{k-1}, {}^0\mathbf{X}_k)$ is defined as:

$$\mathbf{e}_k({}^0\mathbf{X}_{k-1}, {}^0\mathbf{X}_k) = ({}^0\mathbf{X}_{k-1}^{-1}\,{}^0\mathbf{X}_k)^{-1}\,{}^{X_{k-1}}_{k-1}\mathbf{T}_k\,, \tag{17}$$

where $\mathbf{T} = \{{}^{X_{k-1}}_{k-1}\mathbf{T}_k \mid k \in \mathscr{T}\}$ is the odometry measurement set with ${}^{X_{k-1}}_{k-1}\mathbf{T}_k \in \mathrm{SE}(3)$ and cardinality n_o. The odometric factors are shown as orange circles in Fig. 3.

The motion model error of points on dynamic objects $\mathbf{e}_{i,l,k}({}^0\mathbf{m}^i_k, {}^0_{k-1}\mathbf{H}^l_k, {}^0\mathbf{m}^i_{k-1})$ is defined as:

$$\mathbf{e}_{i,l,k}({}^0\mathbf{m}^i_k, {}^0_{k-1}\mathbf{H}^l_k, {}^0\mathbf{m}^i_{k-1}) = {}^0\mathbf{m}^i_k - {}^0_{k-1}\mathbf{H}^l_k\,{}^0\mathbf{m}^i_{k-1}\,. \tag{18}$$

The motion of all points on a detected rigid object l are characterised by the same pose transformation ${}^0_{k-1}\mathbf{H}^l_k \in \mathrm{SE}(3)$ given by (6) and the corresponding factor, shown as magenta circles in Fig. 3, is a ternary factor which we call the *motion model of a point on a rigid body*.

It has been shown that incorporating prior knowledge about the motion of objects in the scene is highly valuable in dynamic SLAM ([31], [37]). Motivated by the camera frame rate and the physics laws governing the motion of relatively large objects (vehicles) and preventing their motions to change abruptly, we introduce smooth motion factors to minimise the change in consecutive object motions, with the error term defined as:

$$\mathbf{e}_{l,k}({}^0_{k-2}\mathbf{H}^l_{k-1}, {}^0_{k-1}\mathbf{H}^l_k) = {}^0_{k-2}\mathbf{H}^l_{k-1}{}^{-1}\,{}^0_{k-1}\mathbf{H}^l_k. \tag{19}$$

The object smooth motion factor $\mathbf{e}_{l,k}({}^0_{k-2}\mathbf{H}^l_{k-1}, {}^0_{k-1}\mathbf{H}^l_k)$ is used to minimise the change between the object motion at consecutive time steps and is shown as cyan circles in Fig. 3.

Let $\boldsymbol{\theta}_M = \{{}^0\mathbf{m}^i_k \mid i \in \mathscr{M}, k \in \mathscr{T}\}$ be the set of all 3D points, and $\boldsymbol{\theta}_X = \{{}^0\mathbf{x}^\vee_k \mid k \in \mathscr{T}\}$ as the set of all camera poses. We parameterise the SE(3) object motion ${}^0_{k-1}\mathbf{H}^l_k$ by elements ${}^0_{k-1}\mathbf{h}^l_k \in \mathrm{se}(3)$ the Lie-algebra of SE(3):

$${}^0_{k-1}\mathbf{H}^l_k = \exp({}^0_{k-1}\mathbf{h}^l_k)\,, \tag{20}$$

and define $\boldsymbol{\theta}_H = \{{}^{0\,\vee}_{k-1}\mathbf{h}^l_k \mid k \in \mathscr{T}, l \in \mathscr{L}\}$ as the set of all object motions, with ${}^{0\,\vee}_{k-1}\mathbf{h}^l_k \in \mathbb{R}^6$ and $\mathscr{L}$ the set of all object labels. Given $\boldsymbol{\theta} = \boldsymbol{\theta}_X \cup \boldsymbol{\theta}_M \cup \boldsymbol{\theta}_H$ as all the nodes in the graph, with the Lie-algebra parameterisation of SE(3) for $\mathbf{X}$ and $\mathbf{H}$ (substituting (8) in (16) and (17), and substituting (20) in (18) and (19)), the solution of the least squares cost is given by:

$$\begin{aligned}\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \Big\{ &\sum_{i,k}^{n_z} \rho_h\big(\mathbf{e}^\top_{i,k}({}^0\mathbf{x}_k, {}^0\mathbf{m}^i_k)\,\Sigma_z^{-1}\,\mathbf{e}_{i,k}({}^0\mathbf{x}_k, {}^0\mathbf{m}^i_k)\big) \\ &+ \sum_{k}^{n_o} \rho_h\big(\log(\mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k))^\top\,\Sigma_o^{-1}\,\log(\mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k))\big) \\ &+ \sum_{i,l,k}^{n_g} \rho_h\big(\mathbf{e}^\top_{i,l,k}({}^0\mathbf{m}^i_k, {}^0_{k-1}\mathbf{h}^l_k, {}^0\mathbf{m}^i_{k-1})\,\Sigma_g^{-1} \\ &\qquad \mathbf{e}_{i,l,k}({}^0\mathbf{m}^i_k, {}^0_{k-1}\mathbf{h}^l_k, {}^0\mathbf{m}^i_{k-1})\big) \\ &+ \sum_{l,k}^{n_s} \rho_h\big(\log(\mathbf{e}_{l,k}({}^0_{k-2}\mathbf{h}^l_{k-1}, {}^0_{k-1}\mathbf{h}^l_k))^\top\,\Sigma_s^{-1} \\ &\qquad \log(\mathbf{e}_{l,k}({}^0_{k-2}\mathbf{h}^l_{k-1}, {}^0_{k-1}\mathbf{h}^l_k))\big)\Big\}\,,\end{aligned} \tag{21}$$

where Σ_z is the 3D point measurement noise covariance matrix, Σ_o is the odometry noise covariance matrix, Σ_g is the motion noise covariance matrix with n_g the total number of ternary object motion factors, and Σ_s the smooth motion covariance matrix, with n_s the total number of smooth motion factors. The non-linear least squares problem in (21) is solved using Levenberg-Marquardt method.

IV. System

In this section, we propose a novel object-aware dynamic SLAM system that robustly estimates both camera and object motions, along with the static and dynamic structure of the environment. The full system overview is shown in Fig. 4. The system consists of three main components: image preprocessing, tracking and mapping.

The input to the system is stereo or RGB-D images. For stereo images, as a first step, we extract depth information by applying the stereo depth estimation method described in [62] to generate depth maps and the resulting data is treated as RGB-D.

Although this system was initially designed to be an RGB-D system, as an attempt to fully exploit image-based semantic information, we apply single image depth estimation to achieve depth information from monocular camera. Our "learning-based monocular" system is monocular in the sense that only RGB images are used as input to the system, however the estimation problem is formulated using RGB-D data, where the depth is obtained using single image depth estimation.

A. Pre-processing

There are two challenging aspects that this module needs to fulfil. First, to robustly separate static background and objects, and secondly to ensure long-term tracking of dynamic objects. To achieve this, we leverage recent advances in computer vision techniques for instance level semantic segmentation and dense optical flow estimation in order to ensure efficient object motion segmentation and robust object tracking.

1) Object Instance Segmentation: Instance-level semantic segmentation is used to segment and identify potentially movable objects in the scene. Semantic information constitutes an important prior in the process of separating static and moving object points, e.g., buildings and roads are always static, but cars can be static or dynamic. Instance segmentation helps to further divide semantic foreground into different instance masks, which makes it easier to track each individual object. Moreover, segmentation masks provide a "precise" boundary of the object body that ensures robust tracking of points on the object.

2) Optical Flow Estimation: The dense optical flow is used to maximise the number of tracked points on moving objects. Most of the moving objects only occupy a small portion of the image. Therefore, using sparse feature matching does not guarantee robust nor long-term feature tracking. Our approach makes use of dense optical flow to considerably increase the number of object points by sampling from all the points within the semantic mask. Dense optical flow is also used to consistently track multiple objects by propagating a unique object identifier assigned to every point on an object mask. Moreover, it allows to recover objects masks if semantic segmentation fails; a task that is extremely difficult to achieve using sparse feature matching.

B. Tracking

The tracking component includes two modules; the camera ego-motion tracking with sub-modules of feature detection and camera pose estimation, and the object motion tracking including sub-modules of dynamic object tracking and object motion estimation.

1) Feature Detection: To achieve fast camera pose estimation, we detect a *sparse* set of corner features and track them with optical flow. At each frame, only inlier feature points that fit the estimated camera motion are saved into the map, and used to track correspondences in the next frame. New features are detected and added, if the number of inlier tracks falls below a certain level (1200 in default). These sparse features are detected on static background, i.e., image regions excluding the segmented objects.

2) Camera Pose Estimation: The camera pose is computed using (13) for all detected 3D-2D static point correspondences. To ensure robust estimation, a motion model generation method is applied for initialisation. Specifically, the method generates two models and compares their inlier numbers based on re-projection error. One model is generated by propagating the camera previous motion, while the other by computing a new motion transform using P3P [63] algorithm with RANSAC. The motion model that generates most inliers is then selected for initialisation.

3) Dynamic Object Tracking: The process of object motion tracking consists of two steps. In the first step, segmented objects are classified into static and dynamic. Then we associate the dynamic objects across pairs of consecutive frames.

• Instance-level object segmentation allows us to separate objects from background. Although the algorithm is capable of estimating the motions of all the segmented objects, dynamic object identification helps reduce computational cost of the proposed system. This is done based on scene flow estimation. Specifically, after obtaining the camera pose ${}^0\mathbf{X}_k$, the scene flow vector $\mathbf{f}_k^i$ describing the motion of a 3D point ${}^0\mathbf{m}^i$ between frames $k-1$ and k, can be calculated as in [64]:

$$\mathbf{f}_k^i = {}^0\mathbf{m}_{k-1}^i - {}^0\mathbf{m}_k^i = {}^0\mathbf{m}_{k-1}^i - {}^0\mathbf{X}_k \, {}^{X_k}\mathbf{m}_k^i \,. \tag{22}$$

Unlike optical flow, scene flow—ideally only caused by scene motion—can directly decide whether some structure is moving or not. Ideally, the magnitude of the scene flow vector should be zero for all static 3D points. However, noise or error in depth and matching complicates the situation in real scenarios. To robustly handle this, we compute the scene flow magnitude of all the sampled points on each object. If the magnitude of the scene flow of a certain point is greater than a predefined threshold, the point is considered dynamic. This threshold was set to 0.12 in all experiments carried in this work. An object is then recognised dynamic if the proportion of "dynamic" points is above a certain level (30% of total number of points), otherwise static. Thresholds to identify if an object is dynamic were deliberately chosen as mentioned above, to be more conservative as the system is flexible to model a static object as dynamic and estimate a zero motion at every time step, however, the opposite would degrade the system's performance.

• Instance-level object segmentation only provides single-image object labels. Objects then need to be tracked across frames and their motion models propagated over time. We propose to use optical flow to associate point labels across frames. A point label is the same as the unique object identifier on which the point was sampled. We maintain a finite tracking label set $\mathscr{L} \subset \mathbb{N}$, where $l \in \mathscr{L}$ starts from $l = 1$ for the first detected moving object in the scene. The number of elements in $\mathscr{L}$ increases as more moving objects are being detected. Static objects and background are labelled with $l = 0$.

Ideally, for each detected object in frame k, the labels of all its points should be uniquely aligned with the labels of their correspondences in frame $k-1$. However, in practice this is affected by the noise, image boundaries and occlusions. To overcome this, we assign all the points with the label that

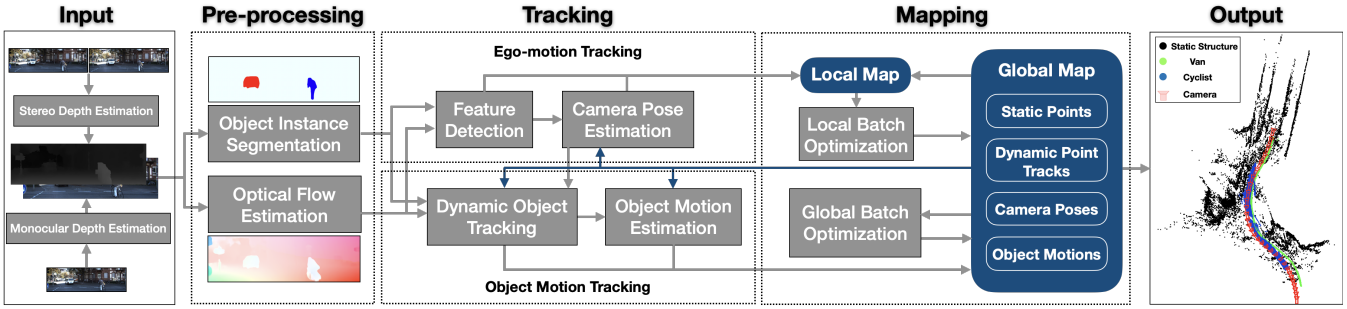

Fig. 4: **Overview of our VDO-SLAM system.** Input images are first pre-processed to generate instance-level object segmentation and dense optical flow. These are then used to track features on static background structure and dynamic objects. Camera poses and object motions estimated from feature tracks are then refined in a global batch optimisation, and a local map is maintained and updated with every new frame. The system outputs camera poses, static structure, tracks of dynamic objects, and estimates of their pose changes over time.

appears most in their correspondences. For a dynamic object, if the most frequent label in the previous frame is 0, it means that the object starts to move, appears in the scene at the boundary, or reappears from occlusion. In this case, the object is assigned a new tracking label.

4) Object Motion Estimation: As mentioned above, objects normally appear in small portions in the scene, which makes it hard to get sufficient sparse features to track and estimate their motions robustly. We sample every third point within an object mask, and track them across frames. Similar to the camera pose estimation, only inlier points are saved into the map and used for tracking in the next frame. When the number of tracked object points decreases below a certain level, new object points are sampled and added. We follow the same method as discussed in Section IV-B2 to generate an initial object motion model.

C. Mapping

In the mapping component, a global map is constructed and maintained. Meanwhile, a local map is extracted from the global map, which is based on the current time step and a window of previous time steps. Both maps are updated via a batch optimisation process.

1) Local Batch Optimisation: We maintain and update a local map. The goal of the local batch optimisation is to ensure accurate camera pose estimates are provided to the global batch optimisation. The camera pose estimation has a big influence on the accuracy of the object motion estimation and the overall performance of the algorithm. The local map is built using a fixed-size sliding window containing the information of the last n_w frames, where n_w is the window size and is set to 20 in this paper. Local maps share some common information; this defines the overlap between the different windows. We choose to only locally optimise the camera poses and static structure within the window size, as locally optimising the dynamic structure does not bring any benefit to the optimisation unless a hard constraint (e.g. a constant object motion) is assumed within the window. However, the system is able to incorporate static and dynamic structure in the local mapping if needed. When a local map is constructed, similarly, a factor graph optimisation is performed to refine all the variables within the local map, and then update them back into the global map.

2) Global Batch Optimisation: The output of the tracking component and the local batch optimisation consists of the camera pose, the object motions and the inlier structure. These are saved in a global map that is constructed with all the previous time steps and is continually updated with every new frame. A factor graph is constructed based on the global map after all input frames have been processed. To effectively explore the temporal constraints, only points that have been tracked for more than 3 instances are added into the factor graph. The graph is formulated as an optimisation problem as described in Section III-C. The optimisation results serve as the output of the whole system.

3) From Mapping to Tracking: Maintaining the map provides history information to the estimate of the current state in the tracking module, as shown in Fig. 4 with blue arrows going from the global map to multiple components in the tracking module of the system. Inlier points from the last frame are leveraged to track correspondences in the current frame and estimate camera pose and object motions. The last camera and object motion also serve as possible prior models to initialise the current estimation as described in Section IV-B2 and IV-B4. Furthermore, object points help associate semantic masks across frames to ensure robust tracking of objects, by propagating their previously segmented masks in case of "indirect occlusion" resulting from the failure of semantic object segmentation.

V. Experiments

We evaluate VDO-SLAM in terms of camera motion, object motion and velocity, as well as object tracking performance. The evaluation is done on the Oxford Multimotion Dataset [65] for indoor, and KITTI Tracking dataset [66] for outdoor scenarios, with comparison to other state-of-the-art methods, including MVO [51], ClusterVO [52], DynaSLAM II [49] and CubeSLAM [24]. Due to the non-deterministic nature in running the proposed system, such as RANSAC processing, we run each sequence 5 times and take median values as the

demonstrating results. All the results are obtained by running the proposed system in default parameter setup. Our open-source implementation includes the demo YAML files and instructions to run the system in both datasets.

A. Deep Model Setup

We adopt a learning-based instance-level object segmentation, Mask R-CNN [67], to generate object segmentation masks. The model of this method is trained on COCO dataset [68], and is directly used in this work without any fine-tuning. For dense optical flow, we leverage a state-of-the-art method; PWC-Net [12]. The model is trained on FlyingChairs dataset [69], and then fine-tuned on Sintel [70] and KITTI training datasets [71]. To generate depth maps for a "monocular" version of our proposed system, we apply a learning-based monocular depth estimation method, MonoDepth2 [72]. The model is trained on Depth Eigen split [73] excluding the tested data in this paper. Feature detection is done using FAST [74] implemented in [75]. All the above methods are applied using the default parameters.

B. Error Metrics

We use a pose change error metric to evaluate the estimated SE(3) motion, i.e., given a ground truth motion transform $\mathbf{T}$ and a corresponding estimated motion $\hat{\mathbf{T}}$, where $\mathbf{T} \in \mathrm{SE}(3)$ could be either a camera relative pose or an object motion. The pose change error is computed as: $\mathbf{E} = \hat{\mathbf{T}}^{-1}\mathbf{T}$. This is similar to Relative Pose Error [76], while we set the time interval $\Delta = 1$ (per frame), because the trajectory of different object in a sequence varies from each other and are normally much shorter than the camera trajectory.

The translational error E_t (meter) is computed as the L_2 norm of the translational component of $\mathbf{E}$. The rotational error E_r (degree) is calculated as the angle of rotation in an axis-angle representation of the rotational component of $\mathbf{E}$. For different camera time steps and different objects in a sequence, we compute the root mean squared error (RMSE) for camera poses and object motions, respectively. The object pose change in body-fixed frame is obtained by transforming the pose change ${}_{k-1}^{0}\mathbf{H}_k$ in the inertial frame into the body frame using the object pose ground-truth

$$ {}_{k-1}^{L_{k-1}}\mathbf{H}_k = {}^{0}\mathbf{L}_{k-1}^{-1}\ {}_{k-1}^{0}\mathbf{H}_k\ {}^{0}\mathbf{L}_{k-1}. \tag{23} $$

We also evaluate the object speed error. The linear velocity of a point on the object, expressed in the inertial frame, can be estimated by applying the pose change ${}_{k-1}^{0}\mathbf{H}_k$ and taking the difference

$$ \begin{aligned} \mathbf{v} \approx {}^{0}\mathbf{m}_k^i - {}^{0}\mathbf{m}_{k-1}^i &= \left({}_{k-1}^{0}\mathbf{H}_k - \mathbf{I}_4\right){}^{0}\mathbf{m}_{k-1}^i \\ &= {}_{k-1}^{0}\mathbf{t}_k - \left(\mathbf{I}_3 - {}_{k-1}^{0}\mathbf{R}_k\right){}^{0}\mathbf{m}_{k-1}^i. \end{aligned} \tag{24} $$

To get a more reliable measurement, we average over all points on an object at a certain time. Define $\mathbf{c}_{k-1} := \frac{1}{n}\sum \mathbf{m}_{k-1}^i$ for all n points on an object at time $k-1$. Then

$$ \begin{aligned} \mathbf{v} &\approx \frac{1}{n}\sum_{i=1}^{n}\left({}_{k-1}^{0}\mathbf{t}_k - \left(\mathbf{I}_3 - {}_{k-1}^{0}\mathbf{R}_k\right){}^{0}\mathbf{m}_{k-1}^i\right) \\ &= {}_{k-1}^{0}\mathbf{t}_k - \left(\mathbf{I}_3 - {}_{k-1}^{0}\mathbf{R}_k\right)\mathbf{c}_{k-1}. \end{aligned} \tag{25} $$

Then the speed error E_s between the estimated $\hat{\mathbf{v}}$ and the ground truth $\mathbf{v}$ velocities can be calculated as: $E_s = |\hat{\mathbf{v}}| - |\mathbf{v}|$.

C. Oxford Multimotion Dataset

The recent Oxford Multimotion Dataset [65] contains sequences from a moving stereo or RGB-D camera sensor observing multiple swinging boxes or toy cars in an indoor scenario. Ground truth trajectories of the camera and moving objects are obtained via a Vicon motion capture system. We only choose the swinging boxes sequence (500 frames) for evaluation, since results of real driving scenarios are evaluated on KITTI dataset. Note that, the trained model for instance segmentation cannot be applied to this dataset directly, since the training data (COCO) does not contain the class of square box. Instead, we use Otsu's method [77], together with color information and multi-label processing to segment the boxes, which works very well for the simple setup of this dataset (color boxes that are highly distinguishable from the background). Table I shows results compared to the state-of-the-art MVO [51] and ClusterVO [52], with data provided by the authors, respectively. As they are both visual odometry systems without global refinement, we switch off the batch optimisation module in our system and generate our results for fair comparison. We use the error metrics described in Section V-B.

Compared to MVO, our proposed method achieves better accuracy in the estimation of camera pose (35%) and motion of the swinging boxes, top-left (15%) and bottom-left (40%). We obtain slightly higher errors when there is spinning rotational motion of the object observed, in particular the top-right swinging and rotating box (in translation only), and the bottom-right rotating box. We believe that this is due to using an optical flow algorithm that is not well optimised for self-rotating objects. The consequence of this is poor estimation of point motion and consequent degradation of the overall object tracking performance. Even with the associated performance loss for rotating objects, the benefits of dense optical flow motion estimation is clear in the other metrics. Our method performs slightly worse than ClusterVO in the estimate of camera pose, and the translation of bottom-right rotating box. Other than that, we achieve more than twice improvements against ClusterVO in the estimate of object motions.

An illustrative result of the trajectory output of our algorithm on Oxford Multimotion Dataset is shown in Fig. 5. Tracks of dynamic features on swinging boxes visually correspond to the actual motion of the boxes. This can be clearly seen in the swinging motion of the bottom-left box shown with purple color in Fig. 5.

D. KITTI Tracking Dataset

The KITTI Tracking Dataset [66] contains 21 sequences in total with ground truth information about camera and object poses. Among these sequences, some are not included in the evaluation of our system; as they contain no moving objects (static only scenes) or only contain pedestrians that are non-rigid objects, which is outside the scope of this work. Note that, as only rotation around Y-axis is provided in the ground

TABLE I: Comparison versus MVO [51] and ClusterVO [52] for camera pose and object motion estimation accuracy on the sequence of swinging_4_unconstrained sequence in Oxford Multi-motion dataset. Bold numbers indicate the better results.

	VDO-SLAM		MVO		ClusterVO	
	E_r(deg)	E_t(m)	E_r(deg)	E_t(m)	E_r(deg)	E_t(m)
Camera	0.7709	0.0112	1.1948	0.0314	**0.7665**	**0.0066**
Top-left Swinging Box	**1.1889**	**0.0207**	1.4553	0.0288	3.2537	0.0673
Top-right Swinging and rotating Box	**0.7631**	0.0132	0.8992	**0.0130**	3.5308	0.0256
Bottom-left Swinging Box	**0.9153**	**0.0149**	1.4949	0.0261	4.9146	0.0763
Bottom-right Rotating Box	0.8469	0.0192	**0.7815**	**0.0115**	4.0675	0.0144

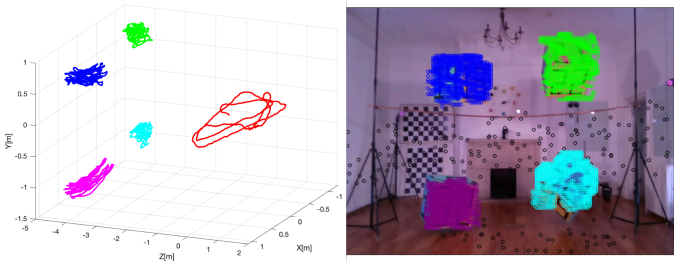

Fig. 5: **Qualitative results of our method on Oxford Multimotion Dataset.** (Left) The 3D trajectories of camera (red) and centres of the four boxes. (Right) Detected points on static background and object body. Black color corresponds to static points and features on each object are shown in a different color.

truth object poses, we assign zeros to the other two axes for the convenience of full motion evaluation.

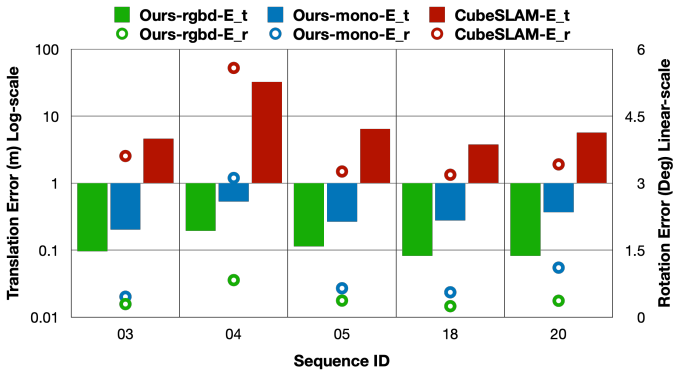

Fig. 6: **Accuracy of object motion estimation of our method compared to CubeSLAM ([24]).** The color bars refer to translation error that is corresponding to the left Y-axis in log-scale. The circles refer to rotation error, which corresponds to the right Y-axis in linear-scale.

1) Camera Pose and Object Motion: Table II demonstrates results of both camera pose and object motion estimation in nine sequences, compared to DynaSLAM II [49] and CubeSLAM [24]. Results of DynaSLAM II is obtained directly from their paper, where only the evaluation of camera pose is available. We initially tried to evaluate CubeSLAM ourselves with the default provided parameters, however errors were much higher, and hence we only report results of five sequences provided by the authors of CubeSLAM after some correspondences. As CubeSLAM is for monocular camera, we also compute results of a learning-based monocular version of our proposed method (as mentioned in Section IV) for fair comparison.

Our proposed method achieves competitive and high accuracy in comparison with DynaSLAM II for the estimate of camera pose. In particular, our method obtains slightly lower rotational errors while higher translational errors than DynaSLAM II. We believe the difference in accuracy is due to the underlying formulations in estimating camera pose. When compared to CubeSLAM, our RGB-D version gets lower errors in camera pose, while our learning-based monocular version slightly higher. We believe the weak performance of monocular version is because the model does not capture the scale of depth accurately with only monocular input. Nevertheless, both versions obtain consistently lower errors in object motion estimation. In particular, as demonstrated in Fig. 6, the translation and rotation errors in CubeSLAM are all above 3 meters and 3 degrees, with errors reaching 32 meters and 5 degrees in extreme cases respectively. However, our translation errors vary between 0.1-0.3 meters and rotation errors between 0.2-1.5 degrees in the case of RGB-D, and 0.1-0.3 meters, and 0.4-3.1 degrees in the case of learning-based monocular, which indicates that our object motion estimation achieves an order of magnitude improvements in most cases. In general, the results suggest that point-based object motion/pose estimation methods is more robust and accurate than those using high-level geometric models, probably due to the fact that geometric model extraction could lead to losing information and introducing more uncertainty.

2) Object Tracking and Velocity: We also demonstrate the performance of tracking dynamic objects, and show results of object speed estimation, which is an important information for autonomous driving applications. Fig. 7 illustrates results of object tracking length and object speed for some selected objects (tracked for over 20 frames) in all the tested sequences. Our system is able to track most objects for more than 80% of their occurrence in the sequence. Moreover, our estimated objects speed is always consistently close to the ground truth.

3) Qualitative Results: Fig. 8 illustrates the output of our system for three of the KITTI sequences. The proposed system is able to output the camera poses, along with the static structure and dynamic tracks of every detected moving object in the scene in a spatiotemporal map representation.

TABLE II: Comparison versus DynaSLAM II [49] and CubeSLAM [24] for camera pose and object motion estimation accuracy on nine sequences with moving objects drawn from the KITTI dataset. Bold numbers indicate the better result.

	DynaSLAM II		VDO-SLAM (RGB-D)				VDO-SLAM (Monocular)				CubeSLAM			
	Camera		Camera		Object		Camera		Object		Camera		Object	
Seq	E_r(deg)	E_t(m)	E_r(deg)	E_t(m)	E_r(deg)	E_t(m)	E_r(deg)	E_t(m)	E_r(deg)	E_t(m)	E_r(deg)	E_t(m)	E_r(deg)	E_t(m)
00	**0.06**	**0.04**	0.0741	0.0674	**1.0520**	**0.1077**	0.1830	0.1847	2.0021	0.3827	-	-	-	-
01	0.04	**0.05**	**0.0382**	0.1220	**0.9051**	**0.1573**	0.1772	0.4982	1.1833	0.3589	-	-	-	-
02	0.02	**0.04**	**0.0182**	0.0445	**1.2359**	**0.2801**	0.0496	0.0963	1.6833	0.4121	-	-	-	-
03	0.04	**0.06**	**0.0311**	0.0816	**0.2919**	**0.0965**	0.1065	0.1505	0.4570	0.2032	0.0498	0.0929	3.6085	4.5947
04	0.06	**0.07**	**0.0482**	0.1114	**0.8288**	**0.1937**	0.1741	0.4951	3.1156	0.5310	0.0708	0.1159	5.5803	32.5379
05	0.03	**0.06**	**0.0219**	0.0932	**0.3705**	**0.1140**	0.0506	0.1368	0.6464	0.2669	0.0342	0.0696	3.2610	6.4851
06	**0.04**	0.02	0.0488	**0.0186**	**1.0803**	**0.1158**	0.0671	0.0451	2.0977	0.2394	-	-	-	-
18	**0.02**	**0.05**	0.0211	0.0749	**0.2453**	**0.0825**	0.1236	0.3551	0.5559	0.2774	0.0433	0.0510	3.1876	3.7948
20	0.04	**0.07**	**0.0271**	0.1662	**0.3663**	**0.0824**	0.3029	1.3821	1.1081	0.3693	0.1348	0.1888	3.4206	5.6986

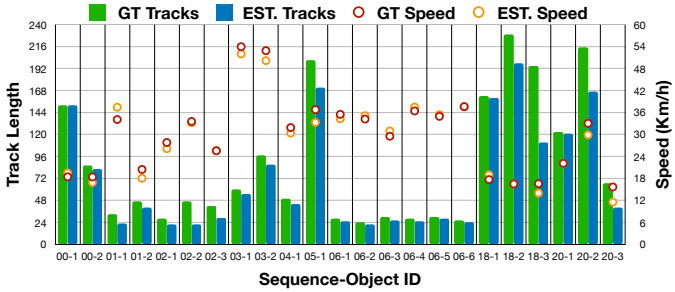

Fig. 7: **Tracking performance and speed estimation.** Results of object tracking length and object speed for some selected objects (tracked for over 20 frames), due to limited space. The color bars represent the length of object tracks, which is corresponding to the left Y-axis. The circles represent object speeds, which is corresponding to the right Y-axis. GT refers to ground truth, and EST. refers to estimated values.

E. Discussion

Apart from the extensive evaluation in Section V-D and V-C, we also provide detailed experimental results to prove the effectiveness of key modules in our proposed system. Finally, the computational cost of the proposed system is discussed.

TABLE III: The number of points tracked for more than five frames on the nine sequences of the KITTI dataset. Bold numbers indicate the better results. Underlined bold numbers indicate an order of magnitude increase in number.

	Background		Object	
Seq	Motion Only	Joint	Motion Only	Joint
00	1798	**12812**	1704	**7162**
01	237	**5075**	907	**4583**
02	7642	**10683**	52	**1442**
03	778	**12317**	343	**3354**
04	9913	**25861**	339	**2802**
05	713	**11627**	2363	**2977**
06	7898	**11048**	482	**5934**
18	4271	**22503**	5614	**14989**
20	9838	**49261**	9282	**13434**

1) Robust Tracking of Points: The graph optimisation explores the spacial and temporal information to refine the camera poses and the object motions, as well as the static and dynamic structure. This process requires robust tracking of good points in terms of both quantity and quality. This was achieved by refining the estimated optical flow jointly with the motion estimation, as discussed in Section III-B3. The effectiveness of joint optimisation is shown by comparing a baseline method that only optimises for the motion (Motion Only) using (9) for camera motion or (11) for object motion, and the improved method that optimises for both the motion and the optical flow (Joint) using (13) or (15). Table III demonstrates that the joint method obtains considerably more points that are tracked for long periods.

TABLE IV: Average camera pose and object motion errors over the nine sequences of the KITTI dataset. Bold numbers indicate the better results.

	Motion Only		Joint	
	E_r(deg)	E_t(m)	E_r(deg)	E_t(m)
Camera	0.0412	0.0987	**0.0365**	**0.0866**
Object	1.0179	0.1853	**0.7085**	**0.1367**

Using the tracked points given by the joint estimation process leads to better estimation of both camera pose and object motion. As demonstrated in Table IV, an improvement of about 10% (camera) and 25% (object) in both translation and rotation errors was observed over the nine sequences of the KITTI dataset shown above.

2) Robustness against Non-direct Occlusion: The mask segmentation may fail in some cases, due to direct or indirect occlusions (illumination change, etc.). Thanks to the mask propagating method described in Section IV-C3, our proposed system is able to handle mask failure cases caused by indirect occlusions. Fig. 9 demonstrates an example of tracking a white van for 80 frames, where the mask segmentation fails in 33 frames. Despite the object segmentation failure, our system is still continuously able to track the van, and estimate its speed with an average error of 2.64 km/h across the whole sequence. Speed errors in the second half of the sequence are higher due to partial direct occlusions, and increased distance to the object getting farther away from the camera.

3) Global Refinement on Object Motion: Initial object motion estimation (in the tracking component of the system) is independent between frames, since it is purely related to the sensor measurements. As illustrated in Fig. 10, the blue curve describes an initial object speed estimate of a wagon observed

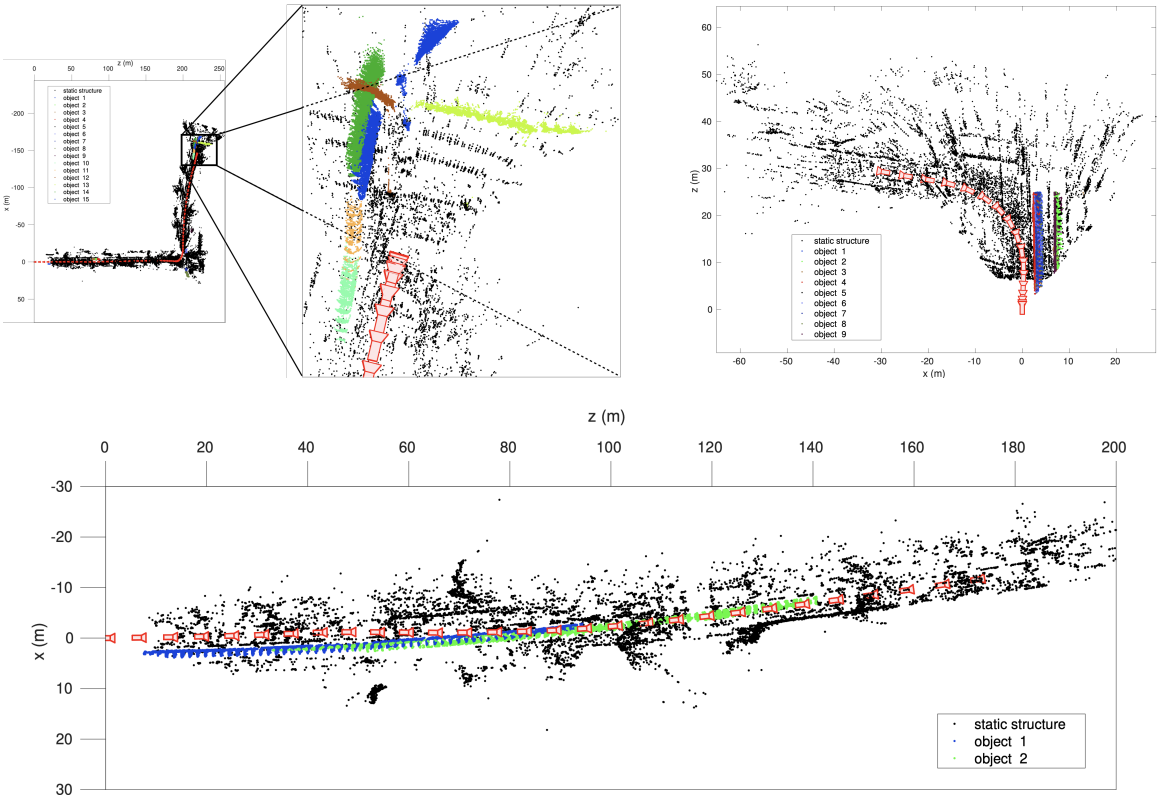

Fig. 8: **Illustration of system output; a dynamic map with camera poses, static background structure, and tracks of dynamic objects.** Sample results of VDO-SLAM on KITTI sequences. Black represents static background, and each detected object is shown in a different colour. Top left figure represents Seq.01 and a zoom-in on the intersection at the end of the sequence, top right figure represents Seq.06 and bottom figure represents Seq.03.

for 55 frames in sequence 03 of the KITTI tracking dataset. As seen in the figure, the speed estimation is not smooth and large errors occur towards the second half of the sequence. This is mainly caused by the increased distance to the object getting farther away from the camera, and its structure only occupying a small portion of the scene. In this case, the object motion estimation from sensor measurements solely becomes challenging and error-prone. Therefore, we formulate a factor graph and refine the motions together with the static and dynamic structure as discussed in Section III-C. The green curve in Fig. 10 shows the object speed results after the global refinement, which becomes smoother in the first half of the sequence and is significantly improved in the second half.

Fig. 11 demonstrates the average improvement for all objects in each sequence of KITTI dataset. With graph optimization, the errors can be reduced up to 39% in translation and 55% in rotation. Interestingly, the translation errors in Seq.18 and Seq.20 increase slightly. We believe it is because the vehicles keep alternating between acceleration and deceleration due to the heavy traffic jams in both sequences, which strongly violates the smooth motion constraint that is set for general cases.

4) Computational Analysis: Finally, we provide the computational analysis of our system. The experiments are carried out on an Intel Core i7 2.6 GHz laptop computer with 16 GB RAM. The object semantic segmentation and dense optical flow computation times depend on the GPU power and the CNN model complexity. Many current state-of-the-art algorithms can run in real time ([30], [78]). In this paper, the semantic segmentation and optical flow results are produced off-line as input to the system. The SLAM system is implemented in C++ on CPU using a modified version of g2o as a back-end [79]. We show the computational time in Table V for both datasets. Overall, the tracking part of our proposed system is able to run at the frame rate of 5-8 fps depending on the number of detected moving objects, which can be improved by employing parallel implementation. The runtime of the global batch optimisation strongly depends on the amount of camera poses (number of frames), and objects (density in terms of the number of dynamic objects observed per frame) present in the scene.

VI. Conclusion

In this paper, we have presented VDO-SLAM, a novel dynamic feature-based SLAM system that exploits image-based semantic information in the scene with no additional knowledge of the object pose or geometry, to achieve simultaneous localisation, mapping and tracking of dynamic objects. The system consistently shows robust and accurate results on indoor and challenging outdoor datasets, and achieves state-of-the-art performance in object motion estimation. We believe

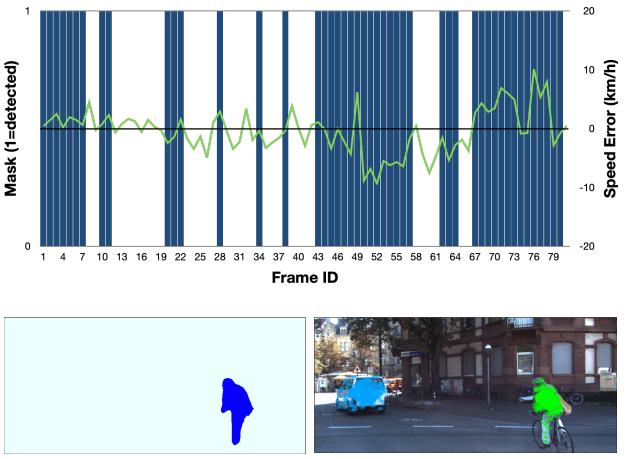

Fig. 9: **Robustness in tracking performance and speed estimation in case of semantic segmentation failure.** An example of tracking performance and speed estimation for a white van (ground-truth average speed 20km/h) in Seq.00. (Top) Blue bars represent a successful object segmentation, and green curves refer to the object speed error. (Bottom-left) An illustration of semantic segmentation failure on the van. (Bottom-right) Result of propagating the previously tracked features on the van by our system.

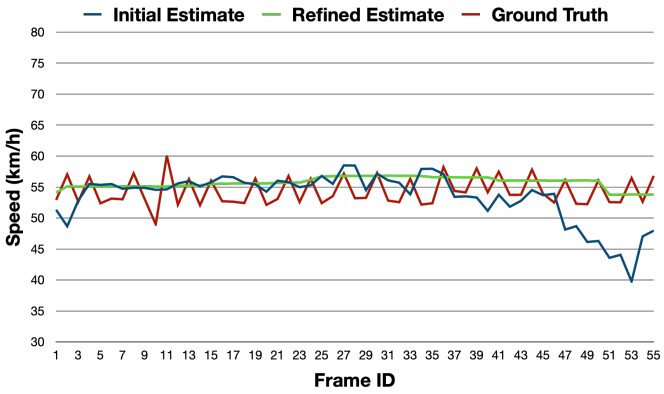

Fig. 10: **Global refinement effect on object speed estimation.** The initial (blue) and refined (green) estimated speeds of a wagon in Seq.03, travelling along a straight road, compared to the ground truth speed (red). Note the ground truth speed is slightly fluctuating. We believe it is due to the ground truth object poses being approximated from lidar scans.

the high performance accuracy achieved in object motion estimation is due to the fact that our system is a feature-based system. Feature points remain to be the easiest to detect, track and integrate within a SLAM system, and that require the front-end to have no additional knowledge about the object model, or explicitly provide any information about its pose.

An important issue to be reduced is the computational complexity of SLAM with dynamic objects. In long-term applications, different techniques can be applied to limit the growth of the graph ([80], [81]). In fact, history summarisation/deletion of map points pertaining to dynamic objects

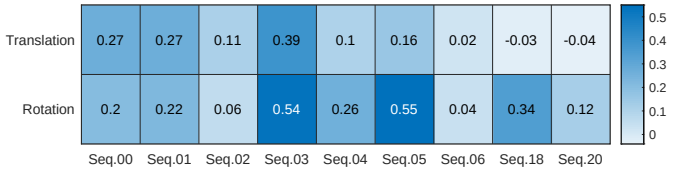

Fig. 11: **Improvement on object motion after graph optimization.** The numbers in the heatmap show the ratio of decrease in error on the nine sequences of the KITTI dataset.

TABLE V: Runtime of different system components for both datasets. The time cost of every component is averaged over all frames and sequences, except for the object motion estimation and object motion estimation that are averaged over the number of objects.

Dataset	Tasks	Runtime (mSec)
KITTI	Feature Detection	16.2550
	Camera Pose Estimation	52.6542
	Dynamic Object Tracking (avg/object)	8.2980
	Object Motion Estimation (avg/object)	22.9081
	Map and Mask Updating	22.1830
	Local Batch Optimisation	18.2828
OMD	Feature Detection	7.5220
	Camera Pose Estimation	32.0909
	Dynamic Object Tracking (avg/object)	7.0134
	Object Motion Estimation (avg/object)	19.5280
	Map and Mask Updating	30.3153
	Local Batch Optimisation	15.3414

observed far in the past seems to be a natural step towards a long-term SLAM system in highly dynamic environments.

ACKNOWLEDGEMENTS

This research is supported by the Australian Research Council through the Australian Centre of Excellence for Robotic Vision (CE140100016), and the Sydney Institute for Robotics and Intelligent Systems. The authors would like to thank Mr. Ziang Cheng and Mr. Huangying Zhan for providing help in preparing the testing datasets.

REFERENCES

[1] D. Hahnel, D. Schulz, and W. Burgard, "Map Building with Mobile Robots in Populated Environments," in *International Conference on Intelligent Robots and Systems (IROS)*, vol. 1. IEEE, 2002, pp. 496–501.

[2] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map Building with Mobile Robots in Dynamic Environments," in *International Conference on Robotics and Automation (ICRA)*, vol. 2. IEEE, 2003, pp. 1557–1563.

[3] D. F. Wolf and G. S. Sukhatme, "Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, 2005.

[4] H. Zhao, M. Chiba, R. Shibasaki, X. Shao, J. Cui, and H. Zha, "SLAM in a Dynamic Large Outdoor Environment using a Laser Scanner," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2008, pp. 1455–1462.

[5] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes," *Robotics and Automation Letters (RAL)*, vol. 3, no. 4, pp. 4076–4083, 2018.

[6] C.-C. Wang, C. Thorpe, and S. Thrun, "Online Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects: Theory and Results from a Ground Vehicle in Crowded Urban Areas," in *International Conference on Robotics and Automation (ICRA)*, vol. 1. IEEE, 2003, pp. 842–849.

[7] I. Miller and M. Campbell, "Rao-blackwellized Particle Filtering for Mapping Dynamic Environments," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2007, pp. 3862–3869.

[8] J. G. Rogers, A. J. Trevor, C. Nieto-Granda, and H. I. Christensen, "SLAM with Expectation Maximization for Moveable Object Tracking," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 2077–2082.

[9] A. Kundu, K. M. Krishna, and C. Jawahar, "Realtime Multibody Visual SLAM with a Smoothly Moving Monocular Camera," in *International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2080–2087.

[10] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 756–771.

[11] D. Sun, S. Roth, and M. J. Black, "Secrets of Optical Flow Estimation and Their Principles," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 2432–2439.

[12] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[13] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNET 2.0: Evolution of Optical Flow Estimation with Deep Networks," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 2462–2470.

[14] C. Vogel, K. Schindler, and S. Roth, "Piecewise Rigid Scene Flow," in *International Conference on Computer Vision (ICCV)*. IEEE, 2013, pp. 1377–1384.

[15] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3061–3070.

[16] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning Scene Flow in 3D Point Clouds," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 529–537.

[17] H. Jiang, D. Sun, V. Jampani, Z. Lv, E. Learned-Miller, and J. Kautz, "SENSE: A Shared Encoder Network for Scene-flow Estimation," in *International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 3195–3204.

[18] P. de la Puente and D. Rodríguez-Losada, "Feature Based Graph-SLAM in Structured Environments," *Autonomous Robots*, vol. 37, no. 3, pp. 243–260, 2014.

[19] M. Kaess, "Simultaneous Localization and Mapping with Infinite Planes," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4605–4611.

[20] M. Henein, M. Abello, V. Ila, and R. Mahony, "Exploring the Effect of Meta-structural Information on the Global Consistency of SLAM," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1616–1623.

[21] M. Hsiao, E. Westman, G. Zhang, and M. Kaess, "Keyframe-based Dense Planar SLAM," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5110–5117.

[22] B. Mu, S.-Y. Liu, L. Paull, J. Leonard, and J. P. How, "SLAM with Objects using a Nonparametric Pose Graph," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4602–4609.

[23] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013, pp. 1352–1359.

[24] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D Object SLAM," *Transactions on Robotics (T-RO)*, vol. 35, no. 4, pp. 925–938, 2019.

[25] D. Gálvez-López, M. Salas, J. D. Tardós, and J. Montiel, "Real-time Monocular Object SLAM," *Robotics and Autonomous Systems*, vol. 75, pp. 435–449, 2016.

[26] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," *arXiv:1603.00831 [cs]*, Mar. 2016, arXiv: 1603.00831. [Online]. Available: http://arxiv.org/abs/1603.00831

[27] A. Byravan and D. Fox, "SE3-Nets: Learning Rigid Body Motion using Deep Neural Networks," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 173–180.

[28] P. Wohlhart and V. Lepetit, "Learning Descriptors for Object Recognition and 3D Pose Estimation," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3109–3118.

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2018.

[30] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time Instance Segmentation," in *International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 9157–9166.

[31] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic SLAM: The Need for Speed," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2123–2129.

[32] J. Huang, S. Yang, Z. Zhao, Y. Lai, and S. Hu, "ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation," in *International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 5874–5883.

[33] J. Zhang, M. Henein, R. Mahony, and V. Ila, "Robust Ego and Object 6-DoF Motion Estimation and Tracking," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5017–5023.

[34] P. F. Alcantarilla, J. J. Yebes, J. Almazán, and L. M. Bergasa, "On Combining Visual SLAM and Dense Scene Flow to Increase the Robustness of Localization and Mapping in Dynamic Environments," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 1290–1297.

[35] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust Monocular SLAM in Dynamic Environments," in *International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2013, pp. 209–218.

[36] P. Kaveti and H. Singh, "A Light Field Front-end for Robust SLAM in Dynamic Environments," *arXiv preprint arXiv:2012.10714*, 2020.

[37] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous Localization, Mapping and Moving Object Tracking," *International Journal of Robotics Research (IJRR)*, vol. 26, no. 9, pp. 889–916, 2007.

[38] N. D. Reddy, P. Singhal, V. Chari, and K. M. Krishna, "Dynamic Body VSLAM with Semantic Constraints," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1897–1904.

[39] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust Dense Mapping for Large-Scale Dynamic Environments," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[40] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013, pp. 1352–1359.

[41] K. Tateno, F. Tombari, and N. Navab, "When 2.5D is Not Enough: Simultaneous Reconstruction, Segmentation and Recognition on Dense SLAM," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2295–2302.

[42] E. Sucar, K. Wada, and A. Davison, "NodeSLAM: Neural Object Descriptors for Multi-View Shape Reconstruction," *arXiv preprint arXiv:2004.04485*, 2020.

[43] M. Runz, M. Buffier, and L. Agapito, "MaskFusion: Real-time Recognition, Tracking and Reconstruction of Multiple Moving Objects," in *International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2018, pp. 10–20.

[44] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "MID-Fusion: Octree-based Object-level Multi-instance Dynamic SLAM," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5231–5237.

[45] M. Hosseinzadeh, K. Li, Y. Latif, and I. Reid, "Real-time Monocular Object-model Aware Sparse SLAM," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7123–7129.

[46] L. Nicholson, M. Milford, and N. Sünderhauf, "QuadricSLAM: Dual Quadrics from Object Detections as Landmarks in Object-oriented SLAM," *Robotics and Automation Letters (RAL)*, vol. 4, no. 1, pp. 1–8, 2018.

[47] P. Li, T. Qin, *et al.*, "Stereo Vision-based Semantic 3D Object and Ego-motion Tracking for Autonomous Driving," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 646–661.

[48] P. Li, J. Shi, and S. Shen, "Joint Spatial-temporal Optimization for Stereo 3D Object Tracking," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, pp. 6877–6886.

[49] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: Tightly-coupled Multi-object Tracking and SLAM," *Robotics and Automation Letters (RAL)*, vol. 6, no. 3, pp. 5191–5198, 2021.

[50] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Motion-based Detection and Tracking in 3D Lidar Scans," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4508–4513.

[51] K. M. Judd, J. D. Gammell, and P. Newman, "Multimotion Visual Odometry (MVO): Simultaneous Estimation of Camera and Third-party Motions," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3949–3956.

[52] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu, "ClusterVO: Clustering Moving Instances and Estimating Visual Odometry for Self and Surroundings," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, pp. 2168–2177.

[53] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[54] G. S. Chirikjian, R. Mahony, S. Ruan, and J. Trumpf, "Pose Changes from a Different Point of View," in *The ASME International Design Engineering Technical Conferences (IDETC)*. ASME, 2017.

[55] D. Nistér, O. Naroditsky, and J. Bergen, "Visual Odometry," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1. IEEE, 2004, pp. I–I.

[56] P. J. Huber, "Robust Estimation of a Location Parameter," in *Breakthroughs in Statistics*. Springer, 1992, pp. 492–518.

[57] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," *International Journal of Robotics Research (IJRR)*, vol. 25, no. 12, pp. 1181–1203, 2006.

[58] S. Agarwal, K. Mierle, and Others, "Ceres Solver," http://ceres-solver.org, 2012.

[59] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental Smoothing and Mapping using the Bayes Tree," *International Journal of Robotics Research (IJRR)*, p. 0278364911430419, 2011.

[60] L. Polok, V. Ila, M. Solony, P. Smrz, and P. Zemcik, "Incremental Block Cholesky Factorization for Nonlinear Least Squares in Robotics," in *Robotics: Science and Systems (RSS)*, Berlin, Germany, June 2013.

[61] V. Ila, L. Polok, M. Šolony, and P. Svoboda, "SLAM++-A Highly Efficient and Temporally Scalable Incremental SLAM Framework," *International Journal of Robotics Research (IJRR)*, vol. Online First, no. 0, pp. 1–21, 2017.

[62] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 756–771.

[63] T. Ke and S. I. Roumeliotis, "An Efficient Algebraic Solution to the Perspective-three-point Problem," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[64] Z. Lv, K. Kim, A. Troccoli, J. Rehg, and J. Kautz, "Learning Rigidity in Dynamic Scenes with a Moving Camera for 3D Motion Field Estimation," in *European Conference on Computer Vision (ECCV)*. Springer, 2018.

[65] K. M. Judd and J. D. Gammell, "The Oxford Multimotion Dataset: Multiple SE(3) Motions with Ground Truth," *Robotics and Automation Letters (RAL)*, vol. 4, no. 2, pp. 800–807, 2019.

[66] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1231–1237, 2013.

[74] E. Rosten and T. Drummond, "Machine Learning for High-speed Corner Detection," in *European Conference on Computer Vision (ECCV)*. Springer, 2006, pp. 430–443.

[67] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 2980–2988.

[68] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[69] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 4040–4048.

[70] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A Naturalistic Open Source Movie for Optical Flow Evaluation," in *European Conference on Computer Vision (ECCV)*. Springer, 2012, pp. 611–625.

[71] A. Geiger, P. Lenz, and R. Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012.

[72] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into Self-supervised Monocular Depth Estimation," in *International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 3828–3838.

[73] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-scale Deep Network," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 2366–2374.

[75] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," in *International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2564–2571.

[76] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 573–580.

[77] N. Otsu, "A Threshold Selection Method from Gray-level Histograms," *Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[78] T.-W. Hui, X. Tang, and C. C. Loy, "A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization." IEEE, 2020. [Online]. Available: http://mmlab.ie.cuhk.edu.hk/projects/LiteFlowNet/

[79] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A General Framework for Graph Optimization," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3607–3613.

[80] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige, "Double Window Optimisation for Constant Time Visual SLAM," in *International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2352–2359.

[81] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based Compact Pose SLAM," *Transactions on Robotics (T-RO)*, vol. 26, no. 1, pp. 78–93, 2010.