IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

DynoSAM: Open-Source Smoothing and Mapping Framework for Dynamic SLAM

Jesse Morris*, Yiduo Wang, Mikolaj Kliniewski and Viorela Ila

***Abstract*—Traditional Visual Simultaneous Localization and Mapping systems focus solely on static scene structures, overlooking dynamic elements in the environment. Although effective for accurate visual odometry in complex scenarios, these methods discard crucial information about moving objects. By incorporating this information into a Dynamic SLAM framework, the motion of dynamic entities can be estimated, enhancing navigation whilst ensuring accurate localization. However, the fundamental formulation of Dynamic SLAM remains an open challenge, with no consensus on the optimal approach for accurate motion estimation within a SLAM pipeline.**

Therefore, we developed *DynoSAM*, an open-source framework for Dynamic Objects SLAM that enables the efficient implementation, testing, and comparison of various Dynamic SLAM optimization formulations. We further propose a novel formulation that encodes rigid-body motion model in object pose estimation as well as an error metric agnostic to object frame definition. *DynoSAM* integrates static and dynamic measurements into a unified optimization problem solved using factor graphs, simultaneously estimating camera poses, static scene, object motion or poses, and object structures. We evaluate *DynoSAM* across diverse simulated and real-world datasets, achieving state-of-the-art motion estimation in indoor and outdoor environments, with substantial improvements over existing systems. Additionally, we demonstrate *DynoSAM*'s contributions to downstream applications, including 3D reconstruction of dynamic scenes and trajectory prediction, thereby showcasing potential for advancing dynamic object-aware SLAM systems. Code is open-sourced at https://github.com/ACFR-RPG/DynOSAM

***Keywords*—Dynamic SLAM, Mapping, RGBD Perception**

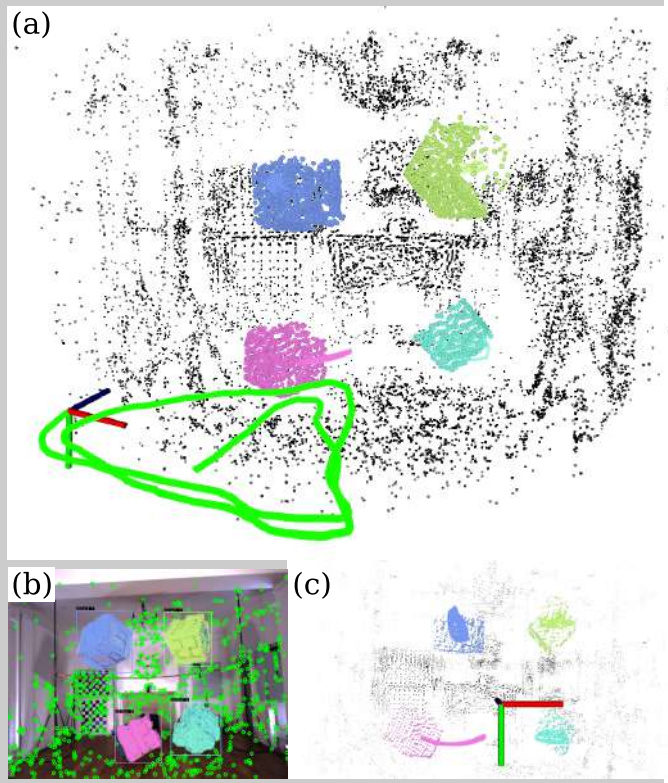


Fig. 1: DynoSAM is an open-source smoothing and mapping framework for Dynamic SLAM. **(a)** presents the system output, which includes camera and object trajectories, as well as the static and per-object dynamic map. **(b)** visualizes the feature-based front-end which performs multi-object tracking in addition to visual odometry. **(c)** shows the dynamic map from the camera's perspective, highlighting the estimated trajectory of each object and the tracked 3D points.

I. Introduction

Simultaneous Localization and Mapping (SLAM) has been a key research area for over three decades [1]. Despite significant advancements, most SLAM systems are designed with the assumption of a predominantly static environment [2], [3], [4]. This limitation presents challenges in real-world scenarios, as dynamic objects are common and must be accounted for.

Traditionally, SLAM systems classify sensor measurements associated with moving objects as outliers, excluding such data from the estimation process [5], [6], and in doing so, discard valuable information about dynamic elements. Incorporating these objects into the SLAM framework enables accurate modeling of dynamic environments [7], [8], which directly benefits navigation and task planning systems [9], [10] and enhances the overall robustness of SLAM [11], [12].

Although learning-based methods have advanced robotic capabilities in dynamic environments, especially in autonomous vehicle applications with access to large datasets, there remains a critical need for SLAM solutions in scenarios where motion models are unknown and data is scarce. These challenges are particularly pronounced when operating in unstructured environments [13] or space exploration applications [14].

As a result, there is increasing interest in extending SLAM systems to incorporate observations of dynamic entities and estimate their motions [1], [15], [16], [17], [18], [19]. In this paper we define the term *Dynamic SLAM* as SLAM solutions that optimize for object motions and/or poses in addition to robot trajectory and the static structure, to distinguish from those that exclude objects to be robust in dynamic environments [6], [20], [21]. This paper focuses on studying purely Visual SLAM (vSLAM) formulations, though we have further discussed IMU-integrated experiments and results.

There exists limited consensus in the community for how to best formulate the Dynamic SLAM problem. The seminal work of Henein *et al*. [11] showed that the estimation problem

* Corresponding author
Jesse Morris, Yiduo Wang, Mikolaj Kliniewski and Viorela Ila are with the Australian Centre For Robotics (ACFR), School of Aerospace, Mechanical and Mechatronic Engineering (AMME), University of Sydney, 2006 Sydney, Australia. {jesse.morris, yiduo.wang, mikolaj.kliniewski, viorela.ila}@sydney.edu.au

can be designed around estimating for the full $\mathrm{SE}(3)$ object motion. These motions are jointly optimized with the camera poses, and the static and dynamic structures, from which the per-object trajectory and velocities can be recovered. Their following work, VDO-SLAM [15], extended this formulation into a complete pipeline, but was never formally published. The works of Judd *et al*. [17], [22] similarly prioritize object motion, using this information to segment moving objects and track objects through occlusion. Comparatively, many systems directly estimate for object poses [23], [24], [25], typically using a constant-motion model fully constrain the estimation. However, downstream applications such as reconstruction [8], navigation [26] and prediction[27] often require information on both object pose and motion.

Our prior work [18] identified a further point of comparison between Dynamic SLAM formulations. Instead of only considering the estimation of either object pose or motion, this study termed and compared *object-centric* formulations, where dynamic points and motions are represented in an object's body frame, with *world-centric* formulations, which represent the variables in the world-frame. The insights from our work demonstrated that the choice of formulation plays a vital role in determining the behavior of convergence and the resulting estimation accuracy. Using the world-centric motion model [11], [15], [18] this study showed that explicit modeling of the object's rigid-body kinematics is essential for accurate results, especially when direct estimation of the object pose is required.

The inconsistencies in results and the variety of approaches, coupled with the communities growing interest in Dynamic SLAM motivated us to develop Dynamic Object Smoothing And Mapping, *DynoSAM*, a framework that allows for easy and modular prototyping, implementation and evaluation of Dynamic SLAM formulations. The suffix '-SAM' is inherited from the GTSAM [28] library which our framework is built upon, thereby allowing highly customizable estimation problems to be designed and implemented.

In this work, we present a detailed theoretical foundation for Dynamic SLAM based on a world-centric model for object motion. Furthermore, to support downstream tasks requiring object poses, we propose a novel formulation that explicitly uses the rigid-body kinematic model for accurate, direct pose estimation. To rigorously evaluate these formulations, we have integrated them into a full Dynamic vSLAM system; Fig. 1 presents an example output of our system estimating complex dynamic object motions. We assess the accuracy and performance of our system using a variety of indoor and outdoor datasets. To standardize and facilitate valid comparisons between systems, we additionally propose a new error metric for evaluating object motion. Compared with other dynamic SLAM pipelines [15], [17], [23], [24], we achieve state-of-the-art accuracy in the estimation of motion and pose of objects. We demonstrate that our camera pose estimation is robust in highly dynamic environments and compare with AirDOS [29], DynaVINS [20] and ORB-SLAM3 [4]. Finally, we demonstrate that our formulations can directly inform downstream navigation tasks by integrating our framework with dynamic object reconstruction [8] and trajectory prediction methods.

In summary, the contribution of this paper is threefold:

- **Theoretical Foundations for Object Motion Estimation.** We present a detailed theoretical formulation for representing and integrating observations of rigid bodies in motion within Dynamic SLAM. We offer a set of formulations that form the theoretical basis for practical system development, including a novel Dynamic SLAM formulation that directly parametrizes each object's pose in the scene and explicitly encodes rigid-body kinematics.
- **A Modular and Practical Dynamic vSLAM System.** We implement a complete Dynamic Visual SLAM system that integrates the various formulations discussed, supporting both batch and sliding-window optimizations. In addition, we provide a practical demonstration of our Dynamic SLAM pipeline with downstream tasks highly relevant for navigation, such as dynamic object reconstruction [8] and trajectory prediction. The framework is modular and extensible, allowing researchers to easily test and benchmark new formulations within a consistent evaluation setup.
- **Extensive Benchmarking and Novel Evaluation Metric.** We introduce an object motion evaluation metric that is agnostic to object reference frame definitions. Our framework is exhaustively evaluated on multiple public datasets, achieving state-of-the-art results. To our knowledge, this work provides one of the most comprehensive evaluations of Dynamic SLAM formulations to date.

As part of our contribution we open-source our C++ implementation of DynoSAM, which is integrated with the Robotic Operating System 2 (ROS2) [30]. We provide data logging and serialization tools for each module to facilitate evaluation and debugging, as well as an accompanying automated evaluation suite that has been used to generate all results in this paper. To facilitate modularity and parallelization of key components, DynoSAM's implementation is inspired by Kimera [31].

II. Related Work

This section provides an overview on how SLAM systems address the challenges of dynamic environments over the years. We focus on state-of-the-art Dynamic SLAM systems that estimate object poses and motions, but we also include those systems that aim to improve robustness by removing or ignoring dynamic objects.

A. SLAM in Dynamic Environments

In order to handle the dynamic entities in the environment, conventional SLAM systems such as ORB-SLAM 3 [4] and DynaVINS [20] detect any moving observations as outliers and reject them to create a global map that only contains static structures using methods such as RANSAC [32], point correlations [33] and robust bundle adjustment formulation [20], [21]. Deep learning methods have also been recently used to semantically understand the scene, and detect and remove dynamic objects [6], [34], [35]. By taking an active approach in removing dynamic objects, these methods provide robust and accurate camera pose estimation in dynamic environments. However, any relevant information about the motion of the

objects is discarded. Khronos [36] operate in dynamic environments by proposing a spatio-temporal SLAM formulation that unifies short-term changes, e.g. from dynamic objects, with long-term scene changes, and extracts moving objects from their representation using a geometric motion detection method [37]. However, they do not utilize any estimate of object motion and rely on a dense reconstruction pipeline to detect and track objects in the scene.

To improve the robustness of camera pose estimation in dynamic environments, many of these methods either implicitly or explicitly identify and segment out moving objects in the scene. While motion detection and segmentation is highly relevant for Dynamic SLAM, we highlight that our work is primarily interested in the estimation of object motions and poses, in addition to segmenting motions.

B. Dynamic SLAM

To represent the kinematic information of these objects, Dynamic SLAM methods incorporate measurements of dynamic entities in the SLAM formulation in addition to the static ones. These systems first segment dynamic observations from the static background using information such as kinematics [17], [24] and semantics [15], [23], [25], before solving for the pose or motion of these objects in addition to the camera/robot poses and the map of the environment either via joint optimization [18], [23] or separate estimation, such as frame-to-frame object motion tracking using sparse scene flow [38].

Several neural network-based methods have also been proposed in recent years [39], [40], [41], [42] to estimate the scene flow of 3D points or voxels. These systems focus on estimating the motions of individual points in the form scene flow vectors. While they produce impressive results, these systems do not recover the motions or poses of dynamic objects, and are therefore not further discussed in this paper.

1) Object Pose Representation: Within the Dynamic SLAM formulations that explicitly model the object, the most common and intuitive representation is to directly estimate the object's pose [23], [25], [43], [44]. Assuming each object is a rigid body, observed points on these objects are static in a body-fixed local object reference frame. By defining the poses of objects and in turn their local frames, there is an immediate advantage that each object point can be expressed in the object reference frame with a single variable in the optimization problem, reducing the overall number of variables in the system. Huang *et al*. [24], [25] cluster point observations on moving objects based on temporal rigidity to identify underlying rigid bodies. Their method, although able to extract dynamic objects from the scene, demonstrates poor object motion accuracy. Alternatively, several methods model objects using simple geometric primitives such as cuboids [45], [46], [47], [48] and ellipsoids [49] to define their poses. DynaQuadric [19] has proposed a joint optimization framework that models dynamic objects as quadrics, explicitly defining the per-object scale and demonstrating highly accurate camera localization. However, they evaluate only their object pose errors against other quadric-based methods and neglect object motion evaluations.

When focusing on specific object types, BodySLAM++ [50] and DSP-SLAM [51] leverage learned object shape prior models and achieve accurate object pose estimation. Their results demonstrate the benefit of integrating learned priors into a graph-based optimization framework, but BodySLAM++ is restricted to only estimating human poses and DSP-SLAM provides no understanding on object dynamics. Assuming strong knowledge on object motion, TwistSLAM [47] uses mechanical joint constraints to restrict the degrees of freedom for object motion estimation, and estimates accurate object poses under constrained motion. Their results however suggest limitations in estimating complex $\mathrm{SE}(3)$ object motions.

In comparison, other methods seek to estimate the full $\mathrm{SE}(3)$ motion of each object without the addition of semantic-specific constraints. Among these solutions, DynaSLAM II [23] takes an object-centric approach to pose and velocity estimation and reports the most accurate egomotion when compared with other approaches. By comparison, DynaSLAM II presents poor object motion estimation, and the authors consider their use of sparse features as the main reason behind such performance. However, our prior work [18] shows that the formulation used by Bescos *et al*. [23] does not explicitly model the rigid-body kinematics, resulting in poor estimation accuracy even with a dense object representation.

2) Object Motion Representation: An alternative formulation is to estimate object motions directly in a known reference frame, such as a camera frame that moves with a sliding window [17], [22], [52], or a well-defined reference frame like the world frame [15], [18], [29], which commonly coincides with the first camera/robot pose. MVO [17] represents the dynamic points in the camera frame, but models object motions in the object frame. They define object poses using point observations when each object is first observed, similar to the method of Bescos *et al*. [23].

Chirikjian *et al*. [53] showed that a $\mathrm{SE}(3)$ motion can be expressed in any reference frame, and based on this concept, VDO-SLAM [11], [15], [54] proposes a factor graph formulation to represent and estimate rigid-body object motions in the world frame without the need to estimate object poses. AirDOS [29] employs the same rigid-body motion model, extends it to articulated objects, e.g. humans, via a learning-based human pose estimator [55], and estimates the motion of each segment of an articulated object. They quantitatively demonstrate direct improvement in camera pose estimation due to the incorporation of dynamic objects. While the study of non-rigid body motion outside the scope of this paper, we find such a strategy inspiring for our future works. The motion models employed by [15], [29] are interrogated in [18] which showed, by comparing world and object-centric formulations, that a world-centric approach better encodes rigid-body kinematic models, leading to more accurate motion estimation, more efficient convergence behavior and often shorter computation time, despite requiring more variables.

A comprehensive review of the relevant literature highlights practical challenges, including the scarcity of open-source Dynamic SLAM libraries, systems, and datasets, which prevent rigorous comparison and evaluation. Inconsistent definitions of object reference frames further complicate pose and motion

assessment. These limitations motivated the development of DynoSAM, a flexible and comprehensive framework for Dynamic SLAM with frame-agnostic error metrics for evaluation.

III. Background

A. Rigid-Body Motion

Understanding rigid-body motion is fundamental to Dynamic SLAM as it provides a mathematical foundation to model the motion of objects in the environment. Our framework builds upon this understanding to estimate the motion of objects directly using frame-to-frame observations of 3D points on rigid bodies.

A rigid body, by definition, maintains its shape and size throughout motion; and therefore the distance between any pair of points on the object will not change. However, while these points must move together rigidly, the apparent motion of each point depends on the location of the observer. For instance, a point on a rotating wheel may appear stationary to an onboard observer but moving to one to a stationary observer external to the wheel. This example highlights the distinction between what we term as *local* motion, which is perceived by a reference frame that is rigidly attached to the object, and the *observed* motion which is perceived by an external observer.

In accordance with rigid-body kinematics, a reference *frame* is represented by a homogeneous transformation and can be rigidly attached to a rigid body, or can be placed anywhere in the environment. A *pose* of a frame defines the SE(3) homogeneous transform that contains the frame's position and orientation with respect to another reference frame, e.g. the observer frame.

To discuss these concepts more concretely, consider Fig. 2, which depicts a rigid body moving freely from time-step 1 to 2. Two arbitrary frames $\{A\}$ and $\{B\}$ are fixed to the object and observed by the fixed reference frame $\{O\}$. ${}^{O}\mathbf{A}_1, {}^{O}\mathbf{B}_1 \in \mathrm{SE}(3)$ denote the poses of frames $\{A_1\}$ and $\{B_1\}$ relative to $\{O\}$ prior to motion, while ${}^{O}\mathbf{A}_2, {}^{O}\mathbf{B}_2 \in \mathrm{SE}(3)$ denote the corresponding poses after motion, as shown by the solid black arrows in Fig. 2.

For any moving frame, we define its *motion* as a *change in pose* following Chirikjian *et al.* [53]. This motion is a SE(3) homogeneous transformation that converts one pose into another and thus describes the movement of a frame. Importantly, for a rigid body this motion can be represented in any arbitrary frame [53]. To express this information unambiguously, we adopt the three-index notation [53] for these motions which allows the observer frame to be explicitly identified in addition to the moving frame. For example, ${}^{A_1}_{A_1}\mathbf{H}_{A_2}$ and ${}^{B_1}_{B_1}\mathbf{H}_{B_2}$ are the local motions for frames $\{A\}$ and $\{B\}$ respectively, while the observed motion is ${}^{O}_{1}\mathbf{H}_2$. The remainder of the section details how each local motion is depends on its frame of origin, in contrast to the observed motion which is independent of the moving frame and can therefore describe the rigid-body motion of *any* point on the rigid-body.

1) Local Motion: Local motion describes the process of moving a reference frame from one pose relative to another *as perceived by a frame on the moving object*. Local motions are shown in Fig. 2 as green and blue arrows e.g. ${}^{A_1}_{A_1}\mathbf{H}_{A_2} \in \mathrm{SE}(3)$

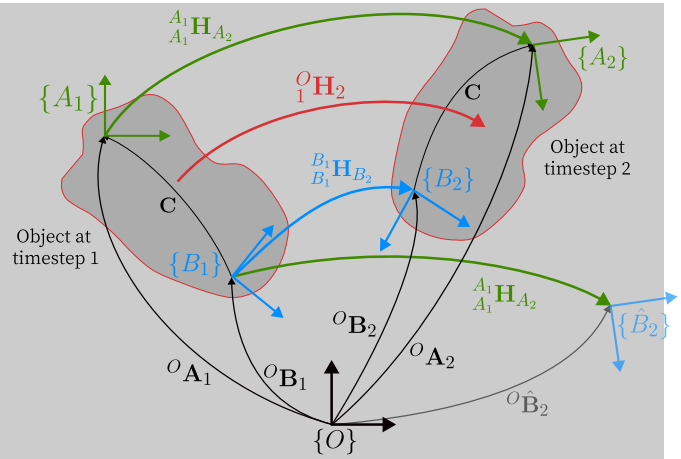


Fig. 2: Illustrative example of two arbitrary frames, $\{A\}$ and $\{B\}$, attached to a moving rigid body. Solid black arrows show the poses of each frame, while green and blue arrows show the corresponding local motions of each frame. The red arrow denotes the observed motion ${}^{O}_{1}\mathbf{H}_2$ which is a unique transform that describes the rigid-body motion of all points on the object. For this reason we draw it unattached to any specific reference frame. $\mathbf{C}$ is a constant transform that defines the rigid-body's kinematic constraint. Observe that both local and observed motions can be used to correctly transform the pose of each frame in accordance the kinematic constraints. However, local motion is dependent on the chosen reference frame and incorrect use leads to the invalid propagation of ${}^{O}\mathbf{B}_1$ to ${}^{O}\hat{\mathbf{B}}_2$

is the transformation of pose ${}^{O}\mathbf{A}_1$ to ${}^{O}\mathbf{A}_2$ from the perspective of frame $\{A_1\}$[1].

These local motions form a standard kinematic chain which allows poses of frame $\{A\}$ and $\{B\}$ to be propagated such that no rigid-body kinematic constraint are violated, i.e. $\mathbf{C}$ remains constant:

$$\begin{aligned} {}^{O}\mathbf{A}_2 &= {}^{O}\mathbf{A}_1 \; {}^{A_1}_{A_1}\mathbf{H}_{A_2} \\ {}^{O}\mathbf{B}_2 &= {}^{O}\mathbf{B}_1 \; {}^{B_1}_{B_1}\mathbf{H}_{B_2}. \end{aligned} \tag{1}$$

Vitally, each local motion is dependent on the frame's origin and can therefore break the rigid-body constraint if applied to a pose with a different origin; this specific case is illustrated in Fig. 2 where ${}^{A_1}_{A_1}\mathbf{H}_{A_2}$ is applied to ${}^{O}\mathbf{B}_1$. The resulting pose ${}^{O}\hat{\mathbf{B}}_2$ violates rigid-body constraints, as the relative transformation from ${}^{O}\hat{\mathbf{B}}_2$ to ${}^{O}\mathbf{A}_2$ is clearly not equal to $\mathbf{C}$.

2) Observed Motion: Observed motion describes the motion of a rigid body as seen by an external observer. In Fig. 2, the observed motion from time-steps 1 to 2 is defined with respect to $\{O\}$ and is therefore written ${}^{O}_{1}\mathbf{H}_2$. Unlike local motion in (1), this motion is invariant to the choice of body-fixed reference frame and can there be applied to any pose associated with a fixed reference on the object, whilst upholding rigid-body constraints: ${}^{O}_{1}\mathbf{H}_2 = {}^{O}_{A_1}\mathbf{H}_{A_2} = {}^{O}_{B_1}\mathbf{H}_{B_2}$. The observed motion is shown with a red arrow that, importantly, is not attached to any specific reference frame. This highlights that, unlike local motions, the observed motion does not have a geometric interpretation yet exists as a unique transform that can propagate the poses ${}^{O}\mathbf{A}_1$ and ${}^{O}\mathbf{B}_1$ to ${}^{O}\mathbf{A}_2$ and ${}^{O}\mathbf{B}_2$, as well as any other point that is rigidly attached to the object.

[1]It is worth noting that usually in robotics ${}^{A}_{A}\mathbf{H}_B$ is considered, which allows for the simpler notation ${}^{A}\mathbf{H}_B$. In this case, ${}^{A}\mathbf{H}_B$ is the coordinate transform from A to B and its matrix is the same as that of the motion ${}^{A}_{A}\mathbf{H}_B$.

While these underlying principles have been previously established, notably in [53] and subsequently applied to the estimation problems in [11], [15], [18], a full derivation of observed motion within the context of Dynamic SLAM has not been previously presented. We believe a more complete derivation is crucial for understanding the benefits of this representation and in this section we show how to derive ${}^{O}_{1}\mathbf{H}_2$ given the relations defined in (1) and by exploiting the rigid-body constraint $\mathbf{C}$ between $\{A\}$ and $\{B\}$. Each local motion can be represented with a single observed motion when expressed in a common origin frame:

$$
\begin{aligned}
{}^{O}_{1}\mathbf{H}_2 &:= {}^{O}\mathbf{A_1}\ {}^{A_1}_{A_1}\mathbf{H}_{A_2}\ {}^{O}\mathbf{A_1}^{-1} && (2.0)\\
&= {}^{O}\mathbf{B_1}\ {}^{B_1}_{B_1}\mathbf{C}_{A_1}\ {}^{A_1}_{A_1}\mathbf{H}_{A_2}\ {}^{B_1}_{B_1}\mathbf{C}^{-1}_{A_1}\ {}^{O}\mathbf{B_1}^{-1} && (2.1)\\
&= {}^{O}\mathbf{B_1}\ {}^{B_1}_{B_1}\mathbf{C}_{A_1}\ {}^{A_1}_{A_1}\mathbf{H}_{A_2}\ {}^{B_2}_{B_2}\mathbf{C}^{-1}_{A_2}\ {}^{O}\mathbf{B_1}^{-1} && (2.2)\\
&= {}^{O}\mathbf{B_1}\ {}^{B_1}_{B_1}\mathbf{H}_{B_2}\ {}^{O}\mathbf{B_1}^{-1}, && (2.3)
\end{aligned}
$$

where (2.0) is the definition of a frame change of a pose transformation [53], and because ${}^{B}\mathbf{C}_A = {}^{O}\mathbf{B}^{-1}{}^{O}\mathbf{A}$, we arrive at (2.1). Given that frames $\{A\}$ and $\{B\}$ belong to the same rigid body, their relative transformation ${}^{B}\mathbf{C}_A$, is invariant to time, i.e. ${}^{B_1}\mathbf{C}_{A_1} = {}^{B_2}\mathbf{C}_{A_2}$, hence we can reach (2.2), which can be re-written as (2.3) following the kinematic chain visualized in Fig. 2.

Equation (2) shows that ${}^{O}_{1}\mathbf{H}_2$ remains constant regardless of the choice of $\{A\}$ and $\{B\}$ and, therefore, this representation of motion is independent of the rigid body's frame of reference. Furthermore, using (1), we can rewrite (2) as:

$$
{}^{O}_{1}\mathbf{H}_2 = {}^{O}\mathbf{A_2}\ {}^{O}\mathbf{A_1}^{-1} = {}^{O}\mathbf{B_2}\ {}^{O}\mathbf{B_1}^{-1}, \tag{3}
$$

allowing new composition relationships to be established:

$$
\begin{aligned}
{}^{O}\mathbf{A_2} &= {}^{O}_{1}\mathbf{H}_2\ {}^{O}\mathbf{A_1}\\
{}^{O}\mathbf{B_2} &= {}^{O}_{1}\mathbf{H}_2\ {}^{O}\mathbf{B_1}.
\end{aligned} \tag{4}
$$

We highlight that the composition order of the observed motion in (4) is not the same as that for the local motion in (1) due to their fundamental differences, and we direct the interested reader to [53] for more details.

In a dynamic scene with many objects and significant occlusions, the distinction between local and observed motion play a crucial role. The camera can only observe the motions of a small set of points on each object in a busy scene and considering only local motion makes it difficult to infer the object's overall motion as each point experiences a unique motion. However, if instead we expressing these points in a common external reference frame, *the apparent motion of all points on the object can be observed directly and described using a single homogeneous transformation*, i.e. the observed motion [53], which is the foundation for our formulation of the Dynamic SLAM problem.

B. Notations

We define the notations for the Dynamic SLAM problem by considering a dynamic scene comprised of camera poses $\mathcal{X}$ and object poses $\mathcal{L}$:

$$
\mathcal{X} = \{{}^{W}\mathbf{X}_k \in \mathrm{SE}(3)\}_{k\in\mathcal{K}}, \quad \mathcal{L} = \{{}^{W}\mathbf{L}^j_k \in \mathrm{SE}(3)\}^{j\in\mathcal{J}_k}_{k\in\mathcal{K}_j}
$$

where $\{W\}$ is the fixed world frame, $\mathcal{K}$ is the set of all time-steps and $\mathcal{J}$ is the set of all object indices. Throughout this work we liberally use the super-script j or sub-script k to specify the subset denoting to a particular object or at a specific time-step, e.g. $\mathcal{J}_k \in \mathcal{J}$ referring to all the observed objects at time-step k. The camera ${}^{W}\mathbf{X}_k$ and object ${}^{W}\mathbf{L}_k$ poses define the location of frames $\{X_k\}$ and $\{L_k\}$ with respect to $\{W\}$.

We denote $\mathbf{m}^i = \left[\tilde{\mathbf{m}}^i, 1\right]^\top$ as the homogeneous coordinates of a 3D point $\tilde{\mathbf{m}}^i \in \mathbb{R}^3$, where i is a unique tracklet index and indicates correspondences across frames. Any point in the world frame can be transformed into the camera frame:

$$
{}^{X_k}\mathbf{m}^i_k = {}^{W}\mathbf{X}^{-1}_k\ {}^{W}\mathbf{m}^i_k.
$$

We use $\mathbf{m} \in \mathcal{M}$ to refer to both static and dynamic points. In the case of a static point, ${}^{W}\mathbf{m}_k \doteq {}^{W}\mathbf{m}$ as we omit the time-step k to indicate the variable is time invariant. To minimize notation clutter, we liberally omit indices i and j when there is no ambiguity.

We denote motion of the camera as $\mathbf{T}$ and the motion of any object j as $\mathbf{H}$. From now onwards we use $\mathbf{H}$ to only refer to the motion of objects, rather denoting general motion as in Sec. III-A. Between time-steps $k-1$ and k the local motions of the camera and object are ${}^{X_k}_{X_k\ 1}\mathbf{T}_{X_k}$ and ${}^{L_k}_{L_k\ 1}\mathbf{H}_{L_k}$ respectively. In the case where our notation is already explicit in denoting which pose the transformation is referring to, i.e., camera pose with $\mathbf{T}$ or object pose with $\mathbf{H}$, we simplify the notation by only using time-steps:

$$
{}^{X_k}_{k-1}{}^{1}\mathbf{T}_k \doteq {}^{X_k}_{X_k\ 1}\mathbf{T}_{X_k}, \quad {}^{L_k}_{k-1}{}^{1}\mathbf{H}_k \doteq {}^{L_k}_{L_k\ 1}\mathbf{H}_{L_k}
$$

These transformations describe local motions and, following (1), can be used to propagate the camera and object poses from $k-1$ to k:

$$
{}^{W}\mathbf{X}_k = {}^{W}\mathbf{X}_{k-1}\ {}^{X_k}_{k-1}{}^{1}\mathbf{T}_k \tag{5}
$$

$$
{}^{W}\mathbf{L}_k = {}^{W}\mathbf{L}_{k-1}\ {}^{L_k}_{k-1}{}^{1}\mathbf{H}_k \tag{6}
$$

For the purposes of object motion estimation, our framework uses an observed motion representation, as discussed in Sec. III-A2. In the context of SLAM this reference frame can be any observable frame, e.g. the camera frame at the start of a sliding-window. In this work, we represent our object motions in the world frame $\{W\}$, following the works of [11], [15], [18]. Such a representation is convenient for downstream applications like mapping and planning, as it allows all state variables in our formulations to be defined with respect to a common coordinate system. For any object j this motion is denoted as ${}^{W}_{k-1}\mathbf{H}_k$. Due to its representation in the world frame we refer specifically to it as the *absolute* motion. We define the set of all object motions as:

$$
{}^{W}_{k-1}\mathcal{H}_k = \{{}^{W}_{k-1}\mathbf{H}^j_k\}^{j\in\mathcal{J}_k}_{k\in\mathcal{K}_j}
$$

Using (3) we define our absolute motion in terms of a change in object pose:

$$
{}^{W}_{k-1}\mathbf{H}_k = {}^{W}\mathbf{L}_k\ {}^{W}\mathbf{L}^{-1}_{k-1} \tag{7}
$$

Following the composition rules defined in (4), ${}^{W}_{k-1}\mathbf{H}_k$ will

move ${}^{W}\mathbf{L}_{k-1}$ to ${}^{W}\mathbf{L}_{k}$ when applied on the left-hand side:

$$ {}^{W}\mathbf{L}_{k} = {}^{W}_{k-1}\mathbf{H}_{k} \; {}^{W}\mathbf{L}_{k-1}. \tag{8} $$

Vitally, this propagation holds for any point i on a rigid-body:

$$ {}^{W}\mathbf{m}^{i}_{k} = {}^{W}_{k-1}\mathbf{H}_{k} \; {}^{W}\mathbf{m}^{i}_{k-1}. \tag{9} $$

Thus, the transform ${}^{W}_{k-1}\mathbf{H}_{k}$ describes the motion of *any and all* points on an object from $k-1$ to k. Since we can directly observe 3D object points, (9) allows us to easily derive cost functions in terms of all tracked points on the object. This equation underpins all our formulations and is fundamental to our Dynamic SLAM framework. A more detailed derivation of (9) is included in Appendix A which highlights how this equation implicitly encodes the rigid-body kinematic constraints discussed in Sec. III-A.

IV. Foundations for Dynamic SLAM

This section details the graph-based Dynamic SLAM formulations included within our framework. We demonstrate how the concept of observed motion, explained in Sec. III, is applied to directly estimate both object motions (Sec. IV-B) and poses (Sec. IV-C). These formulations constitute the back-end of a visual Dynamic SLAM pipeline. As a result, we assume dynamic objects can be tracked and 3D point measurements of static and dynamic features can be provided by a visual SLAM front-end. While we outline our front-end (Sec. V-B) and include it in our open-source implementation, the formulations discussed in our framework are agnostic to any specific front-end implementation.

For each approach, we define residual functions $\mathbf{r}$ associated with covariance matrices Σ. These are used to construct factors ϕ in the form:

$$ \phi(\cdot) \propto \exp\left\{-\frac{1}{2}\|\mathbf{r}\|^2_{\Sigma}\right\} \tag{10} $$

Under the standard assumption of zero-mean Gaussian noise, we take the negative log of (10) and drop the scaling-factor allowing us to collect all residuals and construct a nonlinear least-squares problem:

$$ \theta^{\text{MAP}} = \underset{\theta}{\operatorname{argmin}} \sum \|\mathbf{r}\|^2_{\Sigma} \tag{11} $$

For completeness, our framework also includes the object-centric formulations explored in [18]. However, due to their inferior performance, this work does not discuss them further.

A. Measurements

We define the following notation for visual features. Direct measurements are denoted as $\mathbf{z}_{2\text{D}} \in \mathbb{R}^2$ and represent 2D pixel measurements on the image plane. We denote 3D point measurements in the local sensor frame $\{X\}$ as $\mathbf{z}_{3\text{D}} \in \mathbb{R}^3$. In the context of stereo/RGBD SLAM, we assume the depth d of each pixel measurement $\mathbf{z}_{2\text{D}}$ is available for every time-step k, and 3D measurements are constructed accordingly:

$$ [\mathbf{z}_{3\text{D}}, 1]^{\top} = {}^{X_k}\mathbf{m}_k = \pi^{-1}(\mathbf{z}_{2\text{D}}, d). \tag{12} $$

where $\pi^{-1}(\cdot)$ is a sensor's back-projection function, e.g. pinhole camera model. We further specify the notation for sets

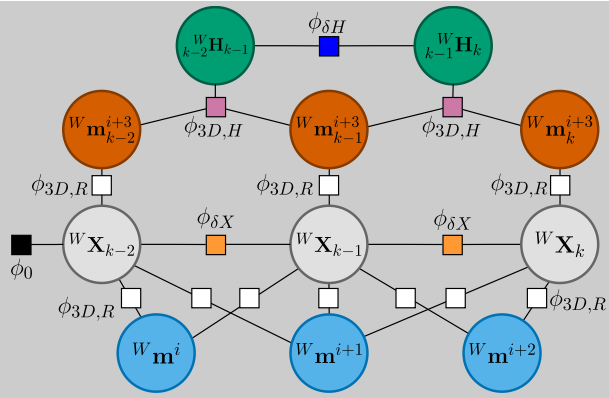


Fig. 3: World-centric motion formulation factor graph. We show three static landmarks (blue) and one landmark on a dynamic object (orange), tracked over three frames. Camera poses are in grey and object motion in green. The *point measurement factor* ($\phi_{3D,R}$) is shown in white, the *ternary motion factor* ($\phi_{3D,H}$) in magenta and the *object motion smoothing factor* ($\phi_{\delta H}$) in dark blue. The prior factor on the first camera pose is shown in black.

of measurements corresponding to static and dynamic entities. $\mathcal{S}_{2\text{D},k}$ and $\mathcal{S}_{3\text{D},k}$ denote the set of static 2D keypoints $\mathbf{z}_{2\text{D}}$ and their corresponding 3D landmark measurements $\mathbf{z}_{3\text{D}}$ at time k, respectively. Similarly, $\mathcal{D}_{2\text{D},k}$ and $\mathcal{D}_{3\text{D},k}$ are the sets of dynamic measurements. The set of all static measurements (2D and 3D) is $\mathcal{S}_k$. The set of dynamic measurements on a particular object is specified using the super-script j, e.g. $\mathcal{D}^{j}_{3\text{D},k}$ is the sub-set of 3D measurements on object j at time k.

B. World-Centric Motion Estimator

The formulation for the World-Centric Motion Estimator (WCME) directly estimates object motions and their rigid-structure, as well as the static scene and camera pose:

$$ \theta \doteq [{}^{W}\mathbf{X}_k, \; {}^{W}_{k-1}\mathcal{H}_k, \; {}^{W}\mathcal{M}_k], \; k \in \mathcal{K}. \tag{13} $$

This formulation was first proposed in [15] and evaluated in [18] which established it as a highly accurate method for object motion estimation. Fig. 3 shows the corresponding factor graph for this formulation and includes three static landmarks (blue) and one landmark (orange) on a dynamic object that are tracked over three frames.

Given a 3D observation of a point (static or dynamic), we constrain the point to the world frame using a *point measurement factor*:

$$ \mathbf{r}_{3\text{D},R} = \mathbf{z}_{3\text{D},k} - {}^{W}\mathbf{X}_k^{-1} \; {}^{W}\mathbf{m}_k. \tag{14} $$

For every static point, ${}^{W}\mathbf{m}$ is initialized as ${}^{W}\mathbf{m} = {}^{W}\mathbf{X}_k \mathbf{z}_{3\text{D},k}$ when the point is first observed. We use the same method to initialize each dynamic point at every frame k.

The relative transformation between consecutive camera poses is modeled using the *between factor*:

$$ \mathbf{r}_{\delta X} = \left[\log\left({}^{X_k}_{k-1}\mathbf{T}_k^{-1} \; {}^{W}\mathbf{X}_{k-1}^{-1} \; {}^{W}\mathbf{X}_k\right)\right]^{\vee}, \tag{15} $$

where the odometry ${}^{X_k}_{k-1}\mathbf{T}_k$ can be estimated by the front-end of a visual SLAM system, such as the proposed one described in Sec. V-B, or from alternative sources such as an Inertial Measurement Unit (IMU) if available. The operation $[\log(\cdot)]^{\vee}$ maps an $\mathrm{SE}(3)$ transformation to $\mathbb{R}^6$.

From (9) the *ternary object motion factor* models the motion of any point i on a rigid body j as in [15], [18]:

$$\mathbf{r}_{3\mathrm{D},H} = {}^{W}\mathbf{m}_k^i - {}^{W}_{k-1}\mathbf{H}_k \; {}^{W}\mathbf{m}_{k-1}^i, \tag{16}$$

This cost function relates a pair of tracked points on a rigid-body object with the object motion.

Finally, for each object j an *object smoothing factor* is introduced between consecutive motions:

$$\mathbf{r}_{\delta H} = \left[\log\left({}^{W}_{k-2}\mathbf{H}_{k-1}^{-1} \; {}^{W}_{k-1}\mathbf{H}_k\right)\right]^{\vee}. \tag{17}$$

This enforces a constant motion model represented in $\{W\}$ and prevents abrupt, drastic and unrealistic changes in object motions between consecutive frames. This constraint holds since any constant change in the body frame also leads to a constant change in the reference frame pose change, as shown in [11]; an extended version of this proof is included in Appendix B.

The nonlinear least-squares problem is constructed using the combination of these factors:

$$\begin{aligned}\theta^{\mathrm{MAP}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\Bigg(&\|\mathbf{r}_0\|^2_{\Sigma_0} + \|\mathbf{r}_{\delta X}\|^2_{\Sigma_{\delta X}} \\ &+ \sum_{m\in\mathcal{S}_{3\mathrm{D},k}} \rho_h\|\mathbf{r}_{3\mathrm{D},R}\|^2_{\Sigma_{3\mathrm{D},R}} + \sum_{m\in\mathcal{D}_{3\mathrm{D},k}} \rho_h\|\mathbf{r}_{3\mathrm{D},R}\|^2_{\Sigma_{3\mathrm{D},R}} \\ &+ \sum_{j\in\mathcal{J}_k}\sum_{m\in\mathcal{D}^j_{3\mathrm{D},k}} \rho_h\|\mathbf{r}_{3\mathrm{D},H}\|^2_{\Sigma_{3\mathrm{D},H}} + \sum_{j\in\mathcal{J}_k}\|\mathbf{r}_{\delta H}\|^2_{\Sigma_{\delta H}}\Bigg),\end{aligned} \tag{18}$$

where $\mathbf{r}_0$ is prior residual on the first camera pose state variable and ρ_h is any robust cost function. In our implementation, we use the Huber [56] function. While not being directly incorporated in this estimation, the trajectory of each object ${}^{W}\mathcal{L}$ can be recovered by recursively propagating the previous pose using the estimated motion following (8):

$${}^{W}\mathbf{L}_k = {}^{W}_{k-1}\mathbf{H}_k \ldots {}^{W}_{s}\mathbf{H}_{s+1} \; {}^{W}\mathbf{L}_s, \tag{19}$$

where s is the first time-step at which this object is observed. This method requires an arbitrary first pose ${}^{W}\mathbf{L}_s$ to be defined, which anchors the resulting trajectory. Since the estimated motion is an absolute motion, we can define this pose anywhere relative to $\{W\}$. Practically, we construct ${}^{W}\mathbf{L}_s$ upon the initial observation of an object using the centroid of all object point observations as its position and identity as its orientation. In addition, any method that provides object pose, such as learned techniques, can easily be adopted. For experiment evaluation, we use the ground truth object pose to set ${}^{W}\mathbf{L}_s$, aligning the origins of estimated object trajectories and that of the ground truth to facilitates proper evaluation as discussed in Sec. VI-B.

C. World-Centric Pose Estimator

Up to this point, we have used the rigid-body motion ${}^{W}\mathbf{H}$ to directly parametrize the object state. This perspective, used in both our work and prior studies, has enabled effective estimation of object motion and recovery of object pose. However, while ${}^{W}_{k-1}\mathbf{H}_k$ provides valuable insight regarding object dynamics, directly estimating the object pose ${}^{W}\mathbf{L}_k$ is highly relevant for downstream tasks. We therefore propose the World-Centric Pose Estimator (WCPE), a novel formulation

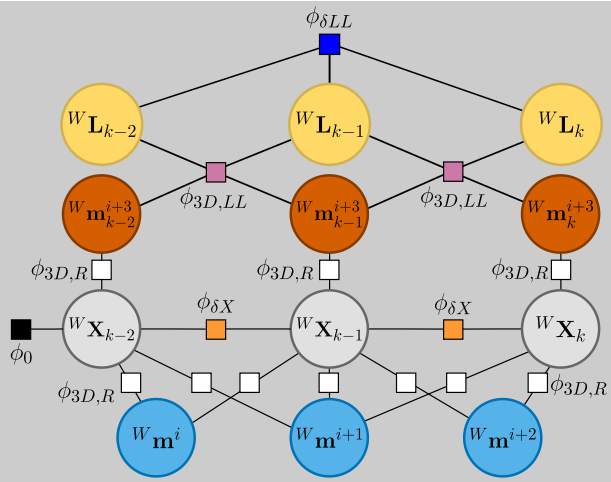


Fig. 4: World-centric pose formulation factor graph. As with the previous formulation (shown in Fig. 3), three static landmarks (blue) and one dynamic landmark (orange) are shown. The object pose, $\mathbf{L}$, is shown in yellow with the modified motion ($\phi_{3D,LL}$) and smoothing factor ($\phi_{\delta LL}$) in magenta and dark blue respectively.

that parametrizes the object state in terms of its pose ${}^{W}\mathbf{L}$. Unlike other estimators that use a motion prior [17], [23] between poses, our formulation uses the rigid-body motion directly as a constraint between poses, based on the measurement of object points. This approach therefore demonstrates the importance of using the absolute motion representation as the foundation for new Dynamic SLAM formulations.

WCPE estimates the states:

$$\theta \doteq [{}^{W}\mathbf{X}_k, {}^{W}\mathcal{L}_k, {}^{W}\mathcal{M}_k] \; k \in \mathcal{K} \tag{20}$$

The core of this approach is as follows: equation (7) shows how a motion ${}^{W}_{k-1}\mathbf{H}_k$ explicitly models a pair of consecutive object poses. Using this idea, we reformulate our factors to make every ${}^{W}\mathbf{L}$ a variable of the system. The corresponding factor graph is shown in Fig. 4. Substituting (7) into (16) forms a new *quaternary object motion factor*:

$$\mathbf{r}_{3\mathrm{D},LL} = {}^{W}\mathbf{m}_k^i - {}^{W}\mathbf{L}_k \; {}^{W}\mathbf{L}_{k-1}^{-1} \; {}^{W}\mathbf{m}_{k-1}^i. \tag{21}$$

We similarly define the smoothing factor in-terms of object pose by substituting (7) into (17). This forms the *object pose smoothing factor*:

$$\mathbf{r}_{\delta LL} = \left[\log\left(\left({}^{W}\mathbf{L}_{k-1} \; {}^{W}\mathbf{L}_{k-2}^{-1}\right)^{-1}\left({}^{W}\mathbf{L}_k \; {}^{W}\mathbf{L}_{k-1}^{-1}\right)\right)\right]^{\vee} \tag{22}$$

The nonlinear least-squared problem is constructed as:

$$\begin{aligned}\theta^{\mathrm{MAP}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\Bigg(&\|\mathbf{r}_0\|^2_{\Sigma_0} + \|\mathbf{r}_{\delta X}\|^2_{\Sigma_{\delta X}} \\ &+ \sum_{s\in\mathcal{S}_{3\mathrm{D},k}} \rho_h\|\mathbf{r}_{3\mathrm{D},R}\|^2_{\Sigma_{3\mathrm{D},R}} + \sum_{d\in\mathcal{D}_{3\mathrm{D},k}} \rho_h\|\mathbf{r}_{3\mathrm{D},R}\|^2_{\Sigma_{3\mathrm{D},R}} \\ &+ \sum_{j\in\mathcal{J}_k}\sum_{m\in\mathcal{D}^j_{3\mathrm{D},k}} \rho_h\|\mathbf{r}_{3\mathrm{D},LL}\|^2_{\Sigma_{3\mathrm{D},LL}} + \sum_{j\in\mathcal{J}_k}\|\mathbf{r}_{\delta LL}\|^2_{\Sigma_{\delta LL}}\Bigg)\end{aligned} \tag{23}$$

We highlight that our formulation represents both object motions and their points in the world frame. This differs from existing methods that also estimate for object pose but

use an object-centric representation for points [23], [18]. This small difference is vital to ensure the rigid-body motion model remains explicit and therefore the potential problems of object-centric formulations outlined by Morris *et al*. [18] are avoided. Furthermore, by continuing to base our formulation on the absolute motion representation, we can define $\{L\}$ anywhere on the object, as the absolute motion representation is independent of the object's reference frame and pose (Sec. III-A). This flexibility enables the integration of various initialization methods, e.g. SLAM front-ends or observation centroids, without compromising rigid-body kinematic constraints.

V. System

Our work is primarily focused on the estimation of object motions and poses and it is highly relevant to provide an open-source Dynamic SLAM system to facilitate evaluation and wide-spread use. Based on the formulations discussed in Sec. IV, this section presents the DynoSAM pipeline, a factor-graph-based Dynamic Visual SLAM system.

Our pipeline is broken into the typical frontend-backend structure. As shown in Fig. 5, our pipeline takes stereo/RGBD, instance segmentation masks and dense optical flow as input, and produces globally consistent trajectories of the camera and objects as well as the static structure and temporal map of each object (Fig. 5 (a-c)). The front-end (Sec. V-B) processes this input data to generate static and dynamic feature tracks and provides initial estimates for the back-end. It is responsible for object-level data association across frames, ensuring that features are consistently tracked on the same object to enable robust estimation. The robust handling of noisy depth and optical flow data, as well as segmentation inaccuracies, have been well studied in *et al*. [11], [54], which serve as inspiration for our front-end implementation. Further evaluations, while relevant for real-world deployment, is beyond the scope of this particular work, since our primary focus is the understanding and evaluation of the Dynamic SLAM back-end. The back-end (Sec. V-C) implements our presented formulations in Sec. IV and defines the Dynamic SLAM estimation problem. It fuses static and dynamic measurements via a GTSAM-based [28] factor-graph optimization to produce a globally consistent estimate of the dynamic scene. Our implementation is flexible to accommodate any custom formulation for Dynamic SLAM and options for full-batch smoothing where the system is optimized over all measurements, as well as a sliding-window estimation which bounds the size of the optimization problem are available. We emphasize that while our primary contributions encompass the back-end of a Dynamic SLAM pipeline, we highlight that our open-source front-end also includes implementation improvements, such as multi-threaded object tracking for improved efficiency, and a range of user options.

We implement both the world-centric motion and pose formulations explained in Sec. IV as part of our system, and exhaustively test and analyze them on a wide variety of datasets, the result of which is presented in Sec. VI.

A. Image Preprocessing

The front-end requires a set of four images $\mathcal{I}_k$ as input per time-step. Each $\mathcal{I}_k$ consists of a RGB image I_k^R , an instance segmentation I_k^S , per-pixel optical-flow I_k^F , and per-pixel depth I_k^D , as shown in Fig. 5. We undistort and align I_k^R with the depth image I_k^D that is obtained from an RGBD or stereo camera system. Both depth and optical-flow are expected to be dense and can be obtained from classical [57] or learned methods [58], [59]. Finally, I_k^S is used in the process of masking out dynamic objects from the static background. Our system requires all background pixels to be labeled as 0 and object pixels in the scene are labeled $1 \dots n_o$. This processing is performed online as an integral part of our complete SLAM pipeline; currently RAFT [58] and YOLOv8 [60] are used to compute dense optical-flow and semantic instance segmentation respectively.

B. Front-end

The DynoSAM front-end performs per-frame visual tracking, producing an initial estimation of camera pose ${}^{W}\mathbf{X}_k$ and per-object motions ${}^{W}_{k-1}\mathcal{H}_k$ as well as a set of visual measurements as defined in Sec. IV-A:

$$\mathcal{Z}_k = \{\mathcal{S}_k, \mathcal{D}_k\}.$$

The reminder of this section introduces each front-end module. The Feature Detection & Tracking module (Sec. V-B1) generates $\mathcal{Z}_k$ by tracking static and dynamic features from $\mathcal{I}_k$. These measurements support camera pose estimation (Sec. V-B2) and multi-object motion estimation (Sec. V-B3). Joint Optical-Flow (Sec. V-B4) and Object Motion Refinement (Sec. V-B5) components are used to further refine camera pose and object motion estimates.

1) Feature Detection & Tracking: The feature detection and tracking module matches corresponding static and per-object features between consecutive frames. I_k^R and I_k^F are used for feature detection and tracking while I_k^S is used to determine whether a keypoint belongs to the static background or a dynamic object. This module ensures consistent tracking of features on the same object while maintaining sufficient spread and density of features across frames to preserve the accuracy of both visual odometry and object motion estimation.

After detection and tracking, all features are initially marked as inliers and may be marked as outliers during the motion estimation module as visualized in Fig. 5. Once both static and dynamic keypoints are tracked, we use the input depth map I_k^D to directly obtain 3D measurements $\mathcal{D}_{3\mathrm{D},k}$ and $\mathcal{S}_{3\mathrm{D},k}$ from 2D pixel measurements, forming the local map shown in Fig. 5. We provide a detailed explanation below.

Static Feature Tracking. At each time-step, a *sparse* set of detected static keypoints $\mathcal{S}_{2\mathrm{D},k}$ are tracked across consecutive frames using optical flow. Our framework offers options to select different feature extraction algorithms depending on the use case; for our experiments we used Shi-Tomasi corners [61] for indoor environments and ORB [62] features for outdoor environments. By default, our front-end uses the Lucas-Kanade tracker [63] to generate feature correspondences. The input optical flow I_k^F can also be used when it is reliable. Once tracked, our front-end performs geometric verification to discard poor correspondences.

To improve robust tracking, we retain only relevant features by applying Adaptive Non-Maximal Suppression

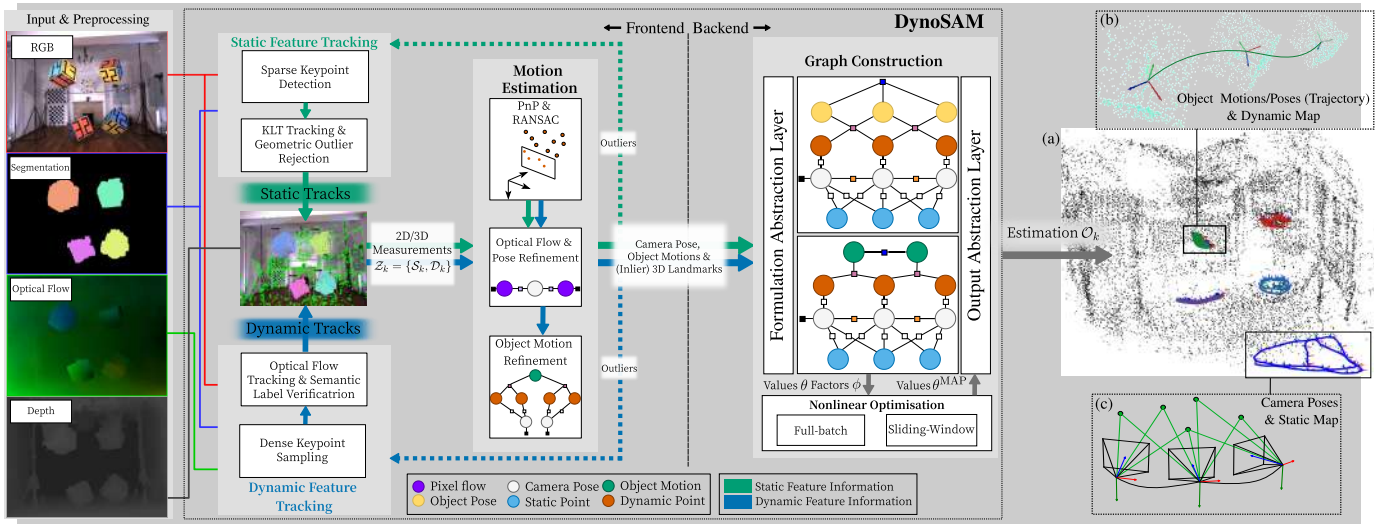


Fig. 5: DynoSAM system diagram. Our pipeline receives RGB, Segmentation, Optical Flow and Dense Depth images (on the left) as input and outputs static/dynamic map points, object and robot trajectories, shown in **(a)**. A complete front-end is included for feature and object tracking as well as a back-end where different Dynamic SLAM formulations are implemented. The per-object map and trajectory is highlighted in **(b)** and the camera trajectory in **(c)**.

(ANMS) [64] to the detected keypoints. This procedure retains only the most informative and reliable features, while simultaneously promoting a spatially uniform distribution of features across the image. ANMS achieves this by culling uninformative features whilst ensuring that a minimum number of features remain (800 in our experiments). Our front-end will detect new keypoints on the static structure if the number of inlier tracks falls below this minimum threshold. This adaptive mechanism ensures that a consistently high number of well-distributed features are tracked across successive frames, thereby enhancing tracking robustness and accuracy.

Dynamic Feature Tracking. We track a *dense* set of features $\mathcal{D}_{2D,k}$ for each object. Dense tracking is important to achieve good coverage over the entire observable object for robust motion estimation [54]. For each dynamic measurement we maintain an associated object-level label j, such that features can be associated to the same object between frames.

The instance segmentation mask I_k^S is used to retrieve the object label j per pixel. While this mask can be generated from any standard instance segmentation network, many of these networks do not guarantee that the instance labels will be temporally consistent. Before tracking dynamic keypoints, object masks are tracked using [65]. We then remap the original instance labels to a temporally consistent label j per object. If the user has their own instance segmentation and tracking method, they can choose to bypass this algorithm. In that case, the user only needs to ensure that each pixel represents the unique tracking label j.

Although the instance segmentation mask I_k^S is able to separate objects from the background, it cannot distinguish between static and moving objects. To address this, we use the method of Zhang *et al.* [54] to identify moving objects based on scene flow, allowing us to focus on tracking features only on the dynamic objects. This module can be easily modified to integrate learning-based methods to identify moving objects.

Once the moving objects in the image are identified and tracked, dense features are extracted by sampling keypoints uniformly within each tracked object mask. Optical flow I_k^F is then used directly to find the correspondences between frames. Our tracking algorithm ensures that, where possible, a consistent and significant number of features (800 in our experiments) on each object are tracked. Similarly to static points, only inliers from the previous frame are used.

2) Initial Camera Pose Estimation: At each time-step, static measurements $\mathcal{S}_k$ are used to estimate the initial camera pose. As shown in the motion estimation module in Fig. 5, the first step is the PnP algorithm [66] which estimates ${}^{W}\mathbf{X}_k$ by minimizing the re-projection error:

$$\mathbf{r}_{2D,R} = \mathbf{z}^i_{2D,k} - \pi({}^{W}\mathbf{X}_k^{-1}\ {}^{W}\mathbf{m}^i) \tag{24}$$

between the tracked static keypoints in the current frame at time-step k and the local map constructed from the previous frame. RANSAC verification and outlier rejection is performed using the implementation as provided by OpenGV [67] to obtain robust initial estimation of the camera pose. This initial camera pose is further refined through joint optical flow and pose optimization method detailed in Sec. V-B4.

3) Initial Object Motion Estimation: The initial object motion ${}^{W}_{k-1}\mathbf{H}_k \in \mathcal{H}_k$ is estimated following the method of Zhang *et al.* [15]:

$$\mathbf{r} = \mathbf{z}^i_{2D,k} - \pi({}^{W}\mathbf{X}_k^{-1}\ {}^{W}_{k-1}\mathbf{H}_k\ {}^{W}\mathbf{m}^i_{k-1}) \tag{25}$$

$$= \mathbf{z}^i_{2D,k} - \pi({}^{W}_{k-1}\mathbf{G}_k\ {}^{W}\mathbf{m}^i_{k-1}), \tag{26}$$

where ${}^{W}_{k-1}\mathbf{G}_k = {}^{W}\mathbf{X}_k^{-1}\ {}^{W}_{k-1}\mathbf{H}_k$. Since (26) is in the same form as (24), we can solve for ${}^{W}_{k-1}\mathbf{G}_k$ in the same fashion, and the object motion:

$${}^{W}_{k-1}\mathbf{H}_k = {}^{W}\mathbf{X}_k\ {}^{W}_{k-1}\mathbf{G}_k$$

can be directly recovered. As before, RANSAC is used to detect outliers and update the inlier tracks in the current frame. Building on [15], we enhance the initial object motion estimation and refine the inlier tracks through two additional steps, detailed in Sec. V-B4 and Sec. V-B5, respectively. The method proposed in Sec. V-B4 was used in [15] to improve the estimation and tracking. With the addition of Sec. V-B5

our method augments this process to further improve the estimation and identify outliers. At each stage of the refinement process, the set of inlier tracks is updated, significantly improving the robustness of the estimation. The three-steps outlined assumes that the motion of each object is independent, allowing DynoSAM to parallelize the object motion estimation process by handling each object instance in a separate thread.

4) Joint Optical-Flow Refinement: Following the work of [15], [54], the optical-flow is jointly refined with the camera pose and object motions. This step ensures robust and accurate tracking of both static and dynamic features and helps to account for any errors in the initial optical-flow calculation. The measured flow and the associated $\mathrm{SE}(3)$ transformation are jointly refined by reformulating the re-projection errors from (26) and (24) in terms of the measured flow:

$$\begin{aligned} \mathbf{r}_{f,H} &= \mathbf{z}^i_{\mathrm{2D},k-1} + \mathbf{f}^i_{k-1,k} - \pi({}_{k-1}^{W}\mathbf{G}_k \; {}^{W}\mathbf{m}^i_{k-1}) \\ \mathbf{r}_{f,X} &= \mathbf{z}^i_{\mathrm{2D},k-1} + \mathbf{f}^i_{k-1,k} - \pi({}^{W}\mathbf{X}_k^{-1} \; {}^{W}\mathbf{m}^i) \end{aligned} \tag{27}$$

where $f_{k-1,k}$ defines the optical flow between two keypoints:

$$f_{k-1,k} = \mathbf{z}_{\mathrm{2D},k} - \mathbf{z}_{\mathrm{2D},k-1}.$$

Given 3D-2D point correspondences, the resulting non-linear least squares problem is formulated using a factor graph:

$$\{\theta, \mathbf{f}_{k-1,k}\} = \underset{\{\theta, \mathbf{f}_{k-1,k}\}}{\operatorname{argmin}} \left(\sum_i^N \rho_h \|\mathbf{r}_{f,\theta}\|^2_{\Sigma_f} + \sum_i^N \|\mathbf{r}_0(f)\|^2_{\Sigma_0} \right), \tag{28}$$

where θ represents either ${}^{W}\mathbf{X}_k$ camera pose or ${}_{k-1}^{W}\mathbf{G}_k$ in the case of object motion estimation. Depending on θ, $\mathbf{r}_{f,\theta}$ is either $\mathbf{r}_{f,X}$ or $\mathbf{r}_{f,H}$, as in (27). The covariance matrix $\Sigma_0 \in \mathbb{R}^{2\times 2}$ associated with the flow prior $\mathbf{r}_0(f)$ is diagonal and associated with the measured optical-flow. As each object moves independently, we construct and solve this optimization problem in parallel per object. During optimization, we enhance robustness by eliminating additional outliers, specifically those exhibiting large re-projection errors.

5) Object Motion Refinement: The object motion estimates can be further refined by employing the rigid-body motion model defined in (16) to formulate an additional nonlinear least-square optimization problem that directly estimates the object motion ${}_{k-1}^{W}\mathbf{H}_k$. In contrast to the preceding step, which uses a 2D pixel error, this residual function is based on 3D point errors. Empirically, we have observed that this 3D-based approach significantly enhances estimation accuracy and removes additional outliers not previously identified. Using the motion estimate from the previous step as an initial estimate, the *motion only refinement graph* shown in Fig. 6 is build per object and defines the nonlinear least-squares problem:

$$\theta_{\mathbf{k}}^{\mathrm{MAP}} = \underset{\mathbf{H}}{\operatorname{argmin}} \Bigg(\|\mathbf{r}_0({}^{W}\mathbf{X}_{k-1})\|^2_{\Sigma_0} + \|\mathbf{r}_0({}^{W}\mathbf{X}_k)\|^2_{\Sigma_0} \tag{29.0}$$

$$+ \sum_{\mathcal{D}^j_{k\;\;1} \cup \mathcal{D}^j_k} \|\mathbf{r}_{\mathrm{2D},R}\|^2_{\Sigma_{\mathrm{2D},R}} \tag{29.1}$$

$$+ \sum_{\mathcal{D}^j_{k\;\;1} \cup \mathcal{D}^j_k} \|\mathbf{r}_{\mathrm{3D},H}\|^2_{\Sigma_{\mathrm{3D},H}} \Bigg), \tag{29.2}$$

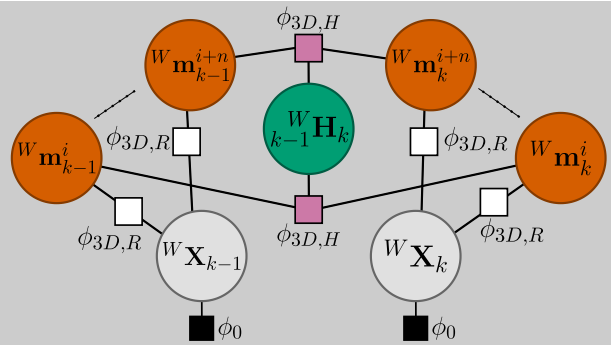


Fig. 6: Factor graph representing the motion only refinement graph. For each object $j \in \mathcal{J}_k$, the motion is refined by minimizing the 3D motion residual using measurements of points tracked between $k-1$ and k. In our example we show the dynamic points in orange, the motion in green, the 3D motion residual in purple and the re-projection factors in white. We only show two sets of points i and $i+n$, while in reality there would be n motion factors.

The Levenberg–Marquardt solver is used to obtain the final object motion estimate ${}_{k-1}^{W}\mathbf{H}_k$. Based on (24), (29.1) forms projection factors with an associated covariance matrix $\Sigma_{2D,R} \in \mathbb{R}^{2\times 2}$, connecting dynamic points and the observing camera poses at time-step $k-1$ and k. (29.2) describes the 3D motion residual of (16) and connects the common motion with each tracked landmark. $\Sigma_{3D,H} \in \mathbb{R}^{3\times 3}$ is the associated covariance matrix. Equation (29.0) represents prior factors on the camera poses. Since this refinement step focuses exclusively on improving object motion estimation, and not camera pose, a strong prior on the camera poses are used during optimization achieved by assigning the covariance matrix $\Sigma_0 \in \mathbb{R}^{6\times 6}$ with very small values (e.g. $\sigma = 0.0001$).

C. Back-end

The back-end fuses static and dynamic measurements to jointly estimate the trajectory of the camera and each object as well as a global map of static and dynamic points. Our back-end allows different Dynamic SLAM formulations to be selected and used for estimation. Full-batch and sliding window methods are provided for optimization.

As previously discussed, formulations can represent objects using different state variables (poses or motions as shown in this work) and in different frames of reference (locally as in DynaSLAM II [23] or in the camera frame as in MVO [17]). Therefore, to support modularity, evaluation and use in downstream applications DynoSAM defines a common interface between the formulation the system output. A pose-estimation processing step is implemented that converts all estimated states into a common output:

$$\mathcal{O}_k = [{}^{W}\mathbf{X}_k, {}_{k-1}^{W}\mathcal{H}_k, {}^{W}\mathcal{L}_k, {}^{W}\mathcal{M}_k] \; k \in \mathcal{K} \tag{30}$$

If the formulation used does not directly estimate a required output element, it must be computed directly from the available information. This behavior is enforced in DynoSAM's back-end implementation via clear layers of abstraction.

D. Implementation

The camera pose and motion estimation algorithms described in the previous sections are implemented in C++. GTSAM 4.2 [28] is used for all non-linear optimization

(a) (b) (c) (d) (e)

Fig. 7: Example scenarios from datasets used: **(a)**: KITTI tracking, **(b)**: OMD, **(c)**: Outdoor Cluster, **(d)**: TartanAir (Shibuya), **(d)**: VIODE.

problems which are solved using Levenberg-Marquardt. For all experiments, unless otherwise specified (Sec. VI-E), full-batch optimization is used.

At each time-step, the back-end component constructs a system by incorporating new factors and variables based on inputs from the front-end, which consists of initial estimates for the camera pose ${}^{W}\mathbf{X}_k$, per-object motions ${}^{W}_{k-1}\mathcal{H}_k$ and a set of visual measurements $\mathcal{Z}_k$. After optimisation, the estimates are processed to generate the output $\mathcal{O}_k$.

For both formulations discussed in Sec. IV, the system defines new variables at each time-step for ${}^{W}_{k-1}\mathbf{H}_k$ or ${}^{W}\mathbf{L}_k$ which represent the state of each object, and initializes from the front-end where applicable.

The combination of factors and state variables results in a Dynamic SLAM problem that, compared to a typical static SLAM system, is significantly more complex, and may contain ternary (16) and quaternary edges (21). Despite the significant increase in graph size and associated bookkeeping DynoSAM efficiently manages the construction of each graph. DynoSAM achieves this by decoupling the measurement bookkeeping from the implementation of individual Dynamic SLAM formulations through abstraction layers, as mentioned in the previous section. This design choice dramatically simplifies the implementation of new formulations, allowing researchers to easily experiment with different approaches. Further technical details regarding graph construction and the abstractions used are available in our documented open-source software.

Finally, we note that loop closure has not been incorporated into the system and is beyond the scope of this work.

VI. Experiments

The primary focus of this work is the estimation object motion and pose. Hence, we evaluate both DynoSAM's WCME (Sec. IV-B) and WCPE (Sec. IV-C) formulations on a range of indoor and outdoor datasets featuring diverse dynamic objects and contain ground truth for both camera pose and object motion/pose. To facilitate a wide range of comparison we evaluate DynoSAM on the KITTI tracking [68], OMD [52] and Outdoor Cluster [24][2]. We use VDO-SLAM [15], MVO [17] and ClusterSLAM [24] as state-of-the-art Dynamic SLAM systems to assess our object motion/pose estimation. Furthermore, we propose a new evaluation metric for object motion and discuss how it better facilitates comparison across systems.

In addition to object motion/pose, we evaluate our camera pose estimation by comparing against DynaSLAM II [23], VDO-SLAM [15], MVO [17], ClusterSLAM [24], AirDOS [29], ORB-SLAM3 [4] and DynaVINS [20]. Datasets that do not contain ground truth object information, such as TartanAir Shibuya [29] and VIODE [69], are also included to validate the accuracy and robustness of our camera pose estimation in highly dynamic and challenging environments. Our evaluation is conducted using the metrics outlined in Sec. VI-B and provides a comprehensive assessment of these systems. All evaluation metrics are included in our open-source framework and are based on the implementation of *evo* [70].

A. Datasets and Third Party Results

1) Sequences with Camera & Object Ground Truth: We use the KITTI tracking dataset [68], Outdoor Cluster [24] and OMD [52] (Fig. 7 a-c) to evaluate object motion and pose estimation in scenes with multiple objects, complex motion, occlusion, and semi-rigid entities such as cyclists. We use a processed version of the KITTI tracking dataset containing ground truth information per object per frame. Outdoor Cluster is simulated and provides ground truth. OMD uses a Vicon system to capture all ground truth trajectories [52].

KITTI and Outdoor Cluster represent large-scale outdoor driving scenarios with numerous moving objects. However, most of these objects experience relatively constant motion. We therefore additionally include *swinging_4_unconstrained* (S4U), the OMD sequence with the most challenging dynamics, which features 4 swinging cubes with unconstrained and unpredictable motions. The hand-held camera motion further increases the difficulty. While MVO uses another OMD sequence to test their object re-association ability after occlusion, this functionality is not relevant for motion evaluation and is therefore not assessed in this work.

2) Challenging Sequences with Only Camera Ground Truth: While not our primary focus, evaluating camera pose estimation is crucial to demonstrate DynoSAM's accuracy and robustness in challenging conditions, including dynamic objects, occlusions, and low light. We test on the simulated TartanAir Shibuya [29] and VIODE [69] datasets. TartanAir features over 30 moving humans with complex motions that frequently occlude the view (Fig. 7 (d)). VIODE poses further challenges with full occlusion scenarios as shown in Fig. 7 (e), low-light, and includes UAV-like motion with extreme translation and rotation. This dataset additionally contains IMU.

3) Obtaining Third Party Results For Object Motions: Obtaining consistent object motion/pose evaluation from third-party systems is challenging due to limited open-source options and different metrics and formats. Of the Dynamic SLAM systems we compare against, only VDO-SLAM [15] is open-source, and we used it to generate results directly. To ensure fairness we run VDO-SLAM on the same processed

[2]The datasets are provided in addition to the paper and can be found at https://huangjh-pub.github.io/page/clusterslam-dataset/. Note that the ground truth rotation is actually in $[qw, qx, qy, qz]$ form instead of the $[qx, qy, qz, qw]$ format stated on the website

54 pt
0.75 in
19.1 mm
54 pt
0.75 in
19.1 mm

data as DynoSAM. VDO-SLAM shares the underlying motion formulation with WCME but differs in its use of g2o [71] for optimizations, as well as in its front-end design and tracking strategy. MVO [17] reports object motions for a few sequences, and the authors kindly provided results on S4U and KITTI 00 for our evaluation. ClusterSLAM [24] uniquely reports average camera and object pose errors accumulated over all Outdoor Cluster sequences. We therefore report an accumulated average error on this dataset and compare against the values reported by Huang *et al.* [24]. DynaSLAM II [23] provides per-object pose errors for a few selected objects per sequence. On the KITTI dataset, without source code, we cannot align object IDs and thus unfortunately compare only camera pose for these sequences reported in their paper. We however note that DynoSAM improves upon DynaSLAM II's reported object pose estimation by 91% in rotation and 36% in translation.

B. Metrics

This section details the metrics used in our DynoSAM framework to evaluate the accuracy of pose and motion estimations for camera and objects. For any error $\mathbf{E}_k \in \mathrm{SE}(3)$ at time-step k, we compute and report the root-mean-squared error (RMSE) of the translation and rotation component separately throughout a sequence:

$$\begin{aligned} \mathbf{E}_t &= \mathrm{RMSE}(trans\,(\mathbf{E_k})) \\ \mathbf{E}_r &= \mathrm{RMSE}(rot\,(\mathbf{E_k})) \\ \mathrm{RMSE}(e) &= \sqrt{\frac{1}{n(\mathcal{K})}\sum_{k\in\mathcal{K}}\|e_k\|^2}, \end{aligned} \tag{31}$$

where e_k is the scalar error of either the L_2 norm of the translational component, denoted as $trans\,(\mathbf{E_k})$, or the angle of the rotational component, denoted as $rot\,(\mathbf{E_k})$, of $\mathbf{E}_k \in \mathrm{SE}(3)$.

1) Pose Evaluation: Camera and object pose estimation accuracy is evaluated using Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) as defined by Sturm *et al.* [72]. Given a ground truth transformation $\mathbf{M}_{\mathrm{gt,\,k}} \in \mathrm{SE}(3)$ at k and a corresponding reference estimate $\mathbf{M}_{\mathrm{k}} \in \mathrm{SE}(3)$, the ATE is defined as:

$$\mathbf{ATE} = \mathrm{RMSE}\big(trans\left(\mathbf{M}_{\mathrm{gt},k}^{-1}\,\mathbf{M}_k\right)\big). \tag{32}$$

The ground truth reference frame is emphasized in red to clearly distinguish it from the estimated reference frame.

ATE is commonly used to assess the global consistency in trajectory estimation [3], [23], [25] and we use ATE to evaluate the accuracy in camera pose. We do not evaluate ATE for objects since it greatly depends on the ATE of the camera and how the object frame is defined for each system.

We use the RPE metric for camera and object pose evaluation. Unlike in some classical visual SLAM systems [3], [68] where RPE evaluates the accumulated camera pose errors over distance traveled, this RPE metric quantifies the difference in relative poses between consecutive frames and is employed by state-of-the-art Dynamic SLAM systems [17], [23], [24], [25]:

$$\mathbf{RPE}_k = \left(\mathbf{M}_{\mathrm{gt},k-1}^{-1}\,\mathbf{M}_{\mathrm{gt},k}\right)^{-1}\left(\mathbf{M}_{k-1}^{-1}\,\mathbf{M}_k\right). \tag{33}$$

We report translation and rotation error separately using (31). For camera pose, we set $\mathbf{M} = {}^{W}\mathbf{X}_k$ to compute ATE and RPE. We assess object RPE by setting $\mathbf{M} = {}^{W}\mathbf{L}_k$.

2) Motion Evaluation: The primary scope of our work focuses not only on poses but, more importantly, on object motion estimation. For proper motion evaluation, we must again consider how the motion changes when represented in different frames. As discussed in Sec. III-A, the local motion of an object depends directly on where the body frame is defined on the object. Since the object frame may be arbitrarily defined, there is no guarantee that the frame $\{L\}$ for any object j will align between different systems and with the ground truth. Setting $\mathbf{M} = {}^{W}\mathbf{L}_k$ in (33) and evaluating the object RPE, as many systems do, highlights the potential discrepancy:

$$\mathbf{RPE}_k(\mathbf{L}) = {}^{L_{\mathrm{gt},k}}_{\;\;k-1}\mathbf{H}_{\mathrm{gt},k}^{-1}\;{}^{L_k}_{\;\;k-1}\mathbf{H}_k,$$

as this error compares the ground truth motion in the *ground truth* object frame with the estimated motion that is represented in the *estimated* object frame. Therefore, depending on the definition of $\{L\}$ relative to $\{L_{gt}\}$ the reported error may be highly inconsistent.

An obvious solution is to instead directly evaluate the observed motion ${}^{W}\mathbf{H}$. This representation is agnostic to the placement of the object frame on the rigid-body and is shared between all systems, including ground truth. However, expressing the motion error in the world frame has the unintended effect of 'scaling' the error's magnitude with the objects position relative to $\{W\}$, as shown in Appendix B. Consequently, even a small discrepancy in the actual motion can have a large error when the object is far from the origin.

To resolve both of these issues, we propose a new metric, Motion Error **ME**, that evaluates all motions in the *ground truth object frame* as a local object frame that is common to all systems. This metric can be easily calculated from the observed motion representation using (35) or if the transform between the estimated and ground truth object frame is well defined. For each object j at k, ME is defined as:

$$\mathbf{ME}_k = {}^{L_{\mathrm{gt},k}}_{\;\;k-1}\mathbf{H}_{\mathrm{gt},k}^{-1}\;{}^{L_{\mathrm{gt},k}}_{\;\;k-1}\mathbf{H}_k \tag{34}$$

where

$${}^{W}\mathbf{L}_{\mathrm{gt},k}\;{}^{\;\;1}_{k-1}\mathbf{H}_k = {}^{W}\mathbf{L}_{\mathrm{gt},k-1}^{-1}\;{}^{W}_{k-1}\mathbf{H}_k\;{}^{W}\mathbf{L}_{\mathrm{gt},k-1}. \tag{35}$$

Conceptually, ME compares the ground truth motion in the ground truth frame with the estimated motion also expressed in the ground truth object frame. Equation (35) demonstrates how the estimated motion ${}^{W}_{k-1}\mathbf{H}_k$ can be expressed in the ground truth object frame using only ground truth object poses.

While both RPE and ME evaluate the accuracy in relative transformations along an objects trajectory, ME is agnostic to how each system defines the object frame. By accounting for any differences in definitions, this metric facilitates valid comparisons between systems. As most existing systems primarily evaluate RPE for objects, we report both RPE and ME metrics. However, we primarily focus on analysing ME as we believe this is a better indicator of accuracy.

As before, we report the RMSE for the translation and rotation error throughout the sequence for all objects, follow-

TABLE I: Quantitative evaluations of camera trajectory (ATE and RPE) against other Dynamic SLAM systems. Entries marked with '-' indicate that results are not available for that sequence. Best results are marked in bold and second best with underscore.

		KITTI									Outdoor Cluster					OMD
		00	01	02	03	04	05	06	18	20	L1	L2	S1	S2	avg.	S4U
ATE(m)	DynaSLAM II	1.29	2.31	0.91	**0.69**	**1.42**	**1.34**	**0.19**	**1.09**	1.36	-	-	-	-	-	0.21
	VDO-SLAM	3.37	6.74	2.47	2.12	4.53	3.8	0.45	9.94	7.82	-	-	-	-	-	0.19
	ClusterSLAM	-	-	-	-	-	-	-	-	-	-	-	-	-	0.53	-
	MVO	1.53	-	-	-	-	-	-	-	-	-	-	-	-	-	**0.05**
	WCPE (ours)	**0.82**	**2.00**	**0.73**	0.82	2.01	1.58	0.31	1.84	**1.26**	**0.61**	**0.52**	**0.09**	**0.13**	**0.34**	0.11
	WCME (ours)	**0.82**	**2.00**	**0.73**	0.82	2.01	1.58	0.31	1.84	**1.26**	**0.61**	**0.52**	**0.09**	**0.13**	**0.34**	0.11
$RPE_r(°)$	DynaSLAM II	0.06	0.04	**0.02**	0.06	**0.06**	**0.03**	**0.04**	**0.02**	**0.04**	-	-	-	-	-	-
	VDO-SLAM	0.08	0.05	0.03	**0.03**	**0.06**	**0.03**	0.1	0.03	**0.04**	-	-	-	-	-	0.77
	ClusterSLAM	-	-	-	-	-	-	-	-	-	-	-	-	-	1.15	-
	MVO	0.19	-	-	-	-	-	-	-	-	-	-	-	-	-	0.76
	WCPE (ours)	0.05	**0.03**	**0.02**	0.05	**0.06**	0.06	0.05	0.04	**0.04**	**0.02**	**0.02**	**0.01**	**0.02**	**0.02**	**0.69**
	WCME (ours)	**0.04**	**0.03**	**0.02**	0.05	**0.06**	0.05	0.05	0.04	**0.04**	**0.02**	**0.02**	**0.01**	**0.02**	**0.02**	**0.69**
$RPE_t(m)$	DynaSLAM II	**0.04**	0.05	0.04	**0.04**	0.07	0.06	0.02	0.05	0.07	-	-	-	-	-	-
	VDO-SLAM	0.09	0.15	0.05	0.09	0.14	0.11	0.04	0.09	0.30	-	-	-	-	-	0.12
	ClusterSLAM	-	-	-	-	-	-	-	-	-	-	-	-	-	1.10	-
	MVO	0.07	-	-	-	-	-	-	-	-	-	-	-	-	-	**0.004**
	WCPE (ours)	**0.04**	**0.04**	**0.03**	0.05	0.07	**0.05**	**0.01**	**0.04**	0.04	**0.04**	**0.02**	**0.01**	**0.02**	**0.02**	0.006
	WCME (ours)	**0.04**	**0.04**	**0.03**	0.05	**0.06**	**0.05**	**0.01**	**0.04**	**0.02**	**0.04**	**0.01**	**0.01**	**0.02**	**0.02**	0.006

ing (31). We report average motion error for each sequence. However, DynoSAM also outputs per-object errors for more detailed comparison if necessary. Only objects observed for at least 3 consecutive frames are included.

3) Frame Alignment: To ensure our framework performs fair and accurate evaluation, it is vital that we correctly align the world, the camera and object frames with their respective ground truth frames. During experiments we align the ground truth origin to the estimated trajectory using Umeyama's method [73]. For camera pose evaluation, the estimated and ground truth states share a common and well defined reference frame, making alignment and comparison simple. To facilitate a valid evaluation of RPE, we define the start of each object trajectory using the first ground truth object pose, i.e. (19) to ensure a common object frame definition. To guarantee that the estimated objects also share the same world frame, we calculate ${}_{k-1}^{W}\mathbf{H}_{\mathrm{gt},k}$ and ${}^{W}\mathbf{L}_{\mathrm{gt},k}$ directly from the dataset using the newly aligned ground truth odometry.

C. Camera Pose Error

Table I, Table II and Table III show comparisons of camera pose estimation using ATE and RPE. Our results demonstrate that in RPE, we predominantly achieve better results than all other systems. In cases where we perform worse, the difference is marginal with the maximum error difference being $0.02°$ in rotation and $0.02\,\mathrm{m}$ in translation.

ATE is a measure of global consistency and accumulated drift. Across all datasets we demonstrate comparable and in some cases better results than other systems. Notably on the TartanAir (Shibuya) dataset, we perform better than VDO-SLAM in all cases and AirDOS in most sequences, as shown in Table II, demonstrating our robustness in environments with a large number of (non-rigid) dynamic objects. DynaSLAM II produces marginally better results over KITTI Sequence 03–18 in ATE. We believe this is due to our more conservative motion segmentation algorithm which results in relatively fewer static tracks, particularly in scenes with many objects. This introduces greater rotation bias into the visual odometry, notably on longer sequences, thus increasing the ATE.

The VIODE dataset is challenging due to the camera's complex motions and scenarios of full occlusion by dynamic objects. Song *et al.* have demonstrated with their Visual-Inertial Odometry (VIO) system, DynaVINS [20], that, in these situations, tight IMU integration is vital for robust ego-motion estimation as no static features are available. We therefore tightly integrate the IMU into our backend using the preintegration method of Forster *et al.* [74] for the VIODE experiments. This highlights the flexibility of our framework to easily integrate inertial data. Table III shows DynoSAM's superior RPE on the majority of sequences. Compared with ORB-SLAM3 [4] (running in IMU mode), we demonstrate superior robustness as DynoSAM completes all sequences while ORB-SLAM3 fails on three, as indicated by *. While we outperform DynaVINS [20] in RPE, DynaVINS produces the best ATE, likely due to its comprehensive mechanism that rejects dynamic object features using IMU-informed pose priors. Consequently, DynaVINS is more robust during the transition phase of total occlusion. These situations occur in all VIODE sequences and biases DynoSAM's eventual ATE, but do not affect our RPE accuracy on average. However, we believe that DynaVINS's VIO mechanism is complimentary to our system and can be integrated in the future to improve overall accuracy and robustness. Running DynoSAM without the IMU results in the estimation diverging, as indicated by † in Table III. We omit the numerical results due to space limitations and because it is clear from the DynaVINS paper that the IMU is needed in these challenging conditions [20].

Finally, both proposed DynoSAM formulations demonstrate almost identical performance. This is expected as they share a common formulation for visual odometry.

Overall we demonstrate our camera pose estimation is accurate and robust in various highly dynamic environments that include rigid and non-rigid objects, full occlusion and complex camera motion. We outperform most systems on RPE and produce comparable if not better ATE results. The IMU

54 pt
0.75 in
19.1 mm
54 pt
0.75 in
19.1 mm

TABLE II: ATE evaluation of camera pose on TartanAir (Shibuya) [29] compared against AirDOS [29] (with mask) and VDO-SLAM [15].

		TartanAir Shibuya						
		Standing Human		Road Crossing (Easy)			Road Crossing (Hard)	
		I	II	III	IV	V	VI	VII
ATE (m)	AirDOS	0.06	**0.02**	0.10	0.03	**0.02**	0.22	0.56
	VDO-SLAM	0.10	0.61	0.38	0.39	0.22	0.24	0.66
	WCPE (ours)	0.03	0.03	**0.02**	0.02	0.03	0.04	**0.18**
	WCME (ours)	**0.02**	0.04	**0.02**	**0.02**	0.04	**0.03**	0.19

TABLE III: ATE and RPE evaluation of camera pose on VIODE [69] compared against DynaVINS [20] and ORB-SLAM3 [4]. Both systems are run in stereo-inertial mode. DynoSAM is run with full IMU fusion in the backend and sequences marked with † indicate failure cases (i.e. trajectory divergence) without the use of the IMU. * indicates total failure cases. Mid/High labels indicate sequence difficulty and number of dynamic objects.

		VIODE					
		City Day		City Night		Parking Lot	
		Mid	High†	Mid	High†	Mid†	High†
ATE (m)	DynaVins	**0.104**	**0.150**	**0.194**	**0.147**	**0.056**	**0.065**
	ORB-SLAM3	0.217	*	1.693	3.006	*	*
	WCPE (ours)	2.515	2.128	1.360	2.560	1.377	0.764
	WCME (ours)	2.515	2.128	1.360	2.560	1.377	0.764
RPE_t(m)	DynaVins	0.024	0.027	0.019	**0.023**	0.019	0.015
	WCPE (ours)	**0.008**	**0.014**	**0.015**	0.020	**0.006**	**0.005**
	WCME (ours)	**0.008**	**0.014**	**0.015**	0.020	**0.006**	**0.005**
RPE_r(°)	DynaVins	0.087	**0.09**	0.102	**0.096**	0.126	0.111
	WCPE (ours)	**0.049**	0.105	**0.070**	0.190	**0.036**	**0.040**
	WCME (ours)	**0.049**	0.105	**0.070**	0.190	**0.036**	**0.040**

integration on the challenging VIODE dataset further demonstrates the flexibility of our framework and the robustness of our algorithm under these challenging conditions.

D. Object Motion and Pose Errors

Table IV report object motion and pose errors across all sequences. Table V summarizes the average percentage improvement of our approach over existing systems to highlight our framework's consistent accuracy. As discussed previously, we have found comprehensive comparison challenging due to the limited number of open-source systems and sequences that each system reports on. We report both RPE and ME evaluations, but we focus on ME since we believe this metric is more accurate for evaluation between systems (Sec. VI-B2). However, existing systems do not report ME and many are closed-source. We use '-' to indicate sequences where results could not be obtained.

ClusterSLAM [24] only reports RPE as an accumulated average. Table V shows that our framework is upwards of 6 times more accurate in both rotation and translation, demonstrating our superior performance.

Table V demonstrates that DynoSAM outperforms MVO [17] across all metrics, except in RPE_r on the OMD where MVO is more accurate by a small margin. Our framework is particularly accurate in translation, with a 39% improvement in ME_t and 12% improvement in RPE_t. However, reporting the ME individually for each object in the S4U sequence in Table VI shows that that DynoSAM outperforms MVO on most objects, thereby reinforcing our accuracy even on this highly dynamic dataset. We additionally outperform VDO-SLAM on all objects. Compared to DynaSLAM II we show an average 445% improvement in object ATE.

Compared to VDO-SLAM [15], our framework in general exhibits superior accuracy in object motion and pose estimation. Such improvement is particularly pronounced on the OMD, where DynoSAM is 34% more accurate in rotation and 21% in translation errors, as measured by the ME metric. However, there remains some situations such as KITTI 05, 06 and 18 where VDO-SLAM performs better. While the difference on KITTI 05 is marginal, 06 and 18 exhibit larger performance differences that merit further investigation. Although these two systems share a similar WCME formulation and are expected to yield comparable results, they differ in the solvers they use and in their underlying approaches to object tracking. Since we report performance averaged across all objects, such variations can significantly affect the final results, especially in sequences with many objects (KITTI 18). We believe this is a key reason for the discrepancies in the corresponding VDO-SLAM and DynoSAM performances. In the future, we will explore tracking quality by including the MOTP metric [23]; however, this is outside the main scope of this paper.

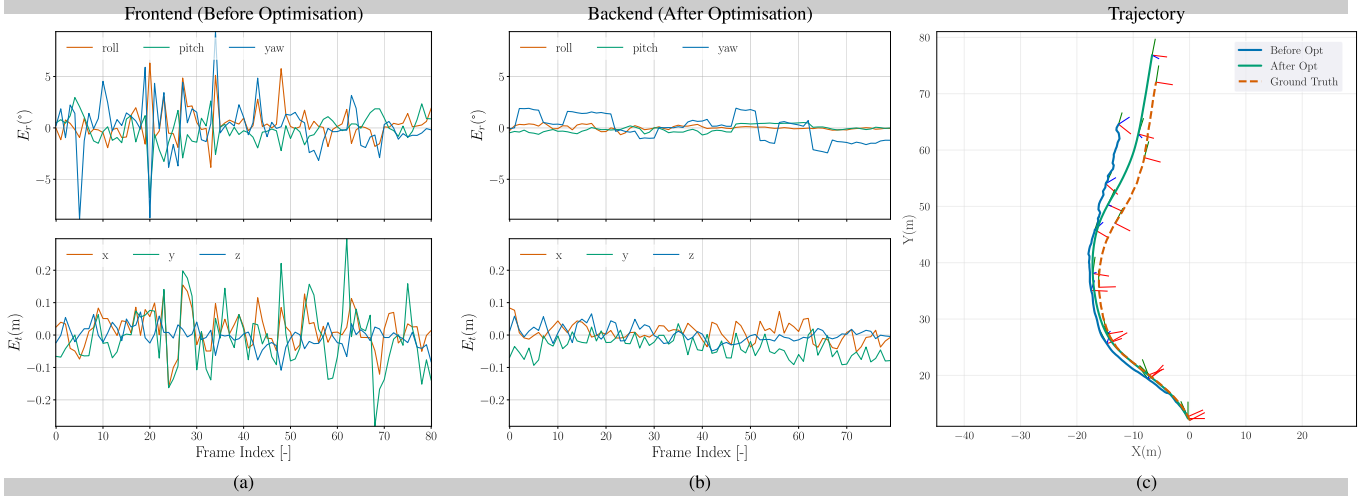
Finally we compare between WCME and WCPE. Across all sequences the average error difference is only 0.06 m in translation and 0.11° in rotation, demonstrating comparable performance. Out of the 14 sequences tested there are only three sequences (KITTI 03, 06 and 20) where the difference in error is greater than one standard-deviation, calculated independently for translation and rotation across all sequences. In these instances, WCPE's slightly degraded performance is likely due to its sensitivity to object pose initialization.

In WCPE, object poses are initialized at each time-step using the centroid of observed object points. Since these observations often represent only a partial view of the object, the resulting initialization may not be consistent with previous estimates, introducing additional noise. This limitation can be mitigated in future work by using pose prediction methods to provide better priors or serve as complementary measurements. While WCPE is not necessarily expected to outperform WCME in motion estimation specifically, it remains highly relevant, as direct pose estimation is attractive for many downstream applications such as navigation [27] and reconstruction [8]. Therefore, we believe that it remains relevant to introduce WCPE and examine its performance despite the slight degradation in motion accuracy.

Finally, Fig. 8 compares the per-frame ME before and after optimization on object 2 (cyclist) from KITTI 00, highlighting our accurate estimation across the entire trajectory. These results further demonstrate our systems capability to deal with only partially rigid objects, i.e. a cyclist, despite its motion breaking the explicit rigid-body assumption. We hypothesize that as long as the object experiences a single predominant motion, and sufficient object points can be tracked, our method will recover the overall trajectory, as shown in Fig. 8 (c).

TABLE IV: Quantitative evaluations of average object motions (ME) and poses (RPE) for all experiments. Entries marked with '-' indicate that results are not available for that sequence. Best results are marked in bold and second best with an underscore.

		KITTI									Outdoor Cluster					OMD
		00	01	02	03	04	05	06	18	20	L1	L2	S1	S2	avg.	S4U
$ME_r(°)$	VDO-SLAM	1.38	2.15	1.68	0.39	2.8	**0.48**	2.8	**0.36**	0.47	-	-	-	-	-	0.96
	ClusterSLAM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	MVO	3.36	-	-	-	-	-	-	-	-	-	-	-	-	-	1.1
	WCPE (ours)	**1.23**	0.91	**0.95**	0.27	**0.76**	0.56	2.8	1.15	0.39	0.86	0.74	**0.68**	2.40	1.17	1.6
	WCME (ours)	1.29	**0.86**	1.06	**0.26**	1.01	0.49	**0.39**	0.6	**0.33**	**0.82**	**0.70**	0.69	**2.36**	**1.14**	**0.71**
$ME_t(m)$	VDO-SLAM	0.11	0.35	0.43	**0.15**	0.38	0.19	**0.11**	**0.16**	0.57	-	-	-	-	-	**0.02**
	ClusterSLAM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	MVO	0.27	-	-	-	-	-	-	-	-	-	-	-	-	-	0.03
	WCPE (ours)	**0.09**	0.40	0.73	**0.15**	0.1	0.14	0.22	0.31	0.39	**0.07**	0.08	**0.03**	**0.13**	**0.08**	0.07
	WCME (ours)	0.15	**0.34**	**0.4**	**0.15**	**0.09**	**0.13**	**0.11**	0.20	**0.05**	0.08	**0.06**	0.04	0.15	**0.08**	**0.02**
$RPE_r(°)$	VDO-SLAM	1.40	1.25	1.34	0.32	1.04	0.64	**1.49**	**0.38**	0.47	-	-	-	-	-	3.3
	ClusterSLAM	-	-	-	-	-	-	-	-	-	-	-	-	-	10.3	-
	MVO	2.7	-	-	-	-	-	-	-	-	-	-	-	-	-	**3.1**
	WCPE (ours)	**1.27**	0.89	**1.02**	**0.27**	**0.74**	**0.6**	3.00	1.38	0.37	1.7	2.34	**0.61**	2.07	1.68	4.1
	WCME (ours)	1.38	**0.80**	1.06	**0.27**	1.04	0.62	2.74	1.16	**0.33**	**1.3**	**2.28**	0.67	**2.00**	**1.56**	3.2
$RPE_t(m)$	VDO-SLAM	0.28	0.34	**0.30**	0.20	0.94	0.17	**0.46**	**0.13**	**0.09**	-	-	-	-	-	0.06
	ClusterSLAM	-	-	-	-	-	-	-	-	-	-	-	-	-	8.65	-
	MVO	0.28	-	-	-	-	-	-	-	-	-	-	-	-	-	0.05
	WCPE (ours)	0.43	1.18	2.0	0.67	1.28	2.6	1.84	2.17	**0.09**	**1.84**	**0.74**	**0.99**	1.63	1.33	0.06
	WCME (ours)	**0.27**	**0.32**	0.79	**0.19**	**0.92**	**0.16**	0.48	0.2	0.12	1.9	0.72	0.94	**1.50**	**1.27**	**0.04**



(a) (b) (c)

Fig. 8: Example of per-frame ME before (**a**) and after (**b**) optimization using the WCME. We show results for object 2, a non-rigid body (cyclist) on KITTI 00. Translation and rotation errors are shown individually; along the top and bottom rows respectively. The resulting trajectories are compared to the ground truth in (**c**). These results demonstrate our systems capacity to handle non-rigid bodies in the case where the object experiences a predominant motion.

TABLE V: Average percentage improvement of DynoSAM Motion estimator compared to state-of-the-art Dynamic SLAM systems. Green cell blocks indicate improvement over existing systems while red blocks indicate relatively worse accuracy.

	VDO-SLAM [15]	ClusterSLAM [24]	MVO [17]
$ME_r(°)$	+28%	–	+49%
$ME_t(m)$	+9%	–	+39%
$RPE_r(°)$	−9%	+660%	−9%
$RPE_t(m)$	+4%	+681%	+12%

TABLE VI: Per-object ME and ATE errors on the OMD (S4U) using the WCME. Object ID's correspond with the following object's in the dataset - 1: top left, 2: top right, 3: bottom left, 4: bottom right.

		1	2	3	4
$ME_r(°)$	VDO-SLAM	1.256	0.770	0.907	0.927
	MVO	**0.542**	0.843	1.648	0.854
	DynoSAM (ours)	1.138	**0.544**	**0.443**	**0.474**
$ME_t(m)$	VDO-SLAM	0.0243	0.0234	0.0148	0.0293
	MVO	**0.0169**	0.0269	0.0232	0.0309
	DynoSAM (ours)	0.0214	**0.0233**	**0.0086**	**0.0291**
ATE(m)	DynaSLAM II	0.41	0.37	1.09	0.28
	DynoSAM (ours)	**0.09**	**0.21**	**0.08**	**0.15**

E. Sliding Window Optimization

We have so far presented results from a full-batch solution that incorporates all measurements into a single optimization. This solution, while accurate, is computationally intensive and unsuitable for online use. Moreover, we hypotheise that object motions should exhibit increasing independence over extended time horizons, even for objects with strong motion priors, such as vehicles on highways, thus making optimization over the global trajectory potentially unnecessary.

Therefore, we implemented a sliding-window approach that solves smaller, more efficient batch problems every w frames,

TABLE VII: Average object ME comparison of full-batch vs. sliding window approach on the KITTI and OMD datasets.

	Full-Batch		Sliding Window	
	$\mathrm{ME}_r(^\circ)$	$\mathrm{ME}_t(\mathrm{m})$	$\mathrm{ME}_r(^\circ)$	$\mathrm{ME}_t(\mathrm{m})$
KITTI 00	1.11	0.072	**1.039**	**0.065**
OMD (S4U)	0.729	0.022	**0.659**	**0.021**

reusing estimates from the previous window to initialize overlapping variables. As shown in Table VII the sliding-window method achieves marginally better per-frame error. Furthermore, Fig. 9 compares the per-frame error of each approach. The average ME for each object on the KITTI 00 and OMD sequences is reported. Note that full-batch demonstrates slightly more consistent accuracy, as seen in Fig. 9 (a) between frames 70 and 85. The larger error occurs at window overlap and is likely a result of poor front-end tracking at that frame. Our preliminary results indicate that incorporating motion data over a receding time horizon is beneficial.

F. Computation Time

Fig. 10 presents the runtime breakdown for each module in the front-end. The feature tracking module, which includes feature extraction, matching, and geometric verification, requires less than 50 ms. The motion estimation module, responsible for estimating the camera pose and all object motions per frame, averages 100 ms to produce all estimates. However, the majority of processing time is spent on the object motion refinement component, which takes approximately ∼250 ms per object to solve the nonlinear optimization problem. The runtime of this component depends heavily on the number of points involved in the optimization, leading to significant variation in processing times.

Fig. 9: Object motion errors comparing full-batch vs sliding window. We take the **ME** (rotation and translation) for each object *per frame* and report the average for KITTI 00 **(a)** and OMD (S4U) **(b)**. The WCME is used for both sets of results.

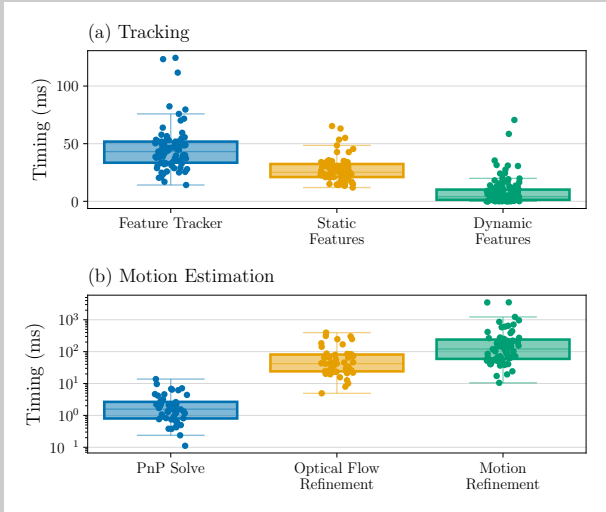


Fig. 10: Runtime breakdown of DynoSAM's front-end. **(a)** shows the timing for feature tracking and **(b)** shows the timing per each estimation module discussed in Sections V-B2, V-B3 (PnP Solve), V-B4 (Joint Optical Flow) and V-B5 (3D Motion Refinement). **(b)** uses log-scale for the timing axes.

In our experiments, the time required for full-batch optimization ranges from 80 s to 700 s, depending on the size of the constructed factor graph. While the full-batch optimization may not meet real-time requirements, it is important to note that practical applications typically rely on much smaller sliding window optimizations in the back-end. This approach significantly reduces computational demands and improves efficiency for online operation. In our experiments, using a window size of 20, the average sliding-window optimization is ∼16 s. Future work will focus on enhancing the efficiency of the back-end, including techniques such as conditional variable elimination [75], to better support real-time applications.

VII. Downstream Tasks

The proposed Dynamic SLAM system accurately estimates the motions of dynamic objects, as validated in Sec. VI. By eliminating the need for prior knowledge of motion models or object categories, our framework enables a range of downstream applications. This section explores how accurate motion estimation, produced by our pipeline, can facilitate dynamic object reconstruction and trajectory prediction, both of which are crucial for navigation systems.

A. Dynamic Object Reconstruction

Many reconstruction systems assume the environment to be static and rigid [76], [77], as with static SLAM systems. Leveraging DynoSAM for object segmentation and motion estimation, we show that the output of DynoSAM can be directly used by a reconstruction system, such as DynORecon [8], to incrementally map dynamic rigid-body objects.

The accurate motion estimation from DynoSAM facilitates an easy and efficient integration of new object measurements into each existing object reconstruction while maintaining object rigidity. Based on (9), applying ${}_{k-1}^{W}\mathbf{H}_k$ as estimated by DynoSAM to all points on the object guarantees a consistent motion from time-step $k-1$ to k. Equation (2) and (4) further

show that the same motion can consistently transform any reference frame fixed to the object. Similar to the treatment of object pose in Sec. IV-B and Sec. IV-C, we can define an arbitrary reference frame that is rigidly attached to the object body without any prior knowledge of its pose, and use ${}_{k-1}^{W}\mathbf{H}_k$ to move this body-fixed frame with respect to the global world frame. This allows us to express all object points in the body-fixed frame, ${}^{L}\mathbf{m}$, where they are static and time-invariant with respect to their local frame. This representation allows new observations to be integrated into each object reconstruction while remaining consistent with previous measurements [8].

Fig. 11 presents an example of DynORecon incrementally constructing all 4 dynamic objects (free-floating cubes) in addition to the static background using DynoSAM's estimations in the OMD experiment [52]. As shown in Fig. 11 (b-d), accurate motion estimations from the upstream Dynamic SLAM system enables a consistent incremental surface reconstruction as the object undergoes complex movement. Building up a correct representation of dynamic objects, as more of them are observed, provides a more comprehensive understanding of moving obstacles in the scene, and is therefore beneficial to other robotic applications such as planning and navigation.

B. Object Trajectory Prediction

Trajectory prediction plays a key role in path planning and control in dynamic environments. Safer and more efficient navigation can be achieved by incorporating the anticipated future movements of dynamic objects into navigation algorithms [10], [26], [78], [79].

We therefore integrated our DynoSAM pipeline with Trajectron++ [27], a state-of-the-art trajectory prediction algorithm that uses a graph-structured recurrent model to forecast trajectories. Trajectron++ uses current and historical state information, including pose and velocity for each object in the scene to generate real-time 2D predictions. Object poses are taken directly from DynoSAM's pose estimates ${}^{W}\mathbf{L}_k$, while velocities are calculated from the estimated motion ${}_{k-1}^{W}\mathbf{H}_k$. This processed DynoSAM output is used to train and test

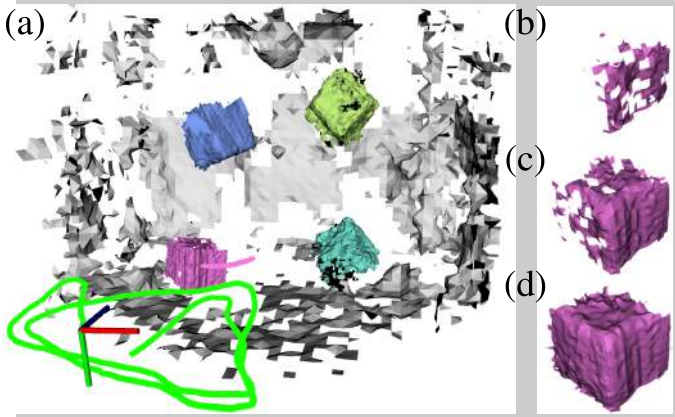


Fig. 11: Our downstream application, DynORecon [8], incrementally builds up surface mesh reconstructions of both dynamic objects and static background in the OMD (S4U) experiment [52]. **(a)**: dynamic object meshes and their trajectories (uniquely colored) as well as the static background (grey), in addition to camera pose and trajectory (green); **(b-d)**: the incrementally constructed mesh of Object 3.

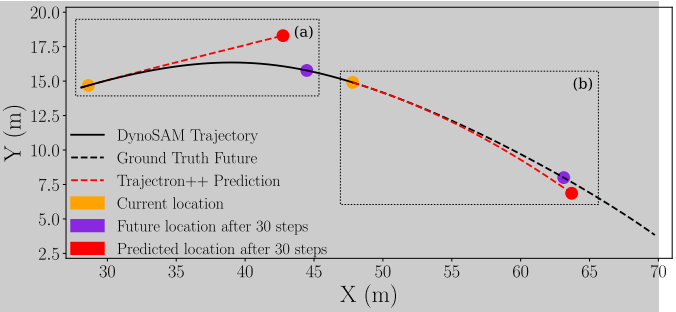


Fig. 12: Predicted object trajectories based on the output estimation of DynoSAM. We show inferred predictions compared to the future ground truth trajectory of Object 2 on KITTI 00. Two snapshots are shown along the object's trajectory where **(a)** illustrates an initial prediction based on only 1 previous state and **(b)** shows prediction based on 4 prior states.

Trajectron++ on the KITTI tracking dataset (Sec. VI-A). We trained the base Trajectron++ model from scratch within the adaptive prediction framework [80].

Fig. 12 presents a snapshot of Object 2's (cyclist) predicted trajectory at frames 67 and 103 (a-b respectively). On average, our predictions closely track the ground truth trajectory without exhibiting significant overshooting. These results, while preliminary, demonstrate that estimates from our Dynamic SLAM framework are accurate enough for trajectory predictions tasks, and our prediction results are of sufficiently high accuracy to be leveraged for planning modules [9], [79], [81], or as informative motion priors within our Dynamic SLAM framework. Furthermore, our future work will investigate trajectory forecasting algorithms that generate 3D predictions fully leveraging the SE(3) output of DynoSAM.

VIII. Limitations and Future Work

Our proposed Dynamic SLAM method achieves high accuracy in both motion and object trajectory estimation, highlighting the crucial role of the back-end in generating precise and smooth object trajectories. While the front-end already operates close to real time, the back-end is currently solved in a full-batch manner and will be improved in future work to support incremental inference.

Currently, our system uses a simplistic approach that relies on a constant motion constraint to model changes in object motion. As discussed in [17], this is akin to a constant velocity model, i.e. zero acceleration, which produces locally plausible motion but cannot capture motions of accelerating objects as accurately. Moving forward, we will exploring more sophisticated motion models, potentially through learning, which better capture an objects the short-term dynamics.

In addition, our system is currently designed to estimate the motion of rigid-bodies. While we have shown examples demonstrating its robustness in semi-rigid cases, a more comprehensive approach to non-rigid cases is highly relevant.

Finally, we plan to leverage advanced transformer models such as SAM2 [82], [83] and FlowFormer [59] to improve scene understanding and explore new directions in Dynamic SLAM. We have designed DynoSAM with an emphasis on modularity and extensibility to facilitate such further research.

IX. Conclusion

We have introduced DynoSAM, a cutting-edge, open-source framework for Dynamic SLAM that represents a significant advancement in the robotics field. By outlining the key theoretical concepts and formulations underpinning our approach, we provide a robust foundation for tackling the challenges of estimation in dynamic environments. DynoSAM offers a well-structured platform for implementing, testing, and evaluating Dynamic SLAM solutions, empowering researchers to develop and benchmark innovative methodologies with greater ease and precision. Importantly, our implementation is designed for flexibility, with clearly define interfaces between modules, facilitating integration with existing and new methods. This contribution paves the way for more reliable and adaptable robotic systems in dynamic and complex settings.

This paper thoroughly examines state-of-the-art methods for the Dynamic SLAM problem and introduces a novel formulation tailored for real-world applications. We highlight the importance of framing the problem in terms of observed motion, which enables accurate estimation and recovery of object trajectories. Additionally, we evaluate all the discussed formulations and demonstrate that our framework outperforms existing systems in both motion estimation and visual odometry, setting a new benchmark for Dynamic SLAM solutions.

The paper also demonstrates DynoSAM's effectiveness in downstream tasks such as motion prediction and 3D reconstruction. These capabilities collectively form the foundation for future dynamic object-aware navigation systems. With its user-friendly infrastructure and comprehensive evaluation suite, we aim for DynoSAM to serve as a robust platform for advancing research in Dynamic SLAM.

Acknowledgments

We would like to thank Kevin Judd and Jonathan Gammell for providing the results of MVO and their discussions. We additionally thank Tim Bailey for pointing out the scaling behavior when evaluating world frame motion error.

Appendix

A. Rigid-body motion on points in the world frame

As discussed in the work of Zhang *et al.* [15], for any point on a rigid-body object measured in the world frame ${}^{W}\mathbf{m}_k$, there is the following equation:

$$\begin{aligned} {}^{W}\mathbf{m}_k &= {}^{W}\mathbf{L}_k \; {}^{L_k}\mathbf{m}_k \\ &= {}^{W}\mathbf{L}_{k-1} \; {}^{L_k}_{k-1}\mathbf{H}_k \; {}^{L_k}\mathbf{m}_k \end{aligned} \tag{36}$$

since ${}^{W}\mathbf{L}_k = {}^{W}\mathbf{L}_{k-1} \; {}^{L_k}_{k-1}\mathbf{H}_k$.

Based on the rigid-body assumption of this object, ${}^{L_k}\mathbf{m}_k$ is time-invariant, and therefore ${}^{L}\mathbf{m} = {}^{L_k}\mathbf{m}_k = {}^{L_k}{}^{1}\mathbf{m}_{k-1}$:

$$\begin{aligned} {}^{W}\mathbf{m}_k &= {}^{W}\mathbf{L}_{k-1} \; {}^{L_k}_{k-1}\mathbf{H}_k \; {}^{L_k}{}^{1}\mathbf{m}_{k-1} \\ &= {}^{W}\mathbf{L}_{k-1} \; {}^{L_k}_{k-1}\mathbf{H}_k \; {}^{W}\mathbf{L}_{k-1}^{-1} \; {}^{W}\mathbf{m}_{k-1} \\ &= {}^{W}_{k-1}\mathbf{H}_k \; {}^{W}\mathbf{m}_{k-1} \end{aligned} \tag{37}$$

where ${}^{W}_{k-1}\mathbf{H}_k := {}^{W}\mathbf{L}_{k-1} \; {}^{L_k}_{k-1}\mathbf{H}_k \; {}^{W}\mathbf{L}_{k-1}^{-1}$, an operation referred to as *a frame change of a pose transformation* by Chirikjian *et al.* [53].

B. Constant motion model in different frames

As discussed by Henein *et al.* [11], we can show that if the body-fixed frame pose change is constant then the absolute reference frame change is constant too:

$$\begin{aligned} {}^{\mathbf{L}_k}_{k-1}{}^{1}\mathbf{H}_k &= \mathbf{C} = {}^{\mathbf{L}_k}_{k}\mathbf{H}_{k+1} \\ {}^{W}_{k-1}\mathbf{H}_k &= {}^{W}\mathbf{L}_{k-1} \; \mathbf{C} \; {}^{W}\mathbf{L}_{k-1}^{-1} \\ {}^{W}_{k}\mathbf{H}_{k+1} &= {}^{W}\mathbf{L}_{k} \; \mathbf{C} \; {}^{W}\mathbf{L}_{k}^{-1} \\ \text{given that,} \quad {}^{W}\mathbf{L}_k &= {}^{W}\mathbf{L}_{k-1} \; \mathbf{C} \\ {}^{W}_{k}\mathbf{H}_{k+1} &= {}^{W}\mathbf{L}_{k-1} \; \mathbf{C} \; \mathbf{C} \; \mathbf{C}^{-1} \; {}^{W}\mathbf{L}_{k-1}^{-1} \\ &= {}^{W}\mathbf{L}_{k-1} \; \mathbf{C} \; {}^{W}\mathbf{L}_{k-1}^{-1} \\ &= {}^{W}_{k-1}\mathbf{H}_k \end{aligned} \tag{38}$$

However, when the change in local motion is not constant, the change in absolute motion scales with distance the object is from $\{W\}$. This relationship is highly relevant for the object smoothing factor in (17) which enforces a constant motion constraint in the $\{W\}$ frame. Let the change in local motion be defined as:

$${}^{L}\mathbf{\Delta} = {}^{\mathbf{L}_k}_{k}\mathbf{H}_{k+1}^{-1} \; {}^{\mathbf{L}_{k+1}}_{k+1}\mathbf{H}_{k+2} \in \mathrm{SE}(3) \tag{39}$$

and the change in absolute motion:

$${}^{W}\mathbf{\Delta} = {}^{W}_{k}\mathbf{H}_{k+1}^{-1} \; {}^{W}_{k+1}\mathbf{H}_{k+2} \in \mathrm{SE}(3) \tag{40}$$

We can then say:

$$\begin{aligned} {}^{W}_{k}\mathbf{H}_{k+1} &= {}^{W}\mathbf{L}_{k+1} \; {}^{W}\mathbf{L}_{k}^{-1} \\ {}^{W}_{k+1}\mathbf{H}_{k+2} &= {}^{W}\mathbf{L}_{k+2} \; {}^{W}\mathbf{L}_{k+1}^{-1} \\ &= {}^{W}\mathbf{L}_{k+1} \; {}^{W}\mathbf{L}_{k}^{-1} \; {}^{W}\mathbf{L}_{k+1} \; {}^{L}\mathbf{\Delta} \; {}^{W}\mathbf{L}_{k+1}^{-1} \\ &= {}^{W}_{k}\mathbf{H}_{k+1} \; {}^{W}\mathbf{L}_{k+1} \; {}^{L}\mathbf{\Delta} \; {}^{W}\mathbf{L}_{k+1}^{-1} \end{aligned} \tag{41}$$

Substituting (41) into (40) defines the relationship between the change in motion when represented in the local-body and world frames:

$${}^{W}\mathbf{\Delta} = {}^{W}\mathbf{L}_{k+1} \; {}^{L}\mathbf{\Delta} \; {}^{W}\mathbf{L}_{k+1}^{-1} \tag{42}$$

Consequently, the residual and covariance of the object smoothing factor will increase proportionally with the object's pose. To mitigate this effect for our smaller-scale experiments, we set the covariance of the smoothing factor to a sufficiently large value. The same 'scaling' behavior as represented by (42) is also present when evaluating motion errors using ${}^{W}\mathbf{H}$ instead of ${}^{L}\mathbf{H}$, as a small discrepancies in the local motion ${}^{L}\mathbf{H}$ can lead to a large error in ${}^{W}\mathbf{H}$ due to (42).

References

[1] D. M. Rosen, K. J. Doherty, A. Terán Espinoza, and J. J. Leonard, "Advances in inference and representation for simultaneous localization and mapping," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 215–242, 2021.

[2] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE/ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

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

[4] C. Campos, R. Elvira, J. J. Gómez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," *IEEE Trans. Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[5] 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 *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 1455–1462.

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

[7] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proc. of the IEEE Intl. Conf. Computer Vision and Pattern Recognition*, June 2015.

[8] Y. Wang, J. Morris, L. Wu, T. Vidal-Calleja, and V. Ila, "Dynorecon: Dynamic object reconstruction for navigation," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2025.

[9] M. N. Finean, W. Merkt, and I. Havoutis, "Simultaneous scene reconstruction and whole-body motion planning for safe operation in dynamic environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021, pp. 3710–3717.

[10] A. Hermann, J. Bauer, S. Klemm, and R. Dillmann, "Mobile manipulation planning optimized for gpgpu voxel-collision detection in high resolution live 3d-maps," in *Intl. Symp. on Robotics/Robotik*, 2014, pp. 1–8.

[11] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic SLAM: The Need for Speed," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 2123–2129.

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

[13] A. Deeb, Y.-J. Pan, and M. L. Seto, "Model-based quasi-static slam in unstructured dynamic environments," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2021, pp. 196–201.

[14] T. P. Setterfield, D. W. Miller, J. J. Leonard, and A. Saenz-Otero, "Mapping and determining the center of mass of a rotating object using a moving observer," *Intl. J. of Robotics Research*, vol. 37, no. 1, pp. 83–103, 2018.

[15] J. Zhang, M. Henein, R. Mahony, and V. Ila, "VDO-SLAM: A Visual Dynamic Object-aware SLAM System," *arXiv preprint arXiv:2005.11052*, 2020.

[16] Y. Liu, C. Guo, Y. Luo, and Y. Wang, "Dynameshslam: A mesh-based dynamic visual slammot method," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5791–5798, 2024.

[17] K. M. Judd and J. D. Gammell, "Multimotion Visual Odometry (MVO)," *Intl. J. of Robotics Research*, 2024.

[18] J. Morris, Y. Wang, and V. Ila, "The importance of coordinate frames in dynamic slam," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2024.

[19] R. Tian, Y. Zhang, L. Yang, J. Zhang, S. Coleman, and D. Kerr, "Dynaquadric: Dynamic quadric slam for quadric initialization, mapping, and tracking," *IEEE Trans. on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 17 234–17 246, 2024.

[20] S. Song, H. Lim, A. J. Lee, and H. Myung, "Dynavins: A visual-inertial slam for dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 523–11 530, 2022.

[21] ——, "Dynavins++: Robust visual-inertial state estimator in dynamic environments by adaptive truncated least squares and stable state recovery," *IEEE Robotics and Automation Letters*, 2024.

[22] K. M. Judd, J. D. Gammell, and P. Newman, "Multimotion visual odometry (MVO): Simultaneous estimation of camera and third-party motions," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3949–3956.

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

[24] J. Huang, S. Yang, Z. Zhao, Y. Lai, and S. Hu, "Clusterslam: A slam backend for simultaneous rigid body clustering and motion estimation," in *Proc. of the Intl. Conf. on Computer Vision (ICCV)*, 2019, pp. 5874–5883.

[25] 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. of the IEEE/CVF Intl. Conf. Computer Vision and Pattern Recognition*, 2020, pp. 2168–2177.

[26] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 5628–5635.

[27] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Proc. of the European Conf. on Computer Vision (ECCV)*. Springer, 2020, pp. 683–700.

[28] F. Dellaert and GTSAM Contributors, "borglab/gtsam," May 2022. [Online]. Available: https://github.com/borglab/gtsam

[29] Y. Qiu, C. Wang, W. Wang, M. Henein, and S. Scherer, "Airdos: Dynamic slam benefits from articulated objects," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2022, pp. 8047–8053.

[30] D. Thomas, W. Woodall, and E. Fernandez, "Next-generation ROS: Building on DDS," in *ROSCon Chicago 2014*. Mountain View, CA: Open Robotics, sep 2014.

[31] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020.

[32] 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.

[33] W. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, "Rgb-d slam in dynamic environments using point correlations," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 373–389, 2022.

[34] R. Hachiuma, C. Pirchheim, D. Schmalstieg, and H. Saito, "Detectfusion: Detecting and segmenting both known and unknown dynamic objects in real-time slam," *arXiv preprint arXiv:1907.09127*, 2019.

[35] T. Zhang, H. Zhang, Y. Li, Y. Nakamura, and L. Zhang, "Flowfusion: Dynamic dense rgb-d slam based on optical flow," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7322–7328.

[36] L. Schmid, M. Abate, Y. Chang, and L. Carlone, "Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments," in *Proc. of the Robotics: Science and Systems (RSS)*, 2024.

[37] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart, "Dynablox: Real-time detection of diverse dynamic objects in complex environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6259 – 6266, 2023.

[38] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust Dense Mapping for Large-Scale Dynamic Environments," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018.

[39] Q. Zhang, Y. Yang, H. Fang, R. Geng, and P. Jensfelt, "Deflow: Decoder of scene flow network in autonomous driving," *arXiv preprint arXiv:2401.16122*, 2024.

[40] Q. Zhang, Y. Yang, P. Li, O. Andersson, and P. Jensfelt, "Seflow: A self-supervised scene flow method in autonomous driving," *arXiv preprint arXiv:2407.01702*, 2024.

[41] Z. Wang, Y. Wei, Y. Rao, J. Zhou, and J. Lu, "3d point-voxel correlation fields for scene flow estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13 621–13 635, 2023.

[42] X. Li, J. Zheng, F. Ferroni, J. K. Pontes, and S. Lucey, "Fast neural scene flow," in *Proc. of the Intl. Conf. on Computer Vision (ICCV)*, 2023, pp. 9844–9856.

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

[44] I. Ballester, A. Fontán, J. Civera, K. H. Strobl, and R. Triebel, "Dot: Dynamic object tracking for visual slam," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2021, pp. 11 705–11 711.

[45] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," *IEEE Trans. Robotics*, vol. 35, no. 4, pp. 925–938, 2019.

[46] P. Li, T. Qin, *et al.*, "Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018, pp. 646–661.

[47] M. Gonzalez, E. Marchand, A. Kacete, and J. Royan, "Twistslam: Constrained slam in dynamic environment," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6846–6853, 2022.

[48] M. Gonzalez, E. Marchand, A. Kacete, and J. Royan, "Twistslam++: Fusing multiple modalities for accurate dynamic semantic slam," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 9126–9132.

[49] L. Nicholson, M. Milford, and N. Sünderhauf, "QuadricSLAM: Dual Quadrics as SLAM Landmarks," in *Proc. of the IEEE/CVF Intl. Conf. Computer Vision and Pattern Recognition*, 2018, pp. 313–314.

[50] D. F. Henning, C. Choi, S. Schaefer, and S. Leutenegger, "Bodyslam++: Fast and tightly-coupled visual-inertial camera and human motion tracking," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2023, pp. 3781–3788.

[51] J. Wang, M. Rünz, and L. Agapito, "Dsp-slam: Object oriented slam with deep shape priors," in *Proc. of Intl. Conf. on 3D Vision*, 2021, pp. 1362–1371.

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

[53] G. S. Chirikjian, R. Mahony, S. Ruan, and J. Trumpf, "Pose changes from a different point of view," in *Proc. of the ASME Intl. Design Engineering Technical Conf. (IDETC)*. ASME, 2017.

[54] J. Zhang, M. Henein, R. Mahony, and V. Ila, "Robust Ego and Object 6-DoF Motion Estimation and Tracking," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 5017–5023.

[55] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "Rmpe: Regional multi-person pose estimation," in *Proc. of the Intl. Conf. on Computer Vision (ICCV)*, 2017, pp. 2353–2362.

[56] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[57] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer, 2003, pp. 363–370.

[58] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Proc. of the European Conf. on Computer Vision (ECCV)*. Springer, 2020, pp. 402–419.

[59] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, and H. Li, "Flowformer: A transformer architecture for optical flow," in *Proc. of the European Conf. on Computer Vision (ECCV)*. Springer, 2022, pp. 668–685.

[60] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[61] J. Shi *et al.*, "Good features to track," in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, 1994, pp. 593–600.

[62] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

[63] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, 1981, pp. 674–679.

[64] O. Bailo, F. Rameau, K. Joo, J. Park, O. Bogdan, and I. S. Kweon, "Efficient adaptive non-maximal suppression algorithms for homogeneous spatial keypoint distribution," *Pattern Recognition Letters*, vol. 106, pp. 53–60, 2018.

[65] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," in *Proc. of the European Conf. on Computer Vision (ECCV)*. Springer, 2022, pp. 1–21.

[66] T. Ke and S. I. Roumeliotis, "An efficient algebraic solution to the perspective-three-point problem," in *Proc. of the IEEE Intl. Conf. Computer Vision and Pattern Recognition*, 2017.

[67] L. Kneip and P. Furgale, "Opengv: A unified and generalized approach to real-time calibrated geometric vision," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1–8.

[68] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Intl. J. of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[69] K. Minoda, F. Schilling, V. Wüest, D. Floreano, and T. Yairi, "Viode: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1343–1350, 2021.

[70] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.

[71] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3607–3613.

[72] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 573–580.

[73] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.

[74] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[75] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4290–4297.

[76] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, 2013.

[77] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray, "Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device," *IEEE Trans. on Visualization and Computer Graphics*, vol. 22, no. 11, 2015.

[78] M. N. Finean, W. Merkt, and I. Havoutis, "Predicted composite signed-distance fields for real-time motion planning in dynamic environments," in *Proc. of the Conf. on Automated Planning and Scheduling*, vol. 31, 2021, pp. 616–624.

[79] M. N. Finean, L. Petrović, W. Merkt, I. Marković, and I. Havoutis, "Motion planning in dynamic environments using context-aware human trajectory prediction," *J. of Robotics and Autonomous Systems*, vol. 166, p. 104450, 2023.

[80] B. Ivanovic, J. Harrison, and M. Pavone, "Expanding the deployment envelope of behavior prediction via adaptive meta-learning," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2023.

[81] M. King-Smith, P. Tsiotras, and F. Dellaert, "Simultaneous control and trajectory estimation for collision avoidance of autonomous robotic spacecraft systems," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 257–264.

[82] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[83] C.-Y. Yang, H.-W. Huang, W. Chai, Z. Jiang, and J.-N. Hwang, "Samurai: Adapting segment anything model for zero-shot visual tracking with motion-aware memory," 2024.

Jesse Morris is a 3rd year PhD student at the Australian Centre For Robotics (ACFR), supervised by Dr Viorela Ila. His thesis focuses on developing estimation frameworks for Dynamic SLAM. He received his bachelors degree in Mechatronic Engineering and Computer Science from the University of Sydney in 2022.

Yiduo Wang is a postdoctoral researcher in the ACFR, working with Dr Viorela Ila. He focuses on SLAM, reconstruction and navigation in dynamic environments. He holds a DPhil Engineering Science degree awarded by University of Oxford, where his research focused on large-scale reconstruction. He also holds an MRes Robotics degree awarded by UCL on combining SLAM with semantic segmentation for dynamic environments.

Mikolaj Kliniewski is a 2nd year PhD student at the Australian Centre For Robotics, supervised by Dr Viorela Ila. His thesis focuses on joint planning and estimation in dynamic environments. He received his Bachelor Honours degree in Computer Science from the University of Liverpool in 2023.

Viorela Ila received a Ph.D. in Information Technologies from the Universitat de Girona, Spain. She worked at the Institut de Robótica i Informàtica Industrial, Barcelona, and was awarded a MICINN/FULBRIGHT post-doctoral fellowship in 2009, joining Georgia Tech, USA. She joined the ROSACE project in LAAS-CNRS, France, in 2010, and was a Research Scientist at Brno University of Technology, Czech Republic (2012–2014) and a Research Fellow at the Australian National University (2015–2018). Currently a Senior Lecturer at the University of Sydney, her research focuses on robot vision, SLAM and 3D reconstruction, leveraging graphical models, optimization methods and information theory.