Semantic Mapping for Enhanced Localization in Indoor Environments

Allen Isaac Jose
Department of Computer Science
Hochschule Bonn-Rehin-Sieg
Germany
allen.isaac@smail.inf.h-brs.de

Deebul Nair
Department of Computer Science
Hochschule Bonn-Rehin-Sieg
Germany
deebul-sivarajan.nair@h-brs.de

Abstract

Simultaneous localization and mapping(SLAM) is employed to construct an environmental map, incorporating both spatial and semantic data from sensors to facilitate later relocalization. Traditional dense mapping techniques proved impractical for resource-limited mobile robots due to computational and storage constraints, necessitating the use of coarse object models to incorporate semantics into the map. Among these, QuadricSLAM and Objected Aided-SLAM(OA-SLAM) are two notable approaches, with the former requiring an RGBD image as input and the latter using an RGB image. Nevertheless, neither of these methods has been assessed on a common dataset to date. To address this gap, a synthetic dataset was generated, allowing for a quantitative assessment of mapping accuracy using various predefined metrics. The results demonstrate OA-SLAM's accuracy and robustness in the face of sensor noise and its capacity to relocalize with a single RGB image of the scene.

1. Introduction

With the increasing deployment of autonomous mobile robots in home and industrial settings, there's a growing need for these robots to have a comprehensive understanding of their environment for reliable and robust task execution. Mapping, the process of creating a virtual representation of the real-world environment, encodes the spatial and semantic features of the environment entirely or of certain landmarks. Localization requires the robot to match the sensor observations with the generated map's structure to determine its pose. In real-world applications where the robot explores a new environment, it must simultaneously address both mapping and localization, a well-known challenge known as SLAM. SLAM entails estimating the robot's pose and generating a map based on observations, accounting for uncertainties in the robot's action and sensor measurements.

One crucial consideration is the intended use of the created map. In this project, we are focused on the localization capability of the robot using the generated map in indoor environments. In the previous works, SLAM algorithms primarily mapped the environment's geometric structure using dense mapping. Semantic information is also added to these maps with the help of per-pixel semantic segmentation. In the 2D context, dense mapping was favoured for its smaller size, while 3D mapping posed storage and processing challenges, making sparse mapping more suitable. Sparse maps focus on extracting keypoint features and the localization is achieved through a bag-of-words approach as seen in ORB-SLAM[13]. However, these point features may get easily occluded when viewing from a different angle, illumination or at a farther distance. This led to the requirement of having bigger features such as objects that are appended into the sparse maps as coarse object models. Cuboids[22] and ellipsoids[24][15] are two such models which require only 9 parameters to represent position, orientation, and size. These object models possess the advantage of being visible from any viewpoint such as top, bottom and sides. In other words, we could say the SLAM is modified to incorporate object detection rather than relying only on keypoint detection.

Some of the important challenges in the domain of SLAM are real-time execution, size of the map, relocalization after tracking is lost, and data association. Real-time execution poses a primary challenge for SLAM algorithms. The tracking component, responsible for processing current images, must not be hindered by slower map optimization tasks. To enable SLAM to handle larger environments, it must eliminate redundant information within the map, such as keyframes, map points, and objects. Furthermore, it should recover when losing pose tracking due to sudden motion through relocalization. Global relocalization in the map should be possible from any camera viewpoint to make SLAM useful in localization tasks. Data association is another significant challenge, as it involves linking 2D bounding box detections across frames to the same object

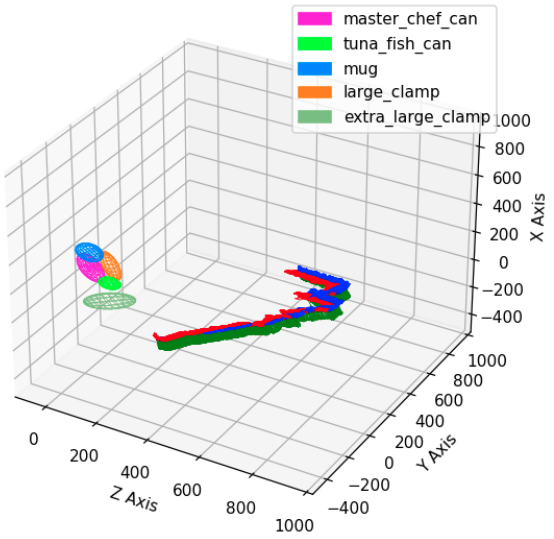

Figure 1. Object centric map - ellipsoids represent objects and the trajectory with 3D pose are plotted

or initializing new objects, with incorrect associations potentially causing object duplications or mergers. Additionally, SLAM typically assumes a static world, which may not always hold in real-world scenarios.

In object-oriented SLAM, the lack of high-quality datasets is a hindrance, primarily because ground truth 6DOF pose and object dimensions are often unavailable. Existing datasets, such as the YCB-Video dataset, are not optimized for SLAM due to low-quality images that hinder feature extraction. Creating a sparse map necessitates stable keypoints during map initialization, and the absence of a complete loop closure trajectory in available data further complicates the refinement of coarse object models to match object shapes. Additionally, images with substantial parallax are required to improve coarse object models. Prior SLAM comparison methods primarily focused on quantitatively analyzing trajectories while providing only qualitative map analysis. In our case, to quantitatively assess the generated object models against ground truth, new metrics must be introduced. One challenge lies in accurately measuring the overlap between the estimated ellipsoid and the ground truth cuboid which requires Monte Carlo sampling within the cuboid and checking if the point lies within the estimated ellipsoid due to their differing shapes.

In monocular RGB-based SLAM, absolute depth scale information is unavailable, and mapping occurs relative to an arbitrary scale. To enable comparative evaluation, a similarity transformation is needed to align the estimated map with the ground truth map. Rescaling can also be achieved by comparing the ratio of estimated object dimensions to the ground truth object dimensions for YCB objects. Map initialization can be problematic due to dominant planar surfaces, leading to unstable initial poses. Some SLAM algorithms, like ORB-SLAM[13], retain only keyframes within the map, making it challenging to compare with the ground truth trajectory. An alternative approach is to use the pose information for individual frames returned by the local tracking component of the SLAM.

This project focuses on the comparative evaluation of two semantic object-oriented mapping algorithms, QuadricSLAM[15] and OA-SLAM[24], both using ellipsoids to represent objects. The main contributions of this work are:

- The research involves conducting a literature survey of existing SLAM methods to justify the need for semantic object models in 3D sparse maps.
- An overview of QuadricSLAM and OA-SLAM is presented, highlighting their internal frameworks.
- To enable a fair comparison, a new dataset with a loop closure trajectory is generated using a blender-based rendering tool, storing ground truth trajectory and object pose information in BOP format for benchmarking.
- Quantitative evaluation metrics are defined to assess mapping accuracy, including runtime performance and system profile evaluation for real-time and resource-constrained applications.
- The robustness of object shape estimation is validated with experiments involving noisy 2D bounding boxes for objects.
- Localization capability is tested in fully mapped and partially mapped environments to assess the enhanced localization with the knowledge of object semantics.

2. Related Work

In 2D SLAM, a robot creates a two-dimensional representation of its environment by mapping obstacles as point features in a planar region. It uses range sensors like LIDAR or RGB-D cameras to estimate whether a cell is occupied or free on the map. Robot motion is tracked using wheel odometry or inertial units, but sensor errors can accumulate over time. To mitigate errors, SLAM is modelled as a probabilistic framework that estimates the robot's pose and map with the least error. Loop closure corrects measurements by revisiting previously explored areas. SLAM involves a front-end for tracking and a back-end for error reduction. Recent advancements integrate deep learning to add semantic information to 2D maps by projecting object labels onto corresponding map regions.

The 2D map types can be broadly classified into four: metric maps, topological maps, semantic maps and hybrid maps. Metric maps aim to create an accurate spatial representation of the environment using range measurement

sensors. In two-dimensional representations, the popular approach is the occupancy grid map[4], where the environment is divided into discrete grids, and each cell's occupancy status is computed as free, filled, or uncertain. Uncertainties in sensor measurements are addressed through a Bayesian filter, recursively updating the probability of each cell's state. A topological map is a compact representation visualized as a graph, with nodes representing free spaces and edges indicating traversability. It offers block-level discretization compared to the pixel-level discretization of an occupancy grid. Topological maps are primarily used in navigation applications for planning global paths. The advent of deep learning has given rise to semantic maps, offering a valuable addition to traditional map representations. While other maps focus on replicating spatial details, semantic maps introduce object semantics and room categories, enhancing navigation and planning tasks. Unlike metric maps where each cell is independent, semantic maps associate occupied cells with specific objects, aiding in scene change detection and providing more meaningful human visualization. These maps require both range sensors and RGB images for object classification[23] and room categorization[7][20]. In practical situations, a robot may need to perform various tasks, and different types of maps discussed earlier can serve specific purposes. This has led to the practice of combining these maps to form hybrid maps[5][17], allowing for overlapping and flexible usage depending on the requirements. Semantic maps can be used to represent individual rooms and topological maps can be used to represent the traversability between them. This approach enhances scalability and manageability by facilitating updates to smaller parts of the environment instead of a single map encompassing all rooms.

The maps mentioned above are created using different SLAM methods. The first one is the filter-based approach that utilises either the Kalman filter or a Particle filter approach to jointly estimate the robot poses and the observations or landmarks iteratively. Odometry and sensor observations act as inputs to the filter and the recursive Bayesian filter updates the current robot pose and the most probable map of the environment. Extended Kalman Filter(EKF) was the first version of SLAM in this family which is succeded by Rao-Blackwellized particle filter[3] which is implemented in the Gmapping SLAM algorithm[6]. Another set of SLAM algorithms is called feature-based mapping and HectorSLAM[9] is one such example. It uses ICP scan matching to find the rigid transformation between two consecutive scans which can be used to estimate the current pose and the map. To enable robust scan matching, the map is stored in multi-resolution scales of lower resolutions similar to image pyramids used in computer vision applications. In graph-based mapping, a factor graph is employed, with nodes representing robot poses and landmarks, and edges representing constraints. These constraints are akin to the elasticity of springs, with their stiffness reflecting measurement certainty. The more certain a measurement, the stiffer the edge, indicating high confidence in the optimization process. Kartoslam[10] is one such example. Graph SLAM comprises a front end for adding constraints and a back end for performing expensive non-linear joint optimization.

Now, localizing in this generated map on the second run is a requirement. The kidnapped robot problem is a famous challenge in robotics where the robot should localize itself within a map available in hand. The starting pose is unknown. This can also be called global localization. The time required to localize, the computational power required and the amount of motion required by the robot to gather enough evidence to localize are some of the properties that can be compared among the algorithms. In the outdoor environment, the GPS can aid in identifying the most probable search area within the map. However, in the indoor environment, the robot has to rely on the features for localizing. Adaptive Monte-Carlo Localization[16] is the most commonly used localization method that utilizes particles to represent the most probable pose of the robot. As the robot moves and more observations are sensed, the uncertainty in the particles reduces over time and converges to an area concentrated around a point. KL divergence is computed to check if the particles have been converged to reduce the number of particles for efficient computation.

2D SLAM methods are commonly employed due to their storage efficiency and reliance on AMCL for localization. However, these methods face limitations when applied to real-world environments that feature non-planar surfaces, stairs, and slopes, challenging the assumption of z=0. The reliance on observations at a fixed height of the 2D LIDAR may result in indistinct landmarks. To address these issues, the demand for 3D maps has emerged, providing a more comprehensive representation of features or landmarks. In 3D SLAM, the map dimension increases from 2 to 3, and the robot's pose transitions from 3DOF to 6DOF, demanding higher computational power and efficient real-time frame tracking strategies. This project focuses specifically on camera-based visual SLAM methods. The mapping methods within SLAM can be broadly categorized into dense mapping, representing the environment with high memory requirements, and sparse mapping, which extracts and stores essential features with limited memory usage. Both mapping approaches can incorporate semantics, with dense mapping using class labels for each pixel in the point cloud and sparse mapping assigning labels to keypoints and employing coarse object models for representation. Additionally, SLAM methods are classified into direct and feature-based methods, where direct methods align the entire image space between frames, while feature-based methods track specific features across multiple frames.

MonoSLAM[2] was one of the first real-time online 3D SLAM approaches using only RGB images from a monocular camera. Lack of depth estimation is compensated by placing a known-sized object in front during initialization to fix the scale. The usage of EKF limited the area of the environment that can be mapped. Other properties are pruning less reliable features, storing each feature as an image template and using it for tracking and relocalization. PTAM (Parallel Tracking And Mapping)[8] is a groundbreaking method that separates the tracking and mapping processes into distinct threads to enhance real-time performance. The Extended Kalman Filter (EKF) backend of MonoSLAM is replaced with bundle adjustment, marking the introduction of non-linear optimization. The algorithm uses FAST corner features and employs feature and keyframe pruning to reduce computational complexity. PTAM conducts local and global bundle adjustments, refining the relative pose of the latest frame and optimizing the entire map, respectively. DTAM (Dense Tracking and Mapping) is a direct mapping method that creates a dense map by photometric error of RGB images for alignment between consecutive frames as compared to reprojection error being used in feature-based methods. The algorithm employs iterative gradient-based operations, accelerated by GPU, to achieve a non-convex solution. KinectFusion[14] utilizes a Kinect RGBD camera to create a dense map in TSDF (Truncated Signed Distance Function) representation. The SLAM algorithm involves generating surface normal maps, and estimating the camera's pose through frame-to-map ICP matching with GPU support. SLAM++[18] is a first-of-kind semantic SLAM algorithm where an indoor environment is enriched with semantic information of known objects. The algorithm involves an initial training phase, creating 3D mesh models of commonly appearing objects using KinectFusion and generating point-pair features (PPF) as descriptors for object detection and 6DOF pose estimation. GPU support is needed and only 5 object models can be trained and searched.

RGB-D Mapping[19] utilizes a modified version of the Iterative Closest Point (ICP) called RGBD-ICP where the SIFT features from the image are also used to aid ICP with depth image. Loop closure was established in another thread with the RANSAC inlier checker for the SIFT features. In ElasticFusion[21], tracking is achieved through joint optimization of photometric (RGB) and geometric (depth) errors, known as frame-to-model tracking instead of pose graph optimization which helped to create a dense map using GPU. A surfel map representation is used where each surfel in the representation includes timestamps of initialization and the latest update, dividing the map into active and inactive regions based on update timestamps. SemanticFusion[12] is a modified version of ElasticFusion where a modified Convolutional Neural Network with VGG architecture is used to produce per-pixel label probabilities on the 3D surfel map. Real-Time Appearance-Based Mapping (RTAB)[11] is a comprehensive SLAM solution, widely embraced in the ROS community for its customizable framework properties and ability to run in real-time on CPU. Visual features are captured from the frontend and a graph structure is used in the backend to create a dense map with a bag-of-words loop closure system. CubeSLAM[22] is a feature-based sparse mapping method that represents objects as cuboids in monocular RGB images. It utilizes 2D bounding boxes from the YOLO object detector and leverages the vanishing point property for sampling cuboid corners. The cuboid is optimized through multiview bundle adjustment, with the ability to add object constraints in the graph-based backend. It can run at 30 Hz on the CPU and can track dynamic objects at 10 Hz.

In 3D maps, the most commonly used localization methods are feature-based[13] and visual-based ones. In the first approach, the extracted 2D features from the query image and correspondence matching are done with bag-of-words, and the Perspective-n-Point (PnP) algorithm is employed for pose estimation. An example of visual localization is RatSLAM[1] where the matching process involves comparing the new input image with template images in the database using a similarity score. An image template is created by cropping, subsampling and normalizing a well-informed image frame.

2.1. QuadricSLAM

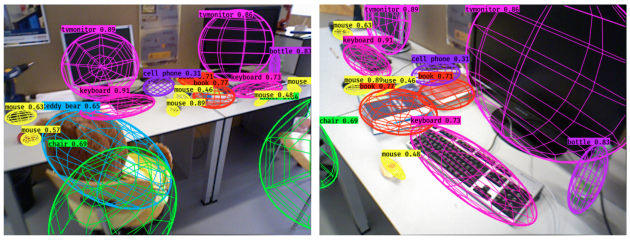

Figure 2. QuadricSLAM [15]*p. 1*

The Quadric SLAM[15] is one of the first semantic object-oriented SLAM approaches that represent each object in the environment using ellipsoids, optimizing their parameters through multi-view geometry. An ellipsoid can capture the centroid, size and orientation properties using 9 parameters(3 each). The input to the framework is RGBD images and the odometry value between each image. The YOLO object classification model is used on each input image to generate 2D bounding boxes for each object. These edges of the bounding boxes are projected as planes into the 3D world using a camera projection matrix and multiple views of the same object help to create a constrained closed space of tangential planes. The ellipsoids are initialized

within this closed space. This photogrammetric method is based on the concept of a dual quadric representation equation of tangential planes to a closed surface. $\pi^T Q^* \pi = 0$, where π is a 4D vector representing the plane parameters and Q^* is the dual quadric representation symmetric matrix of size 4x4 formed using the ellipsoid parameters. The value of this matrix needs to be optimized using multiple views of the same object. The YOLO also helps in assigning a label to each of the object ellipsoid using the Hungarian algorithm. In the backend, a graph optimization network is created with the robot poses and ellipsoid representation as nodes and they are connected by edges which represent the odometry and 2D bounding box uncertainties. Non-linear optimization method like ISAM is used for online mode and LM optimizer is used for offline or batch mode. The graph is created by extending the gtsam library to incorporate the ellipsoid observation constraints and its error term to be fitted into the non-least squares problem that needs to be minimized. In QuadricSLAM, a joint optimization is performed on the graph where the error in odometry and observation is used to correct the camera pose and object pose together. There were some notable problems like error accumulation due to inaccurate tracking. Since the only observation in the graph is the object ellipsoid, it is not a robust anchor as the ellipsoid has a very high uncertainty until a good coverage of the object is mapped. Because of this, the errors in the observation cause the robot pose to deviate a lot to compensate for the errors. This results in a highly erroneous map with object duplications because of the incrementally optimized inaccurate camera pose. Also, the computation is not efficient as the graph grows with a lot of redundant information. As a result, loop closure is not visible when a highly noisy sensor measurements are available. Even though the depth information of the object can be extracted from the depth image, there can also be noises for far aware objects resulting in erroneous ellipsoid initializations.

2.2. OA-SLAM

OA-SLAM[24] is a modified version of ORB-SLAM[13] where the map with ORB features is enriched with semantic information of objects using ellipsoids. The mathematical concept behind ellipsoid creation is same as that of QuadricSLAM, but the only difference is that here RGB image is used instead of RGBD image. Because of this, the ground truth scale of the scene is not available. Also, the odometry isn't needed as an input because the relative transformation between frames is computed internally using visual odometry method as a part of the SLAM framework. The main advantages of this SLAM comes from the ORB-SLAM counterpart. In ORB-SLAM, the tracking and mapping are run over two separate threads so that the faster realtime tracking is not blocked by the slower mapping component caused by

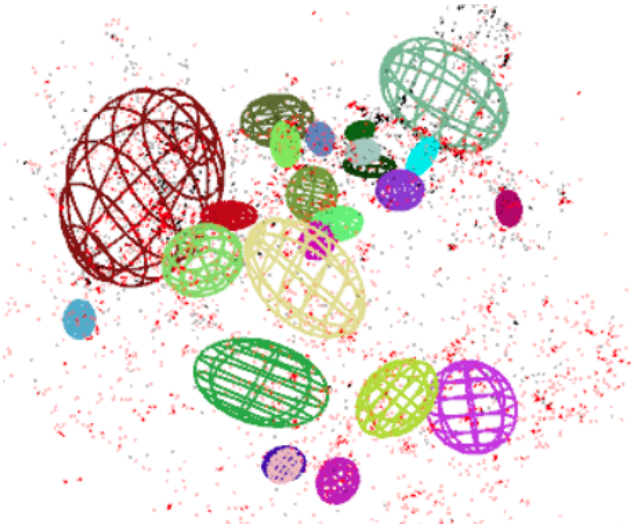

Figure 3. OA-SLAM [24]*p.5*

expensive optimization step in the bundle adjustment. As a result, there exist two bundle adjustment techniques such as local bundle adjustment where a moving window is used to perform a faster tracking optimization in tracking thread and the global bundle adjustment to proporgate the errors to the whole map to correct it. ORB features were superior over other features such as SIFT, SURF, FAST in terms of its invariance to a lot of properties, especially view point. As a result, a global relocalization capability was implemented where the current features in the query frame can be matched with the features in the map using bag-of-words approach for camera pose estimation. Another feature was the culling property of map to either fuse same features or reject features and keyframes that possess redundant information thereby keeping the whole map efficient for storage and the optimization computation. The bag-of-words method also helps in loop closing where the correspondence between similar features can be identified and the similarity transformation can be computed to close the loop. In OA-SLAM, another thread was created for object ellipsoid mapping and processing. It is an independent thread where it interferes with the main thread only when the ORB features is not able to localize by itself. The real use of the mapped ellipsoids is when performing a global relocalization from a viewpoint that is different from a unmapped region. ORB features are not able to detect any corresponding features whereas the objects are able to be detected and the relative position of the objects in the 2D image can be compared with the 3D map to estimate the camera pose using PnP algorithm. The results shows good robustness against measurement noises and the only existing problem is the quantitative evaluation of map due to lack of availability of ground truth depth scale.

3. Methodology

3.1. Dataset generation

To compare the SLAM algorithms, we need a dataset with ground truth information of both the camera poses and the object poses. TUM RGBD dataset lacked ground truth object pose information and YCB-Video dataset images were of low resolution causing ORB features not being stable. Also, loop closure image sequences are not present in both of them which is necessary for correction step in the map.

Figure 4. Sample RGB image of the scene

To address this issue, a synthetic dataset is created with blender-based rendering tool which satisfies all the above requirements. The dataset is generated in BOP benchmarking format which contains ground truth camera pose, object pose, RGB image, depth image, and object bounding boxes. Different use cases of the semantic object-oriented SLAM can be tried out here, for example, by generating cluttered scenes and checking for duplication or missing objects in the generated map. 10 such scenes were generated at different camera height and radius of the mapping trajectory which is a circular one. 5 random YCB objects of known dimensions are placed in each scene. Each scene consists of 1500 images where 1400 images belong to 1 full trajectory along the circle and an additional 100 more images are covered by the camera to ensure the loop closure event can be made by the SLAM. Each dataset generation took around 2.5 hours and the resolution of the images was 640x480 pixels. The bounding boxes and the ground truth odometry is passed to QuadricSLAM and OA-SLAM with sufficient noise to simulate the real world scenario.

To test the robustness of the SLAM frameworks towards the instability or uncertainty of the bounding boxes, the file containing the ground truth bounding boxes is corrupted with noise. This is done either by removing the bounding box for an object, changing the centre of the bounding box or by changing the width or height of the bounding box in each frame. Another experiment was designed to test the localization capability of the object-oriented maps. QuadricSLAM didn't have any inbuilt localization capability. So the experiments were performed only on the OA-SLAM. The experiment aims to prove the concept that having object representations along with feature point representations can aid in improved localization.

3.2. Evaluation criteria

To compare the quality of the trajectory, five metrics were defined. Trajectory deviation error compares the euclidean distance between the ground truth and the estimated 3D position of the camera. Root mean square of all the values in a scene is computed. Rotation error is computed by comparing the ground truth and the estimated 3D angles in quaternion format. The average rotation error for all the values in a scene is computed. The next 3 metrics are used to compare the similarity of the trajectory with the ground truth as in OA-SLAM, as the depth information is lacking. Procrustes Analysis, Fréchet Distance and Chamfer Distance are the metrics used and their values should be as close to 0 in ideal case.

To compare the quality of the mapped ellipsoid, four metrics are defined. Centroid error is used to compare the euclidean distance between the ground truth cuboid and the estimated ellipsoid centers(3D position - x, y, z). Similarity, a rotation error is computed for each object elliposids. Finally, we had to defined two more metrics to compare the IoU and the size of the mapped objects with respect to the ground truth. The IoU approach involves Monte Carlo sampling with 10,000 randomly generated points within the ground truth 3D cuboid. The goal is to determine how many of these points fall within the estimated ellipsoid, providing an estimate of the overlapping percentage. While not providing an exact value due to the geometric differences, this method allows for the comparison of different SLAM methods on the same scene, with higher overlapping volumes indicating better performance. In specific scenarios, the Intersection over Union (IoU) metric may show 0% overlap between objects in their true poses, even when they are very close. To address this, aligning the estimated object pose with the ground truth pose before computing IoU is suggested. This alignment allows for a more meaningful comparison of the volumes of the ground truth and estimated objects. The IoU equation remains the same, with the only modification being the alignment of the objects before calculating the intersection volume.

To compare the system level performance, the system profile is computed involving properties like CPU usage, RAM usage and the computational time needed to complete the SLAM process. The computational time also gives an idea of the FPS of the SLAM algorithm. Also, the size of the generated map is compared to check how efficiently the environment can be mapped while preserving all the important features for operations such as localization.

4. Experiments and results

4.1. Localization experiment

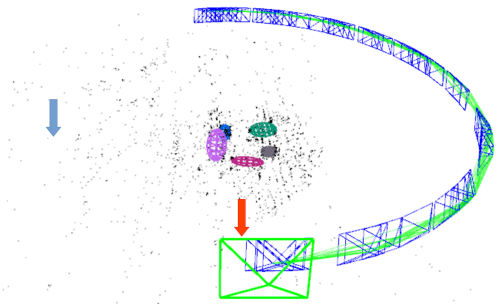

Figure 5. Localization using points on the query image + 9 nearby images

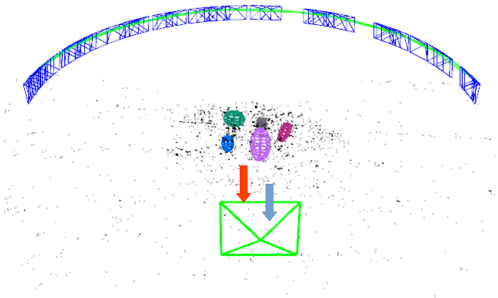

Figure 6. Localization using points+objects on the single query image

To test the relocalization capability of object-oriented SLAM, the localization mode in OA-SLAM is utilized. There are two localization modes: one using only points and the other one utilizing both objects and points. For the points-only localization mode, the current observation needs to be searched using the bag-of-words approach to identify the most probable location. Whereas in the objects+points mode, if the bag-of-words approach couldn't identify the exact pose, then the knowledge of the objects is utilized to perform 2D-3D correspondence(2D information from the query image and 3D information from the available map) using the PnP algorithm. To test the performance, we have mapped the environment scene 000009 using the first 750 of the images to create a Half map. The query image which we selected was 001071.png which belonged to the unmapped region. The ground truth pose is given by blue arrow and predicted pose by red arrow and the green camera frustum. In figure5, we can see that even though we provided 9 nearby images, still the point-only localization mode was not able to match any ORB features as the viewpoint is totally different from that which is mapped. Whereas in figure6, we can see that within a single query image itself, the PnP algorithm was able to localize the camera pose since correspondence matching can be performed using the semantic information of the objects.

4.2. Performance experiment

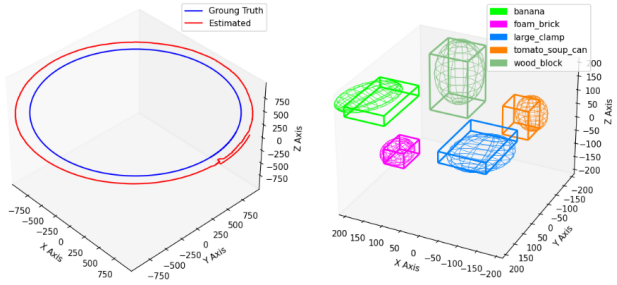

Figure 7. OASLAM - estimated trajectory(left) and objects(right)

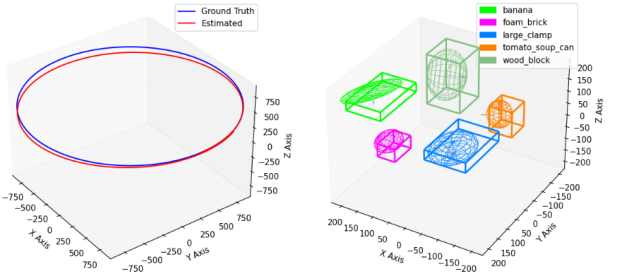

Figure 8. QuadricSLAM(batch mode) - estimated trajectory(left) and objects(right)

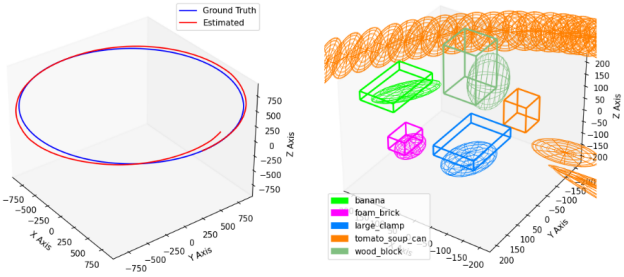

Figure 9. QuadricSLAM(incremental mode) - estimated trajectory(left) and objects(right)

Scene 000009 is selected for mapping. The first figure7 represents the OA-SLAM mapping. The object mapping shows that the objects are well overlapped. The cuboid represents the ground truth object model and the ellipsoid is the estimated object model. In the left figure, we can see that the estimated trajectory shape(red) is well aligned with the ground truth trajectory(blue). We can see a constant bias error and this is because since the monocular camera is used

for SLAM, there is no depth information and we have to perform a scaling operation followed by ICP alignment for the comparative evaluation. Towards the right side of the trajectory, we can see the overlapping 100 extra trajectory frames we added for loop closure which is also well aligned. This indicates the importance of loop closure in monocular visual SLAM where scale drift error accumulates over time.

Figure8 indicates the quadricSLAM in batch optimization mode. Since the depth information is available via RGBD image, the trajectory and mapping are well aligned. However, the object shape is not as perfect compared to OA-SLAM. The incremental mode of QuadricSLAM is needed for real-time operation, but the error accumulation caused the camera pose to deviate heavily from the ground truth and the mapping of the orange-coloured object was duplicated as the data association was not able to be performed.

	OA-SLAM	QuadricSLAM - batch	QuadricSLAM - incremental
Average CPU Utilization	30.59%	16.01%	58.97%
Min CPU Utilization	10.86%	10.61%	10.24%
Max CPU Utilization	35.96%	75.85%	71.43%
Average Memory Utilization	645.09 MB	209.43 MB	261.52 MB
Min Memory Utilization	128.53 MB	205.05 MB	211.66 MB
Max Memory Utilization	744.55 MB	234.16 MB	328.04 MB
Overall Time Taken	107.13 s	49.06 s	532.58 s

Figure 10. Table showing the system profile

The system profile depicted in the table10 illustrates the performance when executing algorithms on an Intel i5-8300H with 4 cores and 8 threads. The average CPU utilization is notably elevated in the case of OA-SLAM because it not only maps objects but also the point features along with a separate thread for loop closing. The QuadricSLAM in incremental mode had the highest CPU utilization because we were using LM optimization instead of iSAM2 incremental optimization as it was showing errors for an underdetermined system. The usage of point features and ORB descriptor vocabulary along with object features explains the higher memory utilization for OA-SLAM. The overall time taken is not exactly comparable as OA-SLAM is written in C++ and QuadricSLAM is written in Python. But still, a rough analysis shows that OA-SLAM processed 1500 images in 107 seconds which means it can operate at 14 FPS. The quadricSLAM in batch mode is really fast as the expensive optimization step is performed only once. But in the online operation case, the incremental mode in QuadricSLAM is to be compared and this operation took 532 seconds which is roughly 3 FPS.

The figure11 depicts that the QuadricSLAM in batch mode was able to have minimal object centroid error as compared to OA-SLAM and this is due to the availability of the depth information from the dataset for QuadricSLAM. In the case of OA-SLAM due to the map generated with arbitrary scale, while the alignment process in post-processing, a constant bias error exists for all objects.

Figure 11. Object Centroid Error

When we computed the average object centroid error of each SLAM, for OA-SLAM, its 30.68mm, for QuadricSLAM in batch mode 24.95 mm and for the incremental mode, its 80.25 mm. This shows the better object centroid estimation capability of QuadricSLAM in batch mode.

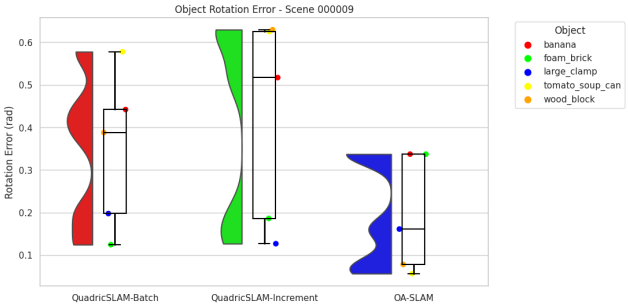

Figure 12. Object Rotation Error

The rotation error is computed using the quaternion representation of the Euler angles of the object's ground truth and estimated 3D pose. Figure12 shows the average rotation error is 0.19 rad, 0.35 rad, 0.42 rad for OA-SLAM, QuadricSLAM-batch and QuadricSLAM-incremental respectively. This highlights the OA-SLAM's capability to capture the object's pose accurately.

The figure13 shows how the Monte-Carlo sampling is performed to compare IoU. 10,000 samples are generated in each object cuboid and are compared with the estimated ellipsoid for overlap checking.

Figure14 shows the higher IoU capability of OA-SLAM. 58.79%, 55.74% and 24.82% are the average overlap percentage for OA-SLAM, QuadricSLAM-batch and QuadricSLAM-incremental respectively. In the case of OA-SLAM, even though the centroid error is higher than that of QuadricSLAM, the exact capture of the shape helped in providing a good overlapping. In the case of QuadricSLAM, the centroid error was lesser, so even though the shape is not accurate, the object was well inside the ground truth to give a good overlapping result.

Even though the previous metric helped in compar-

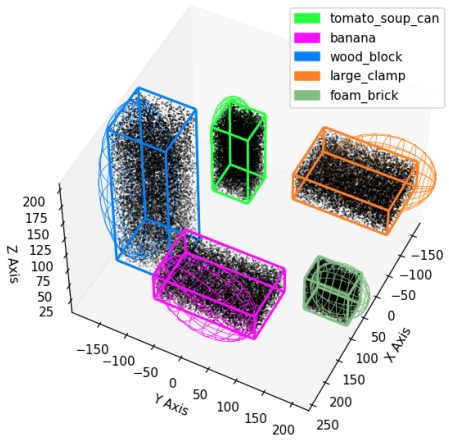

Figure 13. Object IoU Comparison Method

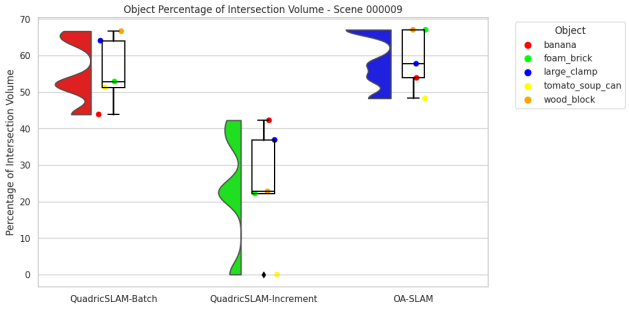

Figure 14. Object Overlap Percentage

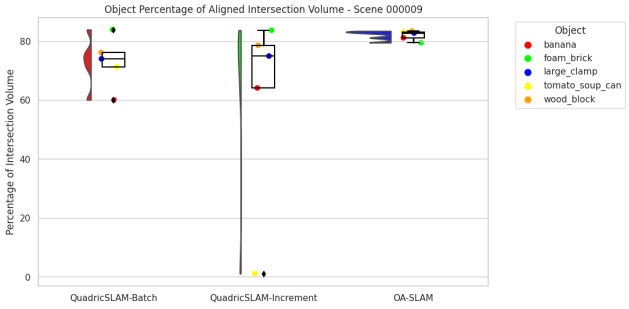

Figure 15. Object Aligned Overlap Percentage

ing the overlapping, since there is a bias error existing in OA-SLAM due to scaling and aligning, we will align the cuboid and ellipsoid and then estimate the IoU which is a good indicator of how accurately the shape is captured. Figure15 shows that 81.97%, 73.06% and 60.48% are the average overlap aligned percentages for OA-SLAM, QuadricSLAM-batch and QuadricSLAM-incremental respectively.

The Euclidean distance between the ground truth and the estimated camera position is compared frame by frame.

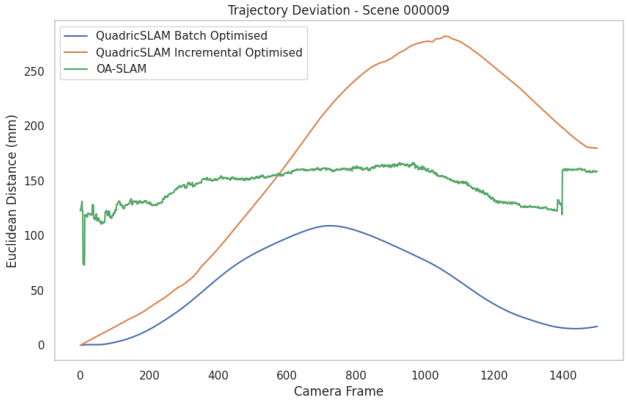

Figure 16. Trajectory Deviation Error

In figure16, we can see that the QuadricSLAM in batch-optimized mode is having the lowest errors. We can see that the error increases over time and peaks at around 108.87 mm. This happens on the opposite side of the starting frame as it is the least constrained side. The QuadricSLAM in incremental mode has the highest error which peaked at 281.64 mm. In the case of OA-SLAM, the error starts at 73.07 mm and peaks only till 166.45 mm. This is a constant bias error due to the scale error of monocular camera-based SLAM. If we consider subtracting the least error 73.07 mm from its error plot, then the error would only be peaking at around 93 mm for OA-SLAM which is better than Quadric-SLAM in batch mode. In the factor graph, similar viewpoints(loop closure views) at the starting and ending points make it a strong anchor and the viewpoints on the other side of the circular trajectory have a weaker anchor. So, when optimization is performed, the other side has a tendency to bend down to compensate for the object pose errors. This is why the errors are more on the side opposite to the starting or ending point. 146.88 mm, 65.45 mm and 191.60 mm are the root mean square error for OA-SLAM, QuadricSLAM-batch and QuadricSLAM-incremental respectively.

The rotation error in figure17 also follows a similar pattern as in the case of trajectory deviation error. 0.04 rad, 0.06 rad and 0.14 rad are the average trajectory rotation error for OA-SLAM, QuadricSLAM-batch and QuadricSLAM-incremental respectively. 0.05 rad, 0.09 rad and 0.23 rad are the peak trajectory rotation error for OA-SLAM, QuadricSLAM-batch and QuadricSLAM-incremental respectively. OA-SLAM had the least error and that too which is a constant one.

There were other trajectory similarity metrics such as Procrustes analysis, Fréchet distance and Chamfer distance. The estimated disparity values using Procrustes analysis are $9.19e^{-5}$, $1.38e^{-5}$ and $1.67e^{-3}$ for OA-SLAM, QuadricSLAM-batch and QuadricSLAM-incremental respectively. The estimated Fréchet Distances are 166.02 mm,

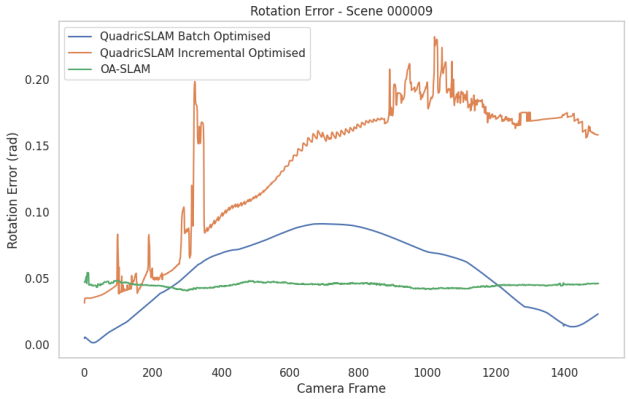

Figure 17. Trajectory Rotation Error

107.60 mm and 128.96 mm for OA-SLAM, QuadricSLAM-batch and QuadricSLAM-incremental respectively. The estimated Chamfer Distances are 285.53 mm, 105.03 mm and 157.29 mm for OA-SLAM, QuadricSLAM-batch and QuadricSLAM-incremental respectively. These values follow a similar trend as that of the trajectory deviation error.

The size of the map is another parameter for comparing the storage of maps. The QuadricSLAM in batch mode takes 454.4 KB and in incremental mode takes 625.7 KB size. The increased size of the incremental mode is due to many duplicated objects present. This map contains the camera trajectory poses and the object quadric parameters. For OA-SLAM, the size is 5.5 MB and this is due to the storage of camera poses, keyframes, the features detected in each keyframe, the map points, the object map points and the object quadric parameters. Camera poses are not required for map representation and are only needed for quantitative evaluation with ground truth. Using the map of OA-SLAM, we can also perform relocalization with sufficient point features. Still, the overall comparison with other dense mapping strategies shows that this type of map representation is very efficient in several orders of magnitudes.

4.3. Robustness experiment

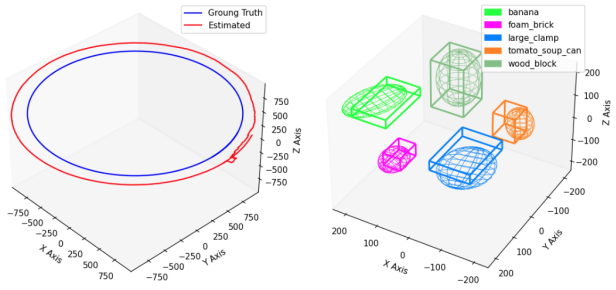

Figure 18. OASLAM - estimated trajectory(left) and objects(right)

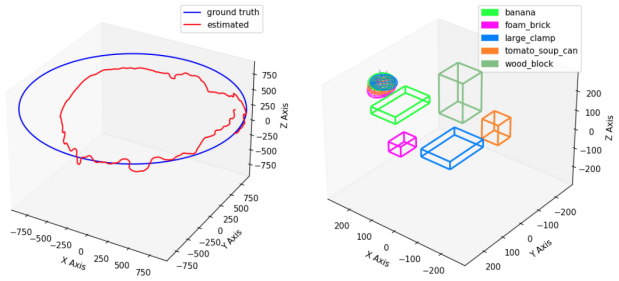

Figure 19. QuadricSLAM(batch mode) - estimated trajectory(left) and objects(right)

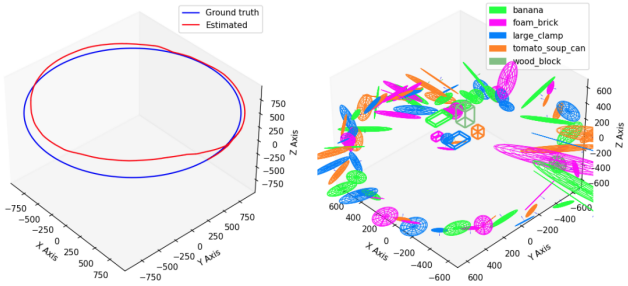

Figure 20. QuadricSLAM(incremental mode) - estimated trajectory(left) and objects(right)

The same scene 000009 is corrupted with observation noise. This is induced by varying the centre of the bounding box, the width and height of the bounding box and even removing the bounding box for 1 of the object in each frame randomly. The results are plotted above. The results shows that the QuadricSLAM in both modes(19, 20) were not able to map the objects correctly and hence further evaluation using metrics is unnecessary. Whereas the OA-SLAM18 was quite robust to noisy observations as the object mapping is performed on a separate independent thread and the main tracking thread is robustly tracking the point features that offer better stable anchors.

5. Conclusion

This project aimed to compare the performance capabilities of semantic object-oriented SLAM methods such as QuadricSLAM and OA-SLAM. We created a dataset consisting of 10 scenes in BOP format which can be used for quantitative analysis of the mapped object poses and shapes. We tested both the SLAM algorithms on these datasets and defined evaluation metrics for quantitative analysis of the camera trajectory and the objects. Also, we modified a scene to have noisy bounding boxes and were able to evaluate its effect on both the SLAM algorithms. Finally, the localization capability of objects was tested and proved in a scene where query images from the unseen part of the

mapped environment were used. The experiments help to identify the areas of improvement and propose some suggestions for future work. Also, a short survey into the history of 2D and 3D SLAM was done to give an idea of what led to the use of coarse-object models in sparse maps.

A possible future work involves incorporating additional constraints, particularly the plane constraint from a non-open-sourced structure-aware SLAM code, into OA-SLAM for improved estimates. The addition of planes can enhance the visual richness of the generated map, aiding users in understanding the environment. Specifically, the introduction of planes like walls can facilitate segmentation of indoor spaces into separate rooms, enabling the creation of a topological map for global path planning. Moreover, floors as planes can assist a robot in identifying traversable paths. The sparse map, consisting of points, objects, and planes, can be voxelized and transformed into a 3D occupancy grid map for navigation.

References

[1] David Ball, Scott Heath, Janet Wiles, Gordon Wyeth, Peter Corke, and Michael Milford. OpenRatSLAM: an open source brain-based SLAM system. *Autonomous Robots*, 34(3):149–176, Apr 2013.

[2] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[3] A. Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. *ArXiv*, abs/1301.3853, 2000.

[4] A. Elfes. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 22(6):46–57, 1989.

[5] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J.A. Fernandez-Madrigal, and J. Gonzalez. Multi-Hierarchical Semantic Maps for Mobile Robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2278–2283, 2005.

[6] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[7] Burak Kaleci, Kaya Turgut, and Helin Dutagaci. 2DLaserNet: A deep learning architecture on 2D laser scans for semantic classification of mobile robot locations. *Engineering Science and Technology, an International Journal*, 28:101027, 2022.

[8] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

[9] Stefan Kohlbrecher, Oskar von Stryk, Johannes Meyer, and Uwe Klingauf. A Flexible and Scalable SLAM System with Full 3D Motion Estimation. In *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160, 2011.

[10] Kurt Konolige, Giorgio Grisetti, Rainer Kümmerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent. Efficient Sparse Pose Adjustment for 2D mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 22–29, 2010.

[11] Mathieu Labbé and François Michaud. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[12] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, 2017.

[13] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[14] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[15] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. QuadricSLAM: Dual Quadrics From Object Detections as Landmarks in Object-Oriented SLAM. *IEEE Robotics and Automation Letters*, 4(1):1–8, Jan 2019.

[16] Patrick Pfaff, Wolfram Burgard, and Dieter Fox. Robust Monte-Carlo Localization Using Adaptive Likelihood Models. In Henrik I. Christensen, editor, *European Robotics Symposium*, pages 181–194, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[17] Andrzej Pronobis and Patric Jensfelt. Large-scale Semantic Mapping and Reasoning with Heterogeneous Modalities. In *IEEE International Conference on Robotics and Automation*, pages 3515–3522, 2012.

[18] Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H.J. Kelly, and Andrew J. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013.

[19] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, Oct 2012.

[20] Niko Sünderhauf, Feras Dayoub, Sean McMahon, Ben Talbot, Ruth Schulz, Peter Corke, Gordon Wyeth, Ben Upcroft, and Michael Milford. Place Categorization and Semantic Mapping on a Mobile Robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5729–5736, 2016.

[21] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. ElasticFusion: Real-Time Dense SLAM and Light Source Estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.

[22] Shichao Yang and Sebastian Scherer. CubeSLAM: Monocular 3-D Object SLAM. *IEEE Transactions on Robotics*, 35(4):925–938, 2019.

[23] Nicky Zimmerman, Tiziano Guadagnino, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Long-Term Localization Using Semantic Cues in Floor Plan Maps. *IEEE Robotics and Automation Letters*, 8(1):176–183, 2023.

[24] Matthieu Zins, Gilles Simon, and Marie-Odile Berger. OA-SLAM: Leveraging Objects for Camera Relocalization in Visual SLAM. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 720–728, Oct 2022.