ObVi-SLAM: Long-Term Object-Visual SLAM

Amanda Adkins, Taijing Chen, and Joydeep Biswas

***Abstract*—Robots responsible for tasks over long time scales must be able to localize consistently and scalably amid geometric, viewpoint, and appearance changes. Existing visual SLAM approaches rely on low-level feature descriptors that are not robust to such environmental changes and result in large map sizes that scale poorly over long-term deployments. In contrast, object detections are robust to environmental variations and lead to more compact representations, but most object-based SLAM systems target short-term indoor deployments with close objects. In this letter, we introduce ObVi-SLAM to overcome these challenges by leveraging the best of both approaches. ObVi-SLAM uses low-level visual features for high-quality short-term visual odometry; and to ensure global, long-term consistency, ObVi-SLAM builds an uncertainty-aware long-term map of persistent objects and updates it after every deployment. By evaluating ObVi-SLAM on data from 16 deployment sessions spanning different weather and lighting conditions, we empirically show that ObVi-SLAM generates accurate localization estimates consistent over long time scales in spite of varying appearance conditions.**

***Index Terms*—SLAM, localization, semantic scene understanding.**

I. Introduction

MOBILE service robots need to be able to operate autonomously over long time periods. Existing visual SLAM approaches are susceptible to failures caused by environmental changes that occur over such time scales, including geometric changes from moving and movable objects, appearance changes from lighting and seasonal variation, and viewpoint differences. Further, visual SLAM systems generally use maps composed of visual feature points, which results in large maps that do not scale well with time.

Unlike classical visual feature extractors, object detectors are trained to be invariant to appearance and large-scale viewpoint changes. Object-based SLAM systems [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] leverage object detections to estimate landmarks that are similarly robust to such changes and provide a low-dimensional environment representation. However, most object-based SLAM systems target short-term, indoor trajectories in which objects are viewed from diverse perspectives, simplifying estimation.

In this work, we introduce ObVi-SLAM: an object-visual SLAM approach designed for long-term deployments. ObVi-SLAM, shown in Fig. 1, aims to generate accurate and consistent trajectory estimates for many deployment sessions over long time scales. To accomplish this objective, ObVi-SLAM utilizes a long-term map of the static objects in the target environment that reflects uncertainty of the current estimates. The use of object detectors trained to be invariant to different appearances enables ObVi-SLAM to be robust to appearance and viewpoint variation, while focusing on static objects provides resilience to movable entities. Forming the map with objects and their estimate uncertainty enables scalability over time and continuous improvement using new measurements from each deployment. Finally, use of low-level visual features tracked within individual deployment sessions enables high short-term accuracy. We summarize our contributions as follows:

i) a joint object-visual SLAM objective function for integrating long-term map information, object detections, and visual feature observations,
ii) a long-term scalable object front-end for initializing objects and associating incoming bounding boxes to existing objects, and
iii) a method for extracting a long-term map of the static objects in an environment with probabilistically-derived uncertainty estimates.

We empirically show that ObVi-SLAM can accurately and consistently estimate trajectories over long-term deployments using data collected outdoors across 16 deployment sessions over two months.

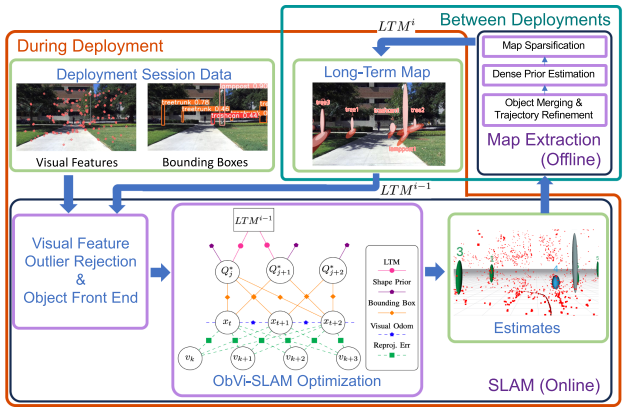


Fig. 1. ObVi-SLAM overview. In the factor graph, factors with solid lines are present for all optimizations, whereas use of factors with dashed lines is dependent on if the optimization is local or global.

II. Related Work

A. Object SLAM

Initial efforts in semantic object SLAM such as SLAM++ [11] used pre-existing models. Many later works either estimate

Manuscript received 25 September 2023; accepted 20 January 2024. Date of publication 7 February 2024; date of current version 14 February 2024. This letter was recommended for publication by Associate Editor J. Stueckler and Editor S. Behnke upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant GRFP DGE-2137420 and Grant CAREER-2046955, and in part by Amazon Lab126. *(Corresponding author: Amanda Adkins.)*

The authors are with the Department of Computer Science, University of Texas at Austin, Austin, TX 78712 USA (e-mail: aaadkins@cs.utexas.edu; taijing@cs.utexas.edu; joydeepb@cs.utexas.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3363534, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3363534

detailed geometry for each instance [12], [13] or use coarse, general models. For computational efficiency, ObVi-SLAM uses coarse models, thus we focus on the latter type. CubeSLAM [1] introduces an object-based SLAM algorithm that represents objects as cuboids and leverages relationships between low-level visual features, objects, and camera poses for estimation. Ellipsoids are also an increasingly popular representation for objects in SLAM. QuadricSLAM [2] is among the first to use ellipsoids to represent objects, but requires known data associations. Various extensions have enhanced object SLAM with ellipsoids, addressing aspects such as fast object initialization and constraints to help with limited viewpoint diversity in ROSHAN [3]; trajectory reinitialization using objects in OA-SLAM [4]; constraints for object arrangement, symmetry, and scale in SO-SLAM [5]; planar and point-cloud based factors for object estimation [14]; and providing scale to monocular SLAM using object prior dimensions [6]. Others focused on object data association or initialization, with ensemble data association [7]; initialization for straight trajectories [8]; object initialization and association for outdoor SLAM [9]; and few-frame, limited viewpoint association and initialization [10]. In contrast to the proposed approach, consideration of long-term SLAM is limited in these works: OA-SLAM is the only approach that addresses any aspects of long-term localization and does so with a fixed map with no uncertainty estimation.

B. Long-Term Map Management

SLAM approaches used for long-term robot deployments need to scale over time. Many approaches do not support multiple mapping sessions. Others [15], [16] support multi-session mapping, but do not address scaling; another category fixes the map after initial data collection, addressing scaling but leaving localization susceptible to initial errors and environmental changes. Some approaches, such as POCD [17], maintain a fixed map but use change detection to determine when to update map regions. Other approaches prune or summarize the oldest data, retaining only the most informative or recent data. Nashed and Biswas [18] generate an uncertainty-aware vector-based map of long-term features using lidar data from multiple deployments. A common technique for reducing map size used by ObVi-SLAM and numerous prior works is marginalization and sparsification. Vallvé et al. [19] proposed a method called factor descent for general SLAM marginalization and sparsification. Hsiung et al. [20] and Wilbers et al. [21] both addressed the problem of online marginalization and sparsification for sliding window optimization; the former uses an information-theoretic approach applied to visual-inertial odometry, whereas the latter generates a single global prior for optimal sparsity. Zhao et al. [22] focused on long-term lidar localization, using marginalization and sparsification to remove old submaps from a pose-graph. Unlike these approaches, ObVi-SLAM specifically uses marginalization for object-based SLAM.

C. Long-Term Robust SLAM

For robustness over long time scales, SLAM must tolerate geometric, appearance, and viewpoint changes. Most approaches that aim to be robust to geometric changes arising from movable and moving entities aim to identify observations that may correspond to such objects, either using observed motion or semantic class. Some of these approaches, such as DOT [23], discard observations from these features, whereas others like CubeSLAM [1] and DynaSLAM II [24] use these objects for local corrections, tracking moving objects.

Engineered feature representations, such as ORB features, can tolerate minor viewpoint and appearance changes. Recently, deep-learning has been applied to improve robustness to such changes. Gridseth et al. [25] developed a network to select sparse keypoints, with a differentiable pose estimation framework. Chen and Barfoot [26] similarly aimed to generate lighting-invariant keypoints in a self-supervised manner. Deja-Vu [27] generates dense feature maps robust to seasonal changes using pairs of images captured in the same location in different conditions. Learning-based methods like these, however, are highly dependent on having training data, often requiring labels or ground-truth, that matches the characteristics of the target environment. ObVi-SLAM is distinct from these approaches in that it uses objects to obtain global position information for long-term SLAM.

III. System Overview and Preliminaries

A. Approach Overview

There are 2 phases in ObVi-SLAM: SLAM and long-term map extraction. SLAM runs online during deployment, estimating the robot's trajectory and the map of visual features and objects. Between deployments, estimates are refined and long-term map extraction uses the SLAM results to update the long-term map for use in subsequent sessions. The long-term map is fixed for each deployment and updated between sessions. Fig. 1 shows the full ObVi-SLAM system.

B. Representation, Notation, and Assumptions

We first define notation and representations for variables and note key assumptions. The following notation is used:

- $i \in [1, L]$ - deployment session index.
- $x_t^i \in SE(3)$ - pose at time t in session i; $t \in [1, T^i]$.
- $x_t^{i\prime} \in SE(3)$ - estimate for x_t^i after the most recent optimization; $t \in [1, T^i]$.
- $v_k^i \in \mathbb{R}^3$ - kth visual feature in session i; $k \in [1, V^i]$.
- $s_m^i \in \mathbb{R}^2$ - mth visual feature observation in session i: pixel at which feature was observed; $m \in [1, M^i]$.
- $B_n^i \in \mathbb{R}^4$ - nth object observation in session i represented as a bounding box parameterized by the minimum and maximum x and y pixels; $n \in [1, N^i]$.
- LTM^i - Long-term map summarizing information gathered through session i.
- Q_j^{i*} - Object j in session i, as an ellipsoid; $j \in [1, J^i]$.
- c_j^i - Semantic class of object Q_j^{i*}.

Unless otherwise noted, variables correspond to deployment session i. Object notation follows conventions in [3] and [2].

There are two supported parameterizations of the ellipsoid: upright and full-DOF. In both cases, there are three position parameters and three dimension parameters. The upright parameterization represents orientation with a single parameter for yaw; the full-DOF parameterization uses three parameters for an $SO(3)$ orientation. Parameterization choice depends on possible object configurations in the target environment. We use the dual quadric form for each ellipsoid, which defines the ellipsoid using the relationship to its tangent planes. In the dual form, an ellipsoid Q_j^* is represented by a 4x4 matrix. For details

on this representation and how the parameters are encoded in Q^*_j, refer to [2] and [3].

The only sensor data required for ObVi-SLAM are camera images; ObVi-SLAM supports arbitrary camera configurations, including monocular and stereo, assuming known relative camera poses. In our formulation, we assume that the robot's pose is combined with the appropriate relative camera pose when assessing an observation likelihood. We also assume that the robot's pose at the beginning of each deployment session is known. Finally, the system should be configured to use object detections of the static classes for the target environment and approximate mean and variation for the dimensions of each semantic class should be given.

ObVi-SLAM's tolerance to long-term changes relies on observable static objects; without such detections, it is effectively classical visual SLAM. ObVi-SLAM is robust to objects that move between sessions, as their corresponding visual features satisfy the static world assumption for the duration of their observability and are not kept in the map. However, like classical visual SLAM, accuracy may degrade when there is a high density of actively moving objects.

IV. Online SLAM Formulation

A. SLAM Factor Graph

During SLAM operation for deployment session i, we aim to find the optimal estimates for the visual features, $v_{1:V}$; objects, $Q^*_{1:J}$; and robot poses, $x_{1:T}$. Optimization inputs are the visual feature observations, $s_{1:M}$; object detections, $B_{1:N}$; prior mean and covariances for the dimensions of classes of interest; and, if it exists, the long-term map from a previous session, LTM^{i-1}. We use two different optimization modes: local adjustment and global adjustment. Local adjustment optimizes the most recent poses and observations to generate high-quality visual odometry estimates. For local adjustment, we use the following model for belief:

$$\text{Bel}(x_{1:T}, v_{1:V}, Q^*_{1:J}) \propto \underbrace{p(s_{1:M}|x_{1:T}, v_{1:V})}_{\text{Reprojection Error}} \underbrace{p(B_{1:N}|Q^*_{1:J}, x_{1:T})}_{\text{Bounding Box Error}} \underbrace{p(Q^*_{1:J}|c_{1:J})}_{\text{Semantic Shape Prior}} \underbrace{p(Q^*_{1:J}|LTM^{i-1})}_{\text{Long-term Map Prior}}. \tag{1}$$

Global adjustment aims to shift the estimates generated by the local optimization for improved global accuracy by using data from the entire trajectory. For computational efficiency, we use a simplified form of the belief, given by

$$\text{Bel}(x_{1:T}, v_{1:V}, Q^*_{1:J}) \propto \underbrace{p(x_{1:T})}_{\text{Visual Odometry Error}} \underbrace{p(B_{1:N}|Q^*_{1:J}, x_{1:T})}_{\text{Bounding Box Error}} \underbrace{p(Q^*_{1:J}|c_{1:J})}_{\text{Semantic Shape Prior}} \underbrace{p(Q^*_{1:J}|LTM^{i-1})}_{\text{Long-term Map Prior}}. \tag{2}$$

A factor graph reflecting these models is shown in Fig. 1. Given the appropriate form of the belief, ObVi-SLAM then uses a nonlinear least squares optimizer to find the estimates that satisfy the following optimization objective:

$$x^*_{1:T}, v^*_{1:V}, Q^*_{1:J} = \underset{x_{1:T}, v_{1:V}, Q^*_{1:J}}{\text{argmax}} \text{Bel}(x_{1:T}, v_{1:V}, Q^*_{1:J}) \tag{3}$$

$$\equiv \underset{x_{1:T}, v_{1:V}, Q^*_{1:J}}{\text{argmin}} [-\log \text{Bel}(x_{1:T}, v_{1:V}, Q^*_{1:J})]. \tag{4}$$

The factors in (1) and (2) are detailed in the next subsections.

1) Reprojection Factor: The reprojection factor uses low-level visual features to generate a smooth trajectory estimate across local frames. Consistent with previous indirect visual SLAM approaches, ObVi-SLAM estimates the 3D locations of features detected in images by matching detections across frames. It adjusts the robot's trajectory and current visual feature position estimates to minimize the distance between the detected features and reprojections of features onto the images. The reprojection factor term in our optimization objective is the sum of terms for each feature detection:

$$-\log p(s_{1:M}|x_{1:T}, v_{1:V}) = \sum_{m=1}^{M} -\log p(s_m|x_{s_m}, v_{s_m}) \tag{5}$$

$$= \frac{1}{2}\sum_{m=1}^{M} ||\Pi_p(x_{s_m}, v_{s_m}) - s_m||^2_{\Sigma_{s_m}}, \tag{6}$$

where x_{s_m} and v_{s_m} are the robot's pose and feature for the detection, $\Pi_p(\cdot,\cdot)$ is the projection function giving the pixel location of a 3D point, and Σ_{s_m} is the detection covariance.

2) Visual Odometry Factor: In the global trajectory optimization, to improve computation time and reduce the impact of local minima, we use visual odometry factors instead of reprojection error factors. This term penalizes deviations from the visual odometry estimates between pairs of subsequent poses obtained from the result of local optimization using reprojection factors. This has the form

$$-\log p(x_{1:T}) = \sum_{t=2}^{T} -\log p(x_t|x_{t-1}, x'_t, x'_{t-1}) \tag{7}$$

$$= \sum_{t=2}^{T} \frac{1}{2}|| \left((x_t \ominus x_{t-1}) \ominus (x'_t \ominus x'_{t-1})\right) ||^2_{\Sigma_t}, \tag{8}$$

where x'_t is the estimate for x_t from the previous optimization, the $\ominus$ operator gives the difference between two poses as in [2], and Σ_t is the covariance for the pose deviation. Σ_t is scaled with the magnitude of $x'_t \ominus x'_{t-1}$.

3) Bounding Box Factor: The bounding box factor aims to update object estimates such that edges of the detected bounding boxes for an object are tangent to the ellipse generated by projecting the ellipsoid onto the camera plane. This factor enables ObVi-SLAM to relate the current trajectory to the long-term map and update object estimates with new information. The bounding box factor contains a term for each bounding box detection as follows:

$$-\log p(B_{1:N}|Q^*_{1:J}, x_{1:T}) = \sum_{n=1}^{N} -\log p(B_n|Q^*_{B_n}, x_{B_n})$$

$$= \frac{1}{2}\sum_{B_n \in s_O} ||\Pi_B(x_{B_n}, Q^*_{B_n}) - B_n||^2_{\Sigma_{B_n}}, \tag{9}$$

where $Q^*_{B_b}$ and x_{B_n} are the object and robot's pose for the detection, $\Pi_B(x_{B_n}, Q^*_{B_n})$ is the projection operator that generates the bounding box for a given ellipsoid observed from a given pose, and Σ_{B_n} gives the uncertainty of the object detection. For

full details on ellipsoid projection, refer to our website or [6]. To handle occlusions resulting from the image boundary, we inflate the entries of Σ_{B_n} corresponding to the occluded edge when a detected bounding box lies near the image boundary.

It should be noted that when either the x or y axes of the camera intersect an ellipsoid, the generated conic is not a proper ellipse,[1] resulting in imaginary bounding box values. In these cases, we set the error to a fixed value.

4) Semantic Shape Prior Factor: The semantic shape prior is important for constraining an object's dimensions in cases of limited viewpoint diversity where the object's depth and dimension along the camera ray are under-constrained. This factor penalizes deviations from the mean dimension of an object's semantic class. The semantic shape prior is a sum of terms, each penalizing a single object, with the form

$$-\log p(Q^*_{1:J}|c_{1:J}) = \sum_{j=1}^{J} ||m_{\text{shape}}(c_j) - d(Q^*_j)||^2_{\Sigma_{c_j}}, \quad (10)$$

where $m_{\text{shape}}(\cdot)$ is a function giving the mean dimension for an object of the given semantic class,[2] $d(\cdot)$ gives the dimensions of an object, and Σ_{c_j} reflects the variance of the dimensions of an object of class c_j. Ok et al. introduced this term in their paper [3], where further details can be found.

5) Long-Term Map Prior Factor: The long-term map prior factor incorporates information from previous sessions in our object estimation. This serves as a prior for objects that have been previously observed, so that their latest estimates appropriately balance previous knowledge and uncertainty with new observations. The ObVi-SLAM framework is generic to any factorization relating objects to each other and to the map frame. The generic form for this factor is

$$p(Q^*_{1:J}|LTM^{i-1}) = \prod_{S_k \in LTM^{i-1}} N(f_{S_k}(Q^*_{1:J^{i-1}})|\mu_{S_k}, \Sigma_{S_k}), \quad (11)$$

where S_k is a component factor of the long-term map, $f_{S_k}(\cdot)$ is a function operating on a subset of the previously observed objects, and μ_{S_k} and Σ_{S_k} are the mean and covariance for S_k. The current ObVi-SLAM formulation uses an independent normal assumption for each previous object, so the long-term map factor described generally in (11) has the form

$$-\log p(Q^*_{1:J}|LTM^{i-1}) = \sum_{j=1}^{J^{i-1}} \frac{1}{2} ||\text{v}(Q^*_j) - \mu_{S_j}||^2_{\Sigma_{S_j}}, \quad (12)$$

where $\text{v}(\cdot)$ is a function that generates a vector of the parameters for object Q^*_j, and μ_{S_j} and Σ_{S_j} are the object estimate mean and uncertainty based on previous sessions.

B. Front-End

To use this optimization, a front-end is needed to provide initial estimates and associations relating observations over time. Visual feature associations and initial estimates are obtained using the tracking component of ORB-SLAM2 [28], which generates feature tracks based on ORB descriptors.

Our object front-end maintains a set of pending objects, adding them to the map after accumulating sufficient observations exceeding a minimum confidence. This guards against false positive object detections and prevents poorly-informed object estimates from influencing the trajectory estimates. Object initialization uses identity for orientation and the mean dimensions for the semantic class. The object's initial position is along the ray through the bounding box center and the depth is the distance at which the mean height yields the detected bounding box height.

Associating incoming bounding boxes to past observations is a two-step process. We first perform local appearance-based matching using ORB features in the bounding boxes. If there are no matches for a bounding box, a new pending object is created. ObVi-SLAM can thus generate high-quality local object estimates using (1), but optimizing only pending objects. After attaining a good local estimate, ObVi-SLAM performs geometric matching against older objects. If an older object's center is close, the pending object's detections are merged into the older object and the pending object is removed. If there are no existing objects that match the pending estimate, a new object is created. This method allows ObVi-SLAM's object front-end to scale with time, as only the features for the most recent bounding boxes are kept, avoiding long-term aggregation of object appearance data.

V. Long-Term Map Extraction

After SLAM for a deployment session, ObVi-SLAM refines results offline before extracting a long-term map for subsequent sessions. A final optimization runs over the full trajectory, first using the model in (2), and then further refining with (1). This refinement is reserved for the post-processing stage due to computational complexity.

ObVi-SLAM then checks if any of the objects should be merged into a single object, as imperfect data association can yield multiple objects for the same physical entity. To address this, ObVi-SLAM searches for pairs of objects that have the same semantic class and are in close proximity, then merges the observations. Global refinement and object merging are repeated until no pairs of objects can be merged.

Following refinement, ObVi-SLAM summarizes the information obtained into a space-efficient long-term object map containing information only for objects' poses and dimensions. Inputs to the long-term map extraction are the estimates $\{Q^{i*}_{1:J^i}, x^i_{1:T^i}, v^i_{1:V^i}\}$ and associations between observations and estimates. Long-term map extraction has two phases: dense prior computation and map sparsification.

A. Dense Prior Computation

The dense prior is a marginal distribution over the objects, with visual features and robot poses marginalized out. As we model the full optimization problem with a normal distribution, the dense long-term map thus takes the form

$$p\left(Q^*_{1:J^i}|LTM^{i^D}\right) = N(Q^*_{1:J^i}|\mu_D, \Sigma_D)$$
$$= N\left(\begin{bmatrix} Q^*_1 \\ \vdots \\ Q^*_{J_i} \end{bmatrix} \middle| \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{J_i} \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \dots & \Sigma_{1J_i} \\ \vdots & & \vdots \\ \Sigma^T_{1J_i} & \dots & \Sigma_{J_iJ_i} \end{bmatrix}\right). \quad (13)$$

[1] If both the x and y axes intersect the ellipsoid, the camera lies within the ellipsoid and the resulting conic is an imaginary ellipse; otherwise, the projection is a hyperbola, with one branch coming from behind the camera. Future work could improve handling of hyperbolic projections.

[2] $m_{\text{shape}}(\cdot)$ is drawn from known object distributions, which could come from human knowledge or existing object datasets.

μ_D and Σ_D are obtained with an optimization over the factors to summarize in the long-term map and the Markov blanket of the states to remove. In our case, we marginalize over the visual feature estimates, $v_{1:V^i}$, and robot pose estimates, $x_{1:T^i}$, obtained in session i. Shape priors are repeated across sessions, so these factors are omitted. Using properties of marginals for a normal distribution, μ_D and Σ_D are formed from blocks corresponding to objects from the full mean and covariance. Thus, μ_D is given by

$$\mu_D = \underset{Q^*_{1:J}}{\operatorname{argmax}} \underbrace{p(s_{1:M}|x_{1:T}, v_{1:V})}_{\text{Reprojection Error}} \underbrace{p(B_{1:N}|Q^*_{1:N}, x_{1:T})}_{\text{Bounding Box Error}} \underbrace{p(Q^*_{1:J}|LTM^{i-1})}_{\text{Long-term Map Prior}}. \quad (14)$$

For a least squares optimization problem given by $y^* = \operatorname{argmin} ||f(y) - z||^2$, where z is a measurement and $f(y)$ is a predicted measurement given the state y, the covariance estimate Σ_y is given by $\Sigma_y = (J^T(y^*)J(y^*))^{-1}$, where $J(y^*)$ is the Jacobian of f evaluated at y^*. Thus, Σ_D is obtained using the Jacobian of the optimization expressed in (14), evaluated at μ_D, taking the blocks of the covariance matrix that correspond to relationships between static objects.

B. Map Sparsification

The dense prior above would increase the computational complexity of optimization in a subsequent session, as modern optimization libraries rely on sparsity. To address this, we aim to find a sparsified long-term map that approximates the dense prior but has fewer correlations between objects in the map. Sparsification first requires a design step to determine the factor topology. The generic form for the sparsified long-term map is shown in (11). Choosing the factor topology can be seen as selecting the functions f_{S_k} that comprise this long-term map representation. Once the factor topology is determined, the parameters μ_{S_k} and Σ_{S_k} need to be found. To make the sparse long-term map prior as close as possible to the dense prior, μ_{S_k} and Σ_{S_k} are identified by finding the values that minimize the KL divergence between the dense prior $N(Q^*|\mu_D, \Sigma_D)$, and the sparse long-term map prior in (11). After μ_{S_k} and Σ_{S_k} are identified, the values for Σ_{S_k} may suggest that a component factor of the sparse long-term map does not carry significant information, in which case minimally meaningful components f_{S_k} can be removed from the topology and the parameter identification can be repeated, encouraging further sparsity in LTM^i.

As noted above, the current version of ObVi-SLAM uses the sparse prior form with independently, normally-distributed ellipsoids. The long-term map prior thus takes the form

$$p(Q^*_{1:J^i}|LTM^i) = N(Q^*_{1:J^i}|\mu_S, \Sigma_S) \quad (15)$$

$$= \prod_{j=1}^{J^i} N(Q^*_j|\mu_{S_j}, \Sigma_{S_j}). \quad (16)$$

Using this topology, the KL divergence between the dense long-term map and sparse long-term map is

$$D_{KL}(p(Q^*_{1:J^i}|LTM^{i^D})||p(Q^*_{1:J^i}|LTM^i)) = \frac{1}{2}\left(\ln\frac{|\Sigma_S|}{|\Sigma_D|} - d + (\mu_S - \mu_D)^T\Sigma_S^{-1}(\mu_S - \mu_D) + \operatorname{Tr}(\Sigma_S^{-1}\Sigma_D)\right), \quad (17)$$

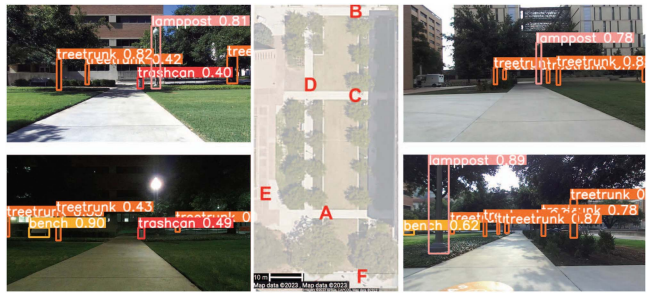


Fig. 2. Test environment: appearance change at waypoint A; satellite view with waypoint labels; waypoint D from different perspectives.

with Σ_S constrained to be a block diagonal matrix with entries Σ_{S_j} and d being the dimension of the normal distribution. Differentiating gives the optimal μ_S equal to μ_D and the optimal Σ_S as a block diagonal matrix with the corresponding entries from in Σ_D, as is consistent with independently distributed ellipsoids. More specifically, this gives μ_{S_j} as the maximum likelihood estimate for the jth ellipsoid for (14) and Σ_{S_j} as the covariance representing the uncertainty after optimization for the jth ellipsoid estimate.

VI. Experimental Results

We aim to answer the following questions in our evaluation: (i) How globally accurate are individual trajectory estimates generated by ObVi-SLAM? (ii) How consistent are ObVi-SLAM localization estimates with respect to a global frame under diverse appearances? (iii) How accurate are object estimates generated by ObVi-SLAM? (iv) How space efficient are ObVi-SLAM maps? (v) What is the impact of each component of the formulation for ObVi-SLAM on its performance? Parameters and code for ObVi-SLAM and our evaluation pipeline can be found in our github repository.[3]

A. Experiment Setup and Implementation Details

To evaluate our approach, we collected 16 trajectories over two months in an outdoor campus environment, shown in Fig. 2, with different lighting conditions, covering varying paths to capture different viewpoints. In each trajectory, the robot operator started and ended the robot at the same location, visiting a subset of six waypoints located consistently across the trajectories. ObVi-SLAM is ignorant of these waypoints: they are only used for evaluation. We used YOLOv5 [30] with a custom model for object detections of tree trunks, lampposts, benches, and trash cans.

For our evaluations, we used the upright parameterization of objects for ObVi-SLAM. We compared against ORB-SLAM3 [15], a visual feature-based multi-session SLAM approach; OA-SLAM [4], a joint object-visual SLAM approach that uses objects for relocalization; and DROID-SLAM [31], a learning-based approach. For all approaches, we used stereo images from a Zed2i on a Clearpath Jackal. DROID-SLAM does not support map saving, so each trajectory was evaluated individually. For ORB-SLAM3, maximum size for a map data structure was reached after the seventh trajectory, so the first

[3]Code + website: https://github.com/ut-amrl/ObVi-SLAM; Code uses Ceres Solver [29].

TABLE I
ABSOLUTE TRAJECTORY ERROR FOR TRANSLATION AND ORIENTATION FOR OBVI-SLAM (OB), OA-SLAM (OA), ORB-SLAM3 (ORB), AND DROID-SLAM (DS)

	Translation ATE (m)				Orientation ATE (deg)			
Traj	Ob	OA	ORB	DS	Ob	OA	ORB	DS
1	0.69	**0.31**	*0.67*	1.39	2.6	**1.6**	2.0	*1.9*
2	**0.71**	3.32	*1.2*	4.76	*2.3*	**1.7**	2.8	2.4
3	**1.32**	3.99	*2.36*	4.92	*2.6*	19.1	12.0	**2.2**
4	**0.26**	*0.6*	1.7	3.96	**1.6**	*1.8*	2.2	13.6
5	**0.34**	*1.11*	*1.11*	3.86	1.8	**1.1**	*1.7*	2.5
6	**0.38**	1.31	*0.98*	4.1	*1.8*	2.0	**1.7**	2.2
7	**0.27**	1.21	*0.83*	4.21	1.9	**1.3**	*1.7*	2.2
8	**0.42**	2.66	*1.8*	3.59	**2.1**	4.4	3.2	*2.3*
9	**0.39**	9.14	7.27	*4.79*	**2.0**	42.7	46.7	*2.2*
10	**0.29**	*0.96*	6.82	3.12	*1.7*	**1.1**	6.7	3.3
11	**0.28**	0.6	*0.5*	2.05	2.3	**1.6**	*1.6*	11.1
12	**0.26**	0.94	*0.67*	7.69	1.9	**1.4**	*1.7*	2.5
13	**0.54**	1.0	*0.91*	4.84	6.4	**1.0**	*1.8*	3.1
14	**0.63**	6.44	*2.29*	6.82	**2.6**	14.9	4.9	*2.7*
15	**0.98**	10.94	*1.38*	4.55	*2.2*	55.3	**2.0**	2.3
16	**0.13**	4.48	*1.35*	3.42	**1.6**	22.6	2.9	*2.2*
Overall	**0.55**	4.21	*2.91*	4.49	**2.7**	18.4	12.3	*4.9*

Best are bold and second-best are italic.

seven were run in sequence, with the remaining nine run individually starting with the seventh map. OA-SLAM was run with the same object detections as ObVi-SLAM and maps passed between sessions were aggregated across the 16 trajectories. Ground truth was obtained by running LeGO-LOAM [32] with an Ouster OS1 lidar, aligning individual trajectories using the waypoints. The average speed of the robot was 0.76 m/s, and due to motion-based keyframing, this lead to 2.98 keyframes per second on average. The online component of ObVi-SLAM processed each frame in an average of 222 ms, enabling real-time operation. Additional timing data are on our website.

B. Trajectory Accuracy

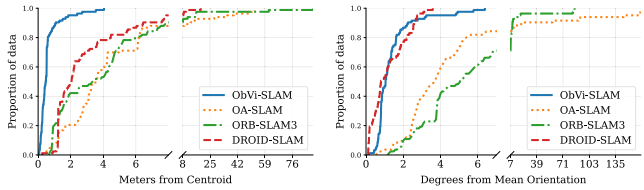
To understand individual trajectory accuracy, we computed translational and rotational absolute trajectory error (ATE) for each approach. We first aligned each trajectory to the respective reference trajectory to mitigate against early estimation error skewing results. ATE results are shown in Table I.[4] ObVi-SLAM has competitive or lower rotational error compared to the other approaches. Comparison approaches sometimes have lower error, but ObVi-SLAM is the only approach with less than 7 degrees of average rotational error for all trajectories. For translational error, ObVi-SLAM outperforms comparision approaches on all but one trajectory.

C. Localization Consistency Under Varying Conditions

Long-term SLAM approaches must be able to generate estimates for key locations that are consistent over time. We thus assess long-term robustness using consistency of estimates for the six waypoints by measuring the deviation for each estimate from the average for the respective waypoint.

Fig. 3 shows the cumulative distribution functions (CDFs) for the position and orientation estimate consistency. For rotation estimate consistency, ObVi-SLAM and DROID-SLAM [31] have notably better performance than ORB-SLAM3 [15] and OA-SLAM [4], with DROID-SLAM slightly outperforming

[4]As with [33], we measure accuracy using ATE as long-term localization focuses on global accuracy. Relative pose error data are on our website.



(a) CDF of position deviation. (b) CDF of orientation deviation.

Fig. 3. Estimate consistency for ObVi-SLAM and comparison approaches. An optimal algorithm would quickly rise to 1.

ObVi-SLAM. However, ObVi-SLAM notably outperforms all comparison approaches on position consistency, having no waypoint position estimates more than 4.05 meters from the respective average waypoint position.

The trajectories and waypoints estimated by LeGO-LOAM [32] (reference), ObVi-SLAM, ORB-SLAM3 [15], OA-SLAM [4], and DROID-SLAM [31] are shown in Fig. 4. ORB-SLAM3 lost localization in one trajectory and the remaining trajectories have rotational drift, caused by features not being recognizable under viewpoint and lighting changes. OA-SLAM suffered relocalization failures that present as jumps in the trajectories, resulting from noisy object estimation, and DROID-SLAM has good orientation consistency, but suffers from scale inconsistency, despite use of stereo data. ObVi-SLAM, by contrast, has trajectories that match the reference well and waypoints are closely colocated.

D. Object Accuracy

To understand ObVi-SLAM's object estimation accuracy, we assess four different metrics using the map result at the end of each trajectory:

1) position accuracy: measures the distance between estimate and ground truth object centers;
2) volume intersection over union (IoU): evaluates volumetric accuracy by computing the intersection of estimated and ground truth objects, divided by their union – an IoU of 1.0 indicates a perfect estimate;
3) estimated objects per ground truth: ratio of number of estimated objects to ground truth objects, providing insight into the generation of duplicate and spurious objects; and
4) recall: measures how many ground truth objects were estimated.

72 ground truth objects were manually annotated in a lidar-SLAM point cloud. We aligned estimates with ground truth frames and matched objects by finding the closest ground truth object with the same semantic class, allowing for multiple matches per ground truth object. Position deviation and volume IoU together assess geometric accuracy, while the remaining metrics examine object instantiation false positives and negatives.

Fig. 5 shows these metrics for ObVi-SLAM and OA-SLAM [4], as the other approaches do not estimate objects. OA-SLAM initially exhibits better median position deviation compared to ObVi-SLAM. However, as additional trajectories are considered, OA-SLAM's positional accuracy degrades. ObVi-SLAM maintains position estimate accuracy over time and exhibits notably better IoU for object estimates. This indicates that OA-SLAM's object estimates are less volumetrically

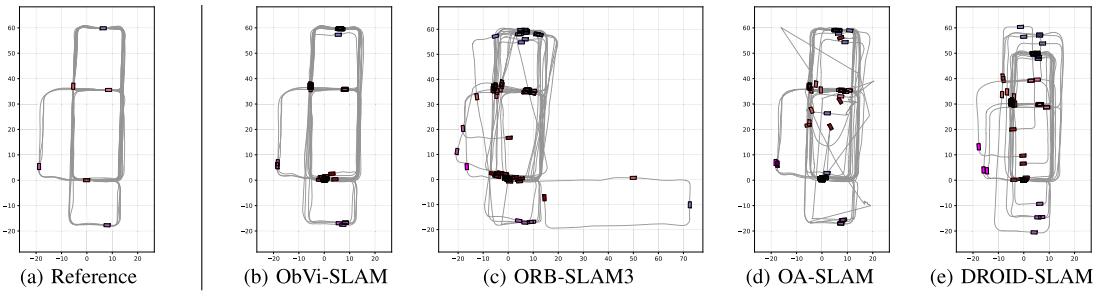


Fig. 4. 16 trajectories through test environment, shown in meters, as estimated by the approaches with highlighted pink/purple waypoints. Performance of an approach is good when all estimates for a given waypoint are colocated.

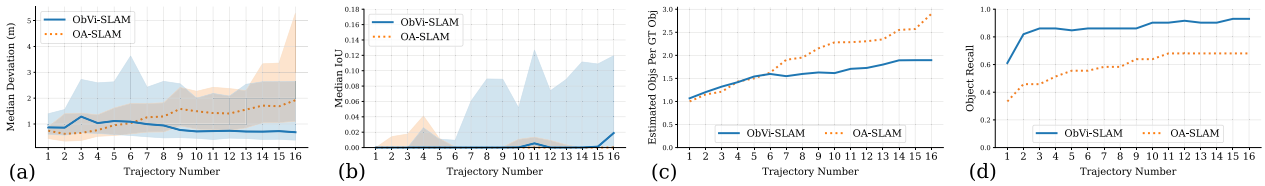
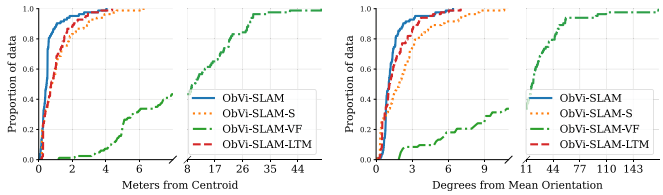


Fig. 5. Object metrics for ObVi-SLAM and OA-SLAM. (a) Lower quartile, median, and upper quartile object center error. (b) Lower quartile, median, and upper quartile IoU between estimates and ground truth. (c) Estimated objects per ground truth object. (d) Recall.

TABLE II
FILE SIZE (F), NUMBER OF VISUAL FEATURES (V) AND NUMBER OF OBJECTS (O) FOR MAPS AFTER SPECIFIED TRAJECTORIES

	ObVi-SLAM		ORB-SLAM3		OA-SLAM		
Traj	F	O	F	V	F	V	O
1	**31 KB**	47	165 MB	37K	1.4 MB	26K	22
7	**62 KB**	96	1.3 GB	207K	8.6 MB	161K	97
16	**80 KB**	127	-		17 MB	308K	233



(a) CDF of position deviation. (b) CDF of orientation deviation.

Fig. 6. Position and orientation consistency for ObVi-SLAM and ablations. An optimal algorithm would quickly rise to 1.

accurate. It should be noted most estimated objects are tall and thin, therefore a small amount of position error can lead to no overlap between estimates and ground truth, explaining the relatively low IoU for both approaches. In addition, ObVi-SLAM has better rates of object estimation than OA-SLAM, producing less duplicate estimates and generating estimates for more objects than OA-SLAM. The higher object recall also explains in part the initial higher upper quartile position error: ObVi-SLAM is more aggressive in generating estimates for objects, initially leading to some less accurate estimates before sufficient data is obtained, but resulting in fewer missed objects overall.

E. Map Space Efficiency

To assess the space efficiency of the maps, we evaluated the map files' size after trajectories 1, 7, and 16, shown in Table II. While the exact size is reflective of implementation, the order of magnitude illustrates scalability, which has implications for memory and compute. DROID-SLAM [31] does not enable map loading, so it is excluded. In addition, as ORB-SLAM3 [15] could not run more than seven trajectories sequentially, the map after trajectory 7 is the last attainable. From these map sizes, it is clear that ObVi-SLAM's object-only map affords a much more compact and scalable representation compared to ORB-SLAM3 and OA-SLAM [4].

F. Ablations

In addition to comparing against other approaches, we assess the performance of ObVi-SLAM when individual factors are removed to understand their importance. These are abbreviated as follows: ObVi-SLAM-S has the shape prior removed; ObVi-SLAM-VF does not include the visual feature factors; and ObVi-SLAM-LTM removes the long-term map, resulting in a sequence of individually-evaluated trajectories. We did not assess ObVi-SLAM without objects, as this is equivalent to a traditional visual SLAM approach.

The localization consistency results for ObVi-SLAM compared to the ablations is shown in Fig. 6. ObVi-SLAM-VF is significantly worse than all other variants, indicating the importance of visual features, with full ObVi-SLAM outperforming the other variants. We provide ATE and object metrics for the ablated versions in our github repository.[3]

VII. CONCLUSION AND FUTURE WORK

This paper introduces ObVi-SLAM, a long-term SLAM approach that combines the benefits of visual features and object detections to achieve scalable, long-term consistent localization

amid lighting and viewpoint changes. We demonstrate that ObVi-SLAM achieves superior performance in single-trajectory and multi-trajectory localization compared to existing SLAM approaches on 16 trajectories collected over two months with significant lighting variations.

There are several directions that would build upon developments introduced in ObVi-SLAM. A learned long-term appearance descriptor for objects could improve robustness of inter-session object association. Further, integrating object-based change detection could aid in removing stale data from objects incorrectly marked static. In addition, more complex factor topologies for the long-term map could be explored. Finally, ObVi-SLAM could be extended with loop closure or initial localization based on the long-term object map.

Acknowledgment

Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object SLAM," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 925–938, Aug. 2019.

[2] L. Nicholson, M. Milford, and N. Sünderhauf, "QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM," *IEEE Robot. Automat. Lett.*, vol. 4, no. 1, pp. 1–8, Jan. 2019.

[3] K. Ok, K. Liu, K. Frey, J. P. How, and N. Roy, "Robust object-based SLAM for high-speed autonomous navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 669–675.

[4] M. Zins, G. Simon, and M.-O. Berger, "OA-SLAM: Leveraging objects for camera relocalization in visual SLAM," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2022, pp. 720–728.

[5] Z. Liao, Y. Hu, J. Zhang, X. Qi, X. Zhang, and W. Wang, "SO-SLAM: Semantic object SLAM with scale proportional and symmetrical texture constraints," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4008–4015, Apr. 2022.

[6] S. Song, J. Zhao, T. Feng, C. Ye, and L. Xiong, "Scale estimation with dual quadrics for monocular object SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 1374–1381.

[7] Y. Wu, Y. Zhang, D. Zhu, Y. Feng, S. Coleman, and D. Kerr, "EAO-SLAM: Monocular semi-dense object SLAM based on ensemble data association," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 4966–4973.

[8] S. Chen et al., "Robust dual quadric initialization for forward-translating camera movements," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4712–4719, Jul. 2021.

[9] R. Tian et al., "Object SLAM with robust quadric initialization and mapping for dynamic outdoors," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 10, pp. 11080–11095, Oct. 2023.

[10] Z. Cao et al., "Object-aware SLAM based on efficient quadric initialization and joint data association," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9802–9809, Oct. 2022.

[11] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1352–1359.

[12] M. Grinvald et al., "Volumetric instance-aware semantic mapping and 3D object discovery," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 3037–3044, Jul. 2019.

[13] L. Schmid et al., "Panoptic Multi-TSDFs: A flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 8018–8024.

[14] M. Hosseinzadeh, K. Li, Y. Latif, and I. Reid, "Real-time monocular object-model aware sparse SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 7123–7129.

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

[16] T. Schneider et al., "Maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1418–1425, Jul. 2018.

[17] J. Qian, V. Chatrath, J. Yang, J. Servos, A. Schoellig, and S. L. Waslander, "POCD: Probabilistic object-level change detection and volumetric mapping in semi-static scenes," in *Proc. Int. Conf. Robot. Sci. Syst.*, 2022. [Online]. Available: https://www.roboticsproceedings.org/rss18/index.html

[18] S. Nashed and J. Biswas, "Curating long-term vector maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4643–4648.

[19] J. Vallvé, J. Solá, and J. Andrade-Cetto, "Graph SLAM sparsification with populated topologies using factor descent optimization," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1322–1329, Apr. 2018.

[20] J. Hsiung, M. Hsiao, E. Westman, R. Valencia, and M. Kaess, "Information sparsification in visual-inertial odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1146–1153.

[21] D. Wilbers, L. Rumberg, and C. Stachniss, "Approximating marginalization with sparse global priors for sliding window SLAM-Graphs," in *Proc. IEEE 3rd Int. Conf. Robot. Comput.*, 2019, pp. 25–31.

[22] M. Zhao et al., "A general framework for lifelong localization and mapping in changing environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 3305–3312.

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

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

[25] M. Gridseth and T. D. Barfoot, "Keeping an eye on things: Deep learned features for long-term visual localization," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 1016–1023, Apr. 2022.

[26] Y. Chen and T. D. Barfoot, "Self-supervised feature learning for long-term metric visual localization," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 472–479, Feb. 2023.

[27] J. Spencer, R. Bowden, and S. Hadfield, "Same features, different day: Weakly supervised feature learning for seasonal invariance," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6459–6468.

[28] R. Mur-Artal and J. D. Tardös, "ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[29] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres solver," 2022. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[30] G. Jocher et al., "ultralytics/yolov5: V6.0 - YOLOv5n 'Nano' models, roboflow integration, TensorFlow export, OpenCV DNN support," *Zenodo*, Oct. 2021, doi: 10.5281/zenodo.5563715. [Online]. Available: https://github.com/ultralytics/yolov5/issues/3995

[31] Z. Teed and J. Deng, "DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 16558–16569.

[32] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[33] N. Zimmerman, T. Guadagnino, X. Chen, J. Behley, and C. Stachniss, "Long-term localization using semantic cues in floor plan maps," *IEEE Robot. Automat. Lett.*, vol. 8, no. 1, pp. 176–183, Jan. 2023.