Open-Set Object Based Data Association

by

Tim Y. Magoun

S.B. Electrical Engineering and Computer Science, MIT, 2022

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2024

Authored by:	Tim Y. Magoun Department of Electrical Engineering and Computer Science January 26, 2024
Certified by:	John J. Leonard Collins Professor of Mechanical and Ocean Engineering, Thesis Supervisor
Accepted by:	Katrina LaCurts Chair, Master of Engineering Thesis Committee

Open-Set Object Based Data Association

by

Tim Y. Magoun

Submitted to the Department of Electrical Engineering and Computer Science
on January 26, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Representing the world using sparse objects allows for compact and semantically meaningful maps in simultaneous localization and mapping (SLAM). Traditionally, object detectors trained on a specific set of objects, such as the YCB objects, are used to provide input to the data association problem, which limits the scope of the system to environments that it has been trained on. With advancements in foundational models, we can extend this representation for objects that are not known a priori and do not have a labeled category during training. This thesis explores a system that creates data associations between open-set objects using an RGB-D camera and how it is used in a sparse object SLAM system. We show comparable trajectory performance to traditional SLAM systems while being more adaptable to out-of-distribution objects.

Thesis supervisor: John J. Leonard
Title: Collins Professor of Mechanical and Ocean Engineering

Acknowledgments

There are many individuals who positively shaped my journey up to this point. I would like to highlight a few who have made an especially large impact in this chapter. Thank you to Prof. Leonard for his continued support and insights during this process. I couldn't have done it without his guidance and pointing me to the right resources. His kindness made the challenges seem more surmountable. Thank you to my lab mates, Alan Papalia, Kurran Singh, and Ziqi Lu for their technical and personal advice. Especially Alan, who was always there to listen and went above and beyond to help in problems ranging from mathematical to administrative. Bowie was a huge help, of course, and his presence never fails to brighten my day. Finally, I have so much gratitude for my parents and grandparents, who have given so much in the past 23 years.

This work is dedicated to 外公 and 外婆.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As robots become more capable and more ubiquitous, their use becomes more widespread and must continue to adapt to new environments. One model of robotics is the perceive-plan-act loop, where the robot repeatedly perceives the world and its location in the world, plans a course of action such that they are accomplishing their tasks safely and effectively, and then acts on that plan by using control methods that are robust to uncertainty and can correct for errors. This thesis focuses on the first part: perception. More specifically, we focus on the problem of simultaneous localization and mapping, or SLAM for short.

SLAM is the problem of estimating the location of a robot in a previously unmapped world while also creating a map of the world[1][2]. This is a chicken-and-egg problem as mapping requires knowing the location of the robot, and knowing the location of the robot in the world requires a map of the world. SLAM systems solve this problem by iteratively updating the map and the robot's location. The map is typically represented as a set of landmarks, which are points in the world that are easily identifiable and can be used to localize the robot. The robot's location is typically represented as a pose, which is a combination of the robot's position and orientation.

While other localization methods exist, such as using a GNSS system or inertial navigation, they come with unique disadvantages that make them difficult to implement in many applications. For example, GNSS systems broadcast signals with very low power levels, which makes them susceptible to interference and jamming. Additionally, the signals are unable to penetrate through walls or water, which makes them unsuitable for indoor or marine applications. Even in areas with some sky-view such as dense urban centers, GNSS systems may perform poorly due to multi-path effects where the GPS signal bounces off buildings before arriving at the receiver. Inertial navigation systems are susceptible to drift, which makes them unsuitable for long-term localization. Additionally, their performance grows with respect to the cost of the sensor, which can drastically increase the overall cost of the system. SLAM systems can overcome these disadvantages by using a combination of sensors, such as cameras and IMUs, to create a map that can be used to limit drift and operate in unknown environments.

The problem of SLAM has been studied for decades[3], and there are many different approaches to solving it. One approach is to use a dense representation of the world, where the map is a 3D point cloud or mesh[4][5] This approach is typically used in systems that use LiDARs as their primary sensor, as LiDARs can create dense representations of the world. Another approach is to use a sparse representation of the world, where the map is a set of select points or objects that are distinct. Traditionally, landmarks can be broken down into two different categories: primitive types and objects. The former include visual descriptors such as ORB[6], SIFT[7], or SURF[8], while the latter include objects with a semantic label such as cars, tables, and trees. Primitive types are more adaptable to unknown environments as they are not limited to a specific set of objects, but they are not semantically meaningful. Objects are more semantically meaningful, but until recently, they solely rely on closed-set object detectors which can only be used in the environments that they are trained on. This thesis explores a system that uses recent developments in open-set detectors to perform semantic object SLAM in environments previously unknown to the user, thus increasing the adaptability of object SLAM systems.

1.1 Motivation

The two main motivations of this thesis are outlined below:

1.1.1 Adaptable Object SLAM

As previously mentioned, current object SLAM systems require a frontend that outputs distinct semantic class labels for each object. These semantic class labels are used in the data association process to match current observations with previous ones, and jointly improve the quality of the trajectory and the map. Closed set object detectors, such as [9][10] need to be trained on a closed set of labeled images, such as COCO[11], YCB[12], and Object365[13]. That makes them only suitable for certain types of environments that are composed of those objects. With an open-set object detector, the system could be used in novel environments without fine-tuning the detector, thus reducing the amount of work required for the transition.

1.1.2 Sparse Semantic Map

The map created by a SLAM system could be used for downstream tasks like task and motion planning. For example, the robot can use the map to determine frontiers for active SLAM or to perform search and rescue operations. A semantically meaningful map can also be more suitable for cooperation with humans, as object-based maps are more intuitive for scene understanding. The semantic objects in the map can also be used to construct higher-level representations of the scene, like scene graphs.

1.2 Related Work

1.2.1 Foundational SLAM Systems

Early approaches to SLAM often relied on a filter-based method, such as [14][15][16][17]. These methods are able to perform SLAM in real time but can only support a smaller state and rely on the irreversible marginalization of variables to maintain its filter form. Graph-based methods, such as [18][19][20] improved on this by maintaining all state variables and transformed a filter-based method into a smoothing and mapping problem. This allows for more accurate estimates of the state and allows greater flexibility in correcting previous errors present in the trajectory. A more detailed review of the past approaches to SLAM can be found in [3]. One important framework that is widely used in SLAM is the Georgia Tech Smoothing and Mapping library (GTSAM) [21], which allows for easy implementation of graph-based SLAM systems with a variety of different sensing modalities. To improve the speed of operation, incremental smoothing and mapping (iSAM) [22] and its successor iSAM2 [23] were developed. These methods can perform online smoothing and mapping by dynamically determining the subset of the problem that needs to be resolved based on additional information. The primary difference between the two works is the use of a Baye's tree in the second work, which removed the need for periodic batch optimization.

Some classic sparse systems include PTAM[24],ORB-SLAM [25][26], and LSD-SLAM [27]. Which use point features calculated from images as landmarks in the map. These systems perform well in textured environments and are commonly regarded as SOTA for camera-based SLAM.

1.2.2 Semantic and Object SLAM

In addition to points as landmarks, objects have been used as map representations in both camera-based and LiDAR-based SLAM systems. In all previous work in this section, a closed-set front-end object detector is used to provide the input object detection and semantic segmentation.

A subset of object SLAM systems are considered dense: that is, all observed points and their corresponding properties are stored in the map. In Fusion++[28] and SuMa++[29], the authors use a TSDF or surface element representation to encapsulate object-level information and use that semantic label in the localization process. Fusion systems such as MID-Fusion, MaskFusion, ElasticFusion, and Co-Fusion[30][31][32][33] also represent the objects as a set of point clouds with semantic labels. While dense object SLAM systems provide a rich reconstruction of the environment and are able to capture the 3D object shape very well, they suffer from a computational bottleneck due to the large amount of data that needs to be processed at each frame. Additionally, they are not able to categorize an object as a singular entity and instead have a loose collection of points that are all of the same class. This makes it difficult to perform downstream functions directly on the map built by this method. Kimera[34] uses a mesher to create volumetric meshes of objects, which aims to reduce the computational requirements.

On the other end of the spectrum, sparse object methods use geometric primitives to represent the object. Such primatives include planes[35], cubes[36], quadrics[37][38], and 6 DOF Poses[39]. These methods are very general and require much fewer bits of information to encode landmark position and size, but they lack the ability to represent the shape of the object. Additionally, certain precautions have to be taken to ensure that object observations are well-aligned with the object model. For example, QuadricSLAM[37] uses a truncated projection of the quadric to the image plane and measures the intersection-over-union of the projection with the observed bounding box, and posed object SLAM needs to use additional techniques to deal with rotationally symmetric objects[40].

There exists an intermediate form of representation where either specific object primitives such as 3D models of chairs and tables are used [41][42], which preserves both the sparsity and the categorical geometric accuracy of the object, as well as representations using a

NERF model[43][44] for each item in the map, which captures the geometry of each instance accurately. NeuSE[45] uses a neural-implicit representation, which compacts the object shape into a latent encoding specific to the class of object. This saves on computation and adds SE(3) equivariance to the objects in the map. However, all of these systems are difficult to extend to new classes of objects since they require object-class-specific models to be trained or specified beforehand. For open-set systems, we can only use primitive geometric or dense representations.

Other semantic systems use scene graphs as a higher level representation of objects, [46] and [47] are good examples how how they are used to represent the world in both open and closed set configurations.

1.2.3 Open-Set Tracking and Mapping

With the increasing amount of data available for training, it recently became possible to train object detectors on an internet-scale dataset. This has led to the emergence of attention-based models such as CLIP[48] or DINO[49][50] which are able to perform zero-shot tasks or tasks that were not included in the training set of the model. ConceptFusion[51] has used this to perform open-set fusion by superimposing the CLIP embeddings of detected objects onto a dense map of the environment. [52], [53], [54] uses a similar approach and creates an open-set map with semantic embeddings to perform robotics-related tasks using natural language. However, these methods are not able to perform SLAM as they do not use the semantic information of the objects in the localization or mapping process. Therefore these works lay adjacent to our work but are not directly comparable. The main technical difference that allows our work to be used for SLAM is the use of data association algorithms that consider both geometric information and semantic information. Works such as [51] only use geometric or photometric information to perform data association in their localization and mapping.

Another adjacent field is open-world and open-vocabulary multi-object tracking (MOT).

This field focuses on tracking multiple dynamic objects through the scene, with a static or dynamic camera. One distinction between MOT and SLAM is that MOT does not require the tracking of objects after they move out of the field of view, which means that it does not maintain a global map. This field was introduced by [55], which proposed a new dataset and a baseline tracking-by-detection pipeline with an open-set detector. Other tracking-by-detection methods soon followed, with SMILETrack [56] and [57]. The performance was improved by learning-based methods that leverage transformers, such as [58][59] and [60]. PROB[61] introduces a probabilistic objectness into the model. The works mentioned have been using a monocular RGB image, but there also exists works that leverage geometric features such as depths and normals: GOOD[62]. Some method, like ViLD[63], uses a region proposal network (RPN) to generate potential objects in the backend.

In addition to previously mentioned works, Facebook AI created a foundational model named Segment-Anything-Model (SAM) [64] which can perform open-set segmentation on a variety of granularities. This model is then extended to perform downstream tasks such as tracking in [65] and semantic grounding in [66]. In addition, there are modifications to the model that improve the inference time by orders of magnitude[67][68].

Chapter 2

Background

This chapter reviews some core concepts used in this system. Understanding these works is helpful when trying to understand the strengths and weaknesses and future areas of improvement.

2.1 SLAM as an Optimization Problem

The transition from filter-based SLAM to factor graph-based SLAM allows a more flexible problem formulation that can be modified to create a better state estimate. In this section, we will review the factor graph formulation of SLAM and how a maximum-a-posteriori (MAP) estimate can be obtained from it.

2.1.1 Factor Graphs

A factor graph, as described in [69] is a bipartite graph that contains two types of nodes: variable nodes and factor nodes. Variable nodes represent the state variables of the system, and factor nodes represent the constraints between the variables. It allows us to describe a complex joint probability distribution as a product of many smaller conditional probability distributions. The smaller distributions can be derived from first principles or experimental data, which makes the factor graph formulation easy to create. Figure 2.1 shows an example factor graph for a simple SLAM problem.

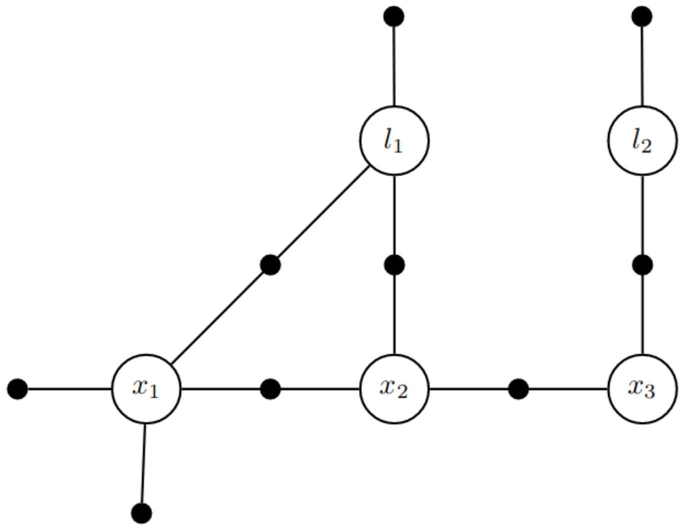


Figure 2.1: Example of a simple factor graph from [69]. The factor graph shows two landmarks and three poses, with multiple factors connecting them. The factors include pose-pose odometry factors, pose-landmark observations, and prior factors. We will use this framework to describe the relation between observations and variables in our system.

The structure of the factor graph also allows clever techniques to be used in the variable elimination process, which is used to obtain the MAP estimate. The approximate minimum degree (AMD) algorithm[70] is one such technique that can be used to minimize the fill-in of the resulting sparse matrix, which reduces the computational complexity of the variable elimination process. This is especially important for large-scale problems, where the number of variables and factors can be in the hundreds of thousands.

2.1.2 MAP Estimate

When we are given a set of measurements $\mathbf{Z}$, we can determine a state $\mathbf{X}$ that is a “best estimate” of the given measurements. A commonly used “best estimate” is one that maximizes the likelihood of the measurements being what we observed when given the state, also

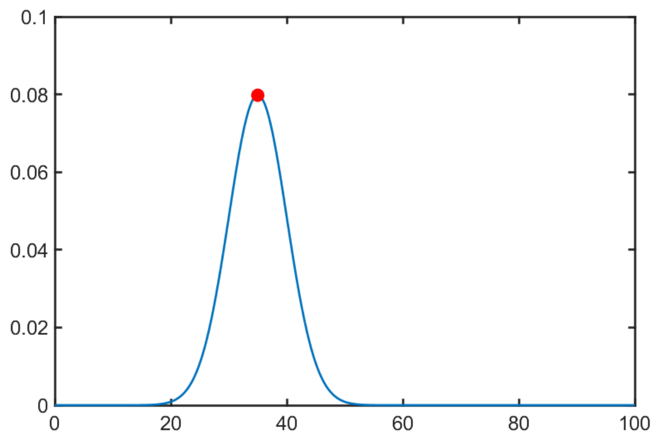


Figure 2.2: Example of a likelihood function. The likelihood function is a probability distribution that describes the probability of observing a measurement given a state. The likelihood function is used to calculate the maximum likelihood estimate, which is shown as a red dot in this figure.

known as the maximum likelihood estimate (MLE):

$$\mathbf{X}_{MLE} = \arg\max_{\mathbf{X}} P(\mathbf{Z}|\mathbf{X}) \tag{2.1}$$

We can use Bayes' theorem to derive the following formulation which we call the maximum-a-posteriori estimate.

$$\begin{aligned} \mathbf{X}_{MAP} &= \arg\max_{\mathbf{X}} P(\mathbf{X}|\mathbf{Z}) && (2.2) \\ &= \arg\max_{\mathbf{X}} \frac{P(\mathbf{Z}|\mathbf{X})P(\mathbf{X})}{P(\mathbf{Z})} && (2.3) \\ &= \arg\max_{\mathbf{X}} P(\mathbf{Z}|\mathbf{X})P(\mathbf{X}) && (2.4) \end{aligned}$$

Since $P(\mathbf{Z})$ is constant, it acts as a normalizing factor that does not affect the $\arg\max$ operation. We can now think of the MAP estimate as the MLE estimate that takes into account a prior on the state of the robot.

2.1.3 Nonlinear Least Squares

After the previous section, we arrive at a maximization problem where we would like to find the state that maximizes a conditional probability. That problem can be rearranged such that it is a product of factors ϕ such that each factor represents a probability density of a measurement conditioned on the state. For example, if we have two measurement $\boldsymbol{z}_1, \boldsymbol{z}_2$ between two variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the MAP estimate would be

$$\arg\max_{\boldsymbol{x}_1, \boldsymbol{x}_2} P(\boldsymbol{z}_1|\boldsymbol{x}_1, \boldsymbol{x}_2)P(\boldsymbol{z}_2|\boldsymbol{x}_1, \boldsymbol{x}_2)P(\boldsymbol{x}_1, \boldsymbol{x}_2) \tag{2.5}$$

. Assuming unbiased Gaussian noise in our sensors, we can expand $P(\mathbf{Z}|\mathbf{X})$ as follows:

$$P(\mathbf{Z}|\mathbf{X}) \propto \prod_i \exp\left\{-\frac{1}{2}\|h(\boldsymbol{x}_i) - \boldsymbol{z}_i\|^2_{\Sigma_i}\right\} \tag{2.6}$$

where $h(\boldsymbol{x})$ is the measurement function and Σ is the covariance of that measurement. The covariance is factored into the loss by the Mahalanobis distance, which can be thought of as a weighted sum of the vector norm.

Since there are multiple factors, the objective function is a product of the probabilities, which can be simplified further by applying the negative logarithm and transforming the product into a sum and the maximization problem into a minimization of negative sums.

$$L(\mathbf{Z}, \mathbf{X}) \propto \sum_i -\log\left\{\exp\left\{-\frac{1}{2}\|h(\boldsymbol{x}_i) - \boldsymbol{z}_i\|^2_{\Sigma_i}\right\}\right\} \tag{2.7}$$

$$\propto \sum_i \|h(\boldsymbol{x}_i) - \boldsymbol{z}_i\|^2_{\Sigma_i} \tag{2.8}$$

$$\propto \sum_i \|(h(\boldsymbol{x}_i) - \boldsymbol{z}_i)\Sigma_i^{-\frac{1}{2}}\|^2 \tag{2.9}$$

Now the problem takes the form of a nonlinear least squares:

$$\mathbf{X}_{MAP} = \arg\min_{\mathbf{X}} \sum_i \|(h(\boldsymbol{x}_i) - \boldsymbol{z}_i)\Sigma_i^{-\frac{1}{2}}\|^2 \tag{2.10}$$

which can be solved using a variety of iterative methods, such as Gauss-Newton (GN), Levenburg-Marquardt (LM)[71], or Powell's Dogleg (PD)[72]. The underlying mechanisms are similar, with all three first identifying a direction of descent and then determining a step size to traverse in the domain.

2.1.4 Linearization Process

As previously mentioned, iterative minimizers first determine the direction of descent for a linearized problem and then determine an appropriate step size. If we have an objective function

$$f(\boldsymbol{x}_i) = \|h_i(\boldsymbol{x}_i) - \boldsymbol{z}_i\|^2 \tag{2.11}$$

where $h(\boldsymbol{x})$ is a nonlinear measurement function, the linearized version will be

$$f(\boldsymbol{x}_i) \approx \|h_i(\hat{\boldsymbol{x}}_i) + H_i\Delta_i - \boldsymbol{z}_i\|^2 \tag{2.12}$$

, where Δ_i is an update vector, and the measurement Jacobian $H_i = \left.\frac{\partial h_i(\boldsymbol{x}_i)}{\partial \boldsymbol{x}_i}\right|_{\hat{\boldsymbol{x}}_i}$. After some rearrangement of terms, we are left with a linear system where the solution Δ_i minimizes the linearized objective function above.

$$H_i\Delta_i - (\boldsymbol{z}_i - h_i(\hat{\boldsymbol{x}}_i)) = A\Delta_i - b = 0 \tag{2.13}$$

If the solution to this system is used as is, we may run into issues with convergence where the objective function is very flat. To improve the convergence speed while not relying on a quadratic approximation, LM uses a slightly different formulation.

$$\left[A^{\mathsf{T}}A + \lambda\,\mathrm{Diag}(A^{\mathsf{T}}A)\right]\Delta_{lm} = A^{\mathsf{T}}b \tag{2.14}$$

The non-negative scaling factor λ is used to define the trust region and modify the step size depending on the landscape of the nonlinear objective function. The update Δ_{lm} is applied to the state and this process is repeated until a convergence criterion is reached.

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^{t} + \Delta_{lm} \tag{2.15}$$

2.2 Measurement Models and Error Functions

We use three different sensors in our system: an odometry system, a simple projective camera with an object detector, and a depth camera. In this section, we will review the measurement models for each of those sensors and their respective factors in the factor graph. Note that this section uses notation from lie algebra, a good review of it can be found in [69].

2.2.1 Odometry

An odometry measurement is a transformation from the previous pose $\boldsymbol{x}_{i-1}$ to the current pose $\boldsymbol{x}_i$. Since our system is in 3D, the transformation T^i_{i-1} resides on the SE(3) manifold and is composed of a translation and a rotation, described by 3 parameters each. The translation component of the pose resides in the Euclidean space, so we a standard vector norm to describe its error. Given $\boldsymbol{x}_i = T_{wi}$, $\boldsymbol{x}_j = T_{wj}$, and measured odometry $\boldsymbol{z}_ij = \tilde{T}_{ij}$ we can calculate the odometry residual as follows:

$$\mathbf{e_t} = \|(t_j - \mathbf{R}_{ij}t_i) - \tilde{t}_{ij}\|^2_{\Sigma}ij \tag{2.16}$$

$$\mathbf{e_R} = \|\mathbf{R}_j^{\mathsf{T}} - \mathbf{R}_i\tilde{\mathbf{R}}_{ij}\|^F_{\Sigma}ij \tag{2.17}$$

Note that we use the Frobenius norm instead of an L2 norm for the rotation error. This corresponds to a Langevin distribution for rotational error. Alternatively, we can use the local parameterization of SE(3) to calculate the error in the tangent space of the manifold. This is useful for the optimization process since the tangent space is a vector space and the error can be represented as a vector. The tangent space of SE(3) is $\mathfrak{se}(3)$, which is a six-dimensional vector space. The error in the tangent space is then calculated as follows:

$$\mathbf{e_{R,t}} = \|\mathrm{Log}(\mathbf{T_j^{-1}T_i\tilde{T}_{ij}})\|^{\mathbf{2}} \tag{2.18}$$

where Log denotes the Logarithm map from SE(3) to $\mathbb{R}^3$. Using this method, we can convert an error on the manifold to its local parameterization which is a vector space. This allows us to use a standard vector norm to calculate the error. Figure 2.3 visualizes the relationship between a manifold and its tangent space. The magnitude of the vector in the tangent space is used to calculate the error.

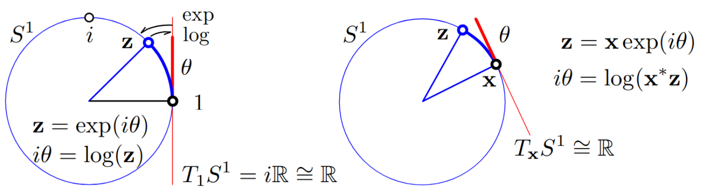

S^1
i
$\mathbf{z}$
exp
log
θ
1
$\mathbf{z} = \exp(i\theta)$
$i\theta = \log(\mathbf{z})$
$T_1S^1 = i\mathbb{R} \cong \mathbb{R}$
S^1
$\mathbf{z}$
θ
$\mathbf{x}$
$\mathbf{z} = \mathbf{x}\exp(i\theta)$
$i\theta = \log(\mathbf{x}^*\mathbf{z})$
$T_\mathbf{x}S^1 \cong \mathbb{R}$

Figure 2.3: Example of a manifold and its tangent space at a point[73]. The circle represents the S^1 manifold and θ is a vector in the tangent space of S^1 that represents $\mathbf{z}$ on the manifold. In our case, the manifold used is the SE(3) and SO(3) manifold, and the tangent space is $\mathfrak{se}(3)$ and $\mathfrak{so}(3)$.

Since we assume that the odometry measurements arrive as a transformation from the previous pose to the current pose, this way of measuring error is the most natural given the limited information. If we decide to expose the internals of our odometry method, it may become advantageous to use another error function.

2.2.2 Projective Camera

A camera captures an image by projecting light rays from a 3D point in space onto a pixel in its imaging plane. Along the way, the light ray is subject to distortion caused by the lens, and the imaging sensors themselves can also introduce uncertainty to the result. While there are many ways to parameterize a camera, we use the pinhole camera model which is shown in Figure 2.4.

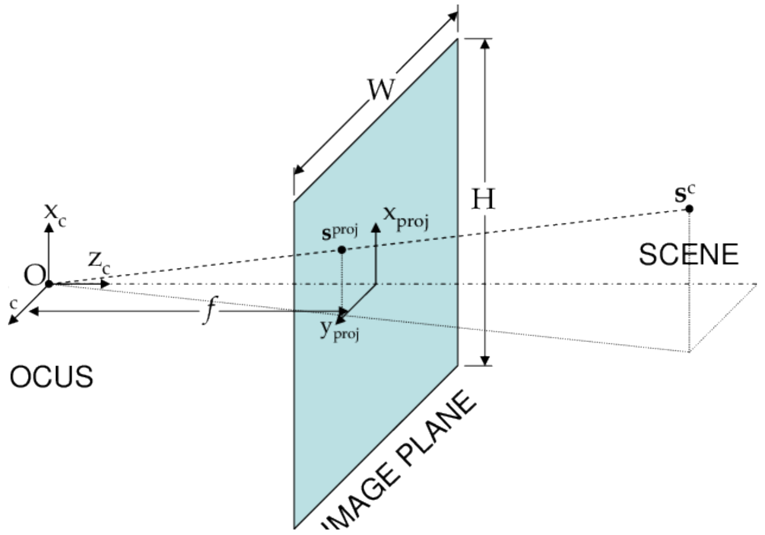


Figure 2.4: The pinhole camera model[74]. This model describes a linear relationship between the point in the world coordinate frame and the projected point on the image plane. It assumes a single scalar focal length for each axis and a constant offset between the principal point and the center of the image.

The measurement model of a pinhole camera is a linear transformation between the point in the world and the point on the sensor. That transformation is usually denoted as P, which stands for the projection matrix. The projection matrix is composed of the intrinsic and extrinsic matrices of the camera, the former contains the focal length f_x, f_y, the principal point (c_x, c_y), and the skew parameter s while the latter contains The extrinsic parameters are the rotation $\mathbf{R}$ and translation t of the camera with respect to the world frame. The measurement model is then given by the following equation:

$$k\boldsymbol{z}_i = P_i\boldsymbol{\ell}_i \tag{2.19}$$

$$k\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_i & t_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.20}$$

One can see that the measurement model first transforms the point from the world frame to the camera frame, and then projects the point onto the sensor. It is important to note that the resulting pixel location is in homogeneous coordinates, so we need to normalize it by dividing by the third component to obtain the actual pixel location. This normalization process is shown in the equation as the k term. We also assume an undistorted image is used.

Using this measurement model, we can calculate the reprojection error for any measurement by finding the difference between the predicted pixel location and the actual pixel location. The predicted pixel location is calculated by projecting the landmark location $\boldsymbol{\ell}_i$ onto the sensor using the measurement model. The actual pixel location is obtained from the object detector. The reprojection error is then calculated as follows:

$$\mathbf{e}_{\mathbf{R},\mathbf{t},\boldsymbol{\ell}} = \|\mathbf{P_i}\boldsymbol{\ell}_\mathbf{i} - \tilde{\mathbf{z}}_\mathbf{i}\|^\mathbf{2} \tag{2.21}$$

2.2.3 Depth

The depth measurement from the camera is calculated as the distance along the optical axis from the camera to the landmark. The measurement model, shown in Figure 2.5 is then given by the following equation:

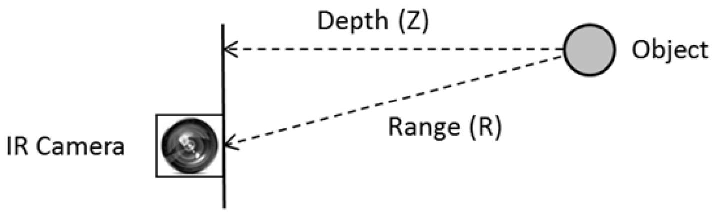


Figure 2.5: Definition of depth measurement from[75]. Note that depth differs from range in that it is the distance along the optical axis from the camera to the landmark, while range is the distance from the camera to the landmark along the line of sight.

$$\boldsymbol{z}_i = [0, 0, 1] \cdot \boldsymbol{x}_i^{-1} \boldsymbol{\ell}_i \tag{2.22}$$

We transform the landmark from the world frame to the camera frame, and then take the third component of the vector. The corresponding error function is then given by:

$$\mathbf{e}_{\mathbf{R},\mathbf{t},\boldsymbol{\ell}} = \|[\mathbf{0}, \mathbf{0}, \mathbf{1}] \cdot \mathbf{x}_\mathbf{i}^{-\mathbf{1}} \boldsymbol{\ell}_\mathbf{i} - \tilde{\mathbf{z}}_\mathbf{i}\|^\mathbf{2} \tag{2.23}$$

2.3 Data Association with Ambiguity

The data association problem for semantic SLAM is essential for the performance of the system. In this section, we will review previous approaches to a multi-hypothesis data association problem and how we apply those techniques in our system. Specifically, we will look at mixture-based models to represent the ambiguity in the data association problem.

Our problem is formulated as follows for each observation at a given frame, we have a set of possible associations $\mathbf{D} \triangleq \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots\}$ where only one association may be active at a time. Each association represents the observation of a landmark and creates a projection factor in our factor graph. We can also add in the null hypotheses to represent an incorrect association or an outlier detection, but that is not implemented in this system.

The optimization problem then becomes:

$$\mathbf{X}_{MAP} = \underset{\mathbf{X},\mathbf{L},\mathbf{D}}{\arg\max}\, P(\mathbf{X}, \mathbf{L}, \mathbf{D}|\mathbf{Z}) \tag{2.24}$$

. We can simplify this problem by marginalizing out the associations and implicitly representing them in the optimization problem by using a mixture model. One important thing to note is that the associations we use do not depend on discrete variables such as the class of a detected object. Doherty et al.[76] has developed a performant library that can optimize over both discrete and continuous variables using alternating minimization. That is especially useful for a system where the confusion matrix of the object detector is known and used for the optimization process. However, since we do not have explicit classes, we can use a simpler mixture model that can work with classical nonlinear least square optimizers.

2.3.1 Factor Graph Representation of Ambiguity

The factor graph representation of the ambiguity is shown in Figure 2.6. Instead of a factor connecting the corresponding landmark and pose pair, we now have a factor that connects a pose to multiple landmarks.

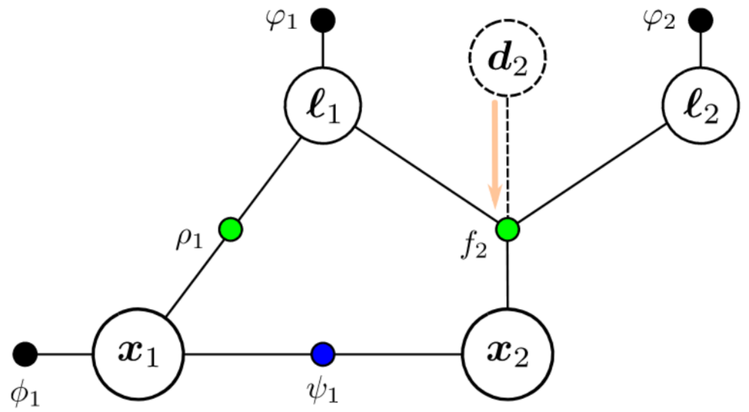


Figure 2.6: Factor graph representation of the ambiguity in the data association problem[77]. A factor graph containing an ambiguous association between $\boldsymbol{x}_2$ and $\boldsymbol{\ell}_1$, $\boldsymbol{\ell}_2$. The discrete decision variable d_2 was marginalized into the factor f_2 to create a max mixture factor.

2.3.2 Mixture Models

The max mixture factor is as follows (equation 13 in [78])

$$\phi\left(\boldsymbol{x}_t, \boldsymbol{\ell}_{\mathcal{H}}\right) = \max_{j \in \mathcal{H}} p\left(\boldsymbol{z}_{tk} \mid \boldsymbol{x}_t, \boldsymbol{\ell}_j\right) p\left(d_{tk} = j \mid \mathbf{Z}^{-}\right) \tag{2.25}$$

Where ϕ represents the factor, $\boldsymbol{z}$ is the measurement, and $p\left(d_{tk} = j \mid \mathbf{Z}^{-}\right)$ represents the discrete variable's probability distribution given previous measurements. We can marginalize the discrete variable to obtain the following factor:

$$\phi\left(\boldsymbol{x}_t, \boldsymbol{\ell}_{\mathcal{H}}\right) = \max_{d} p(\boldsymbol{z}|\boldsymbol{\ell}_d, \mathbf{X}) p(\boldsymbol{d} = d|\mathbf{Z}^{-}) \tag{2.26}$$

We can further simplify using our uniform prior assumption on the association variable to obtain the following factor:

$$\phi\left(\boldsymbol{x}_t, \boldsymbol{\ell}_{\mathcal{H}}\right) = \max_{d} p(\boldsymbol{z}|\boldsymbol{\ell}_d, \mathbf{X}) \tag{2.27}$$

Which can be easily implemented in the GTSAM framework as an error function that iteratively minimizes the error over all possible associations. Note that there does not exist a specific reason to assume a uniform distribution of the association variable, but it is a reasonable assumption given the lack of information.

For the sum mixture model, we can iterate through all the factors and sum the error over all possible associations. This is equivalent to the following factor:

$$\phi\left(\boldsymbol{x}_t, \boldsymbol{\ell}_{\mathcal{H}}\right) = \sum_{d} p(\boldsymbol{z}|\boldsymbol{\ell}_d, \mathbf{X}) \tag{2.28}$$

given the same uniform prior on the association variable. This is also known as the expectation maximization algorithm (EM).

2.4 Visual-Language Foundational Models

Foundational models are a relatively recent development in the field of computer vision. They are trained on internet-scaled datasets and promise to be a model that generalizes very well to rare and unseen data. Some models are trained specifically on image or image-caption pairs, which we call visual and visual-language foundational models. In this section, we will review CLIP and DINO, and how their outputs can be used in our system.

2.4.1 CLIP

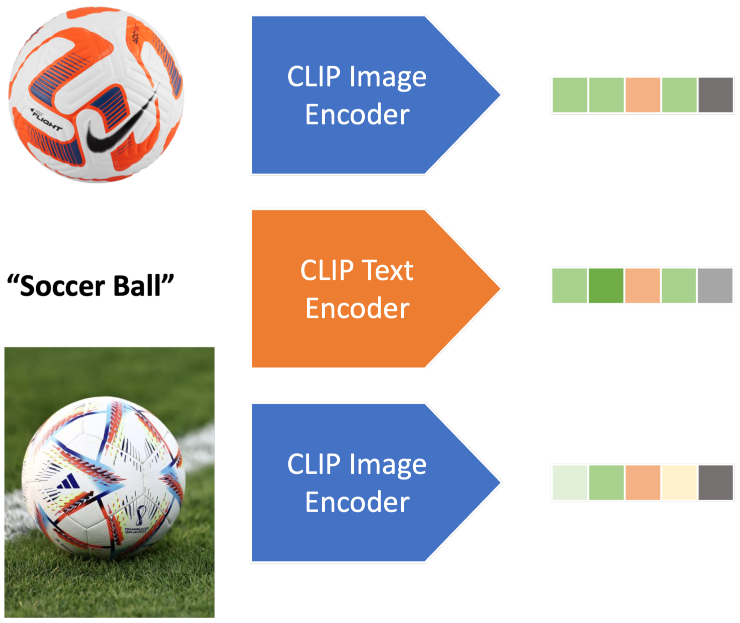


Figure 2.7: Example of CLIP embeddings from [48]. We use colored squares to visually represent the feature embeddings of the image and text. The embeddings are trained such that the cosine similarity between the two embeddings is high if the image and text are semantically similar. Since the second image includes a grassy background, the cosine similarity between the image and text embeddings should be lower than the image of just the soccer ball.

Contrast Language-Image Pre-training (CLIP) is a model developed by OpenAI[48] that is trained on 400 million image-caption pairs. It is trained to predict which caption belongs to which image, and the resulting model can be used to generate embeddings for both images and text. The embeddings are trained such that the cosine similarity between the two embeddings is high if the image and text are semantically similar. This allows us to use the model to generate embeddings for images and text, and then compare the embeddings to determine if the image and text are semantically similar. The text encoder is a Transformer and the image encoder can be either a vision Transformer (ViT) or a ResNet. One important thing to note is that CLIP outputs a single embedding for each image. Figure 2.7 shows an example of how the CLIP image and text encoders transform an image or string of characters into a common embedding space.

2.4.2 DINO

An alternative foundational model is DINO[49], which is a self-distillation technique that was tested on both ResNet and Vision Transformer (ViT) architectures. In the paper, the authors claim that this method is able to learn visual features that can be used for downstream tasks like classification and segmentation. The model's attention maps are interpretable and one can see the outlines of an object in them. Figure 2.8 shows an example of the attention map for one of the heads on two input images, originally published in the paper. Since the model outputs dense maps, we can use the output to create pixel-wise embeddings for the image. Advancements using DINO as a base have been created, with works using the language grounding to create open-set object detectors that follow a prompt[79], using a parallel network to perform pixel-wise segmentation[80]

In this chapter we reviewed the factor graph formulation of SLAM and previous systems that performed SLAM using object-based representations. We also reviewed the measurement models for the sensors used in our system and how they are represented in the factor graph. Finally, we reviewed foundational models and how they can be used to create a semantic representation of the world. In the next chapter, we will describe our system in detail and how it performs data association using a max-mixture model.

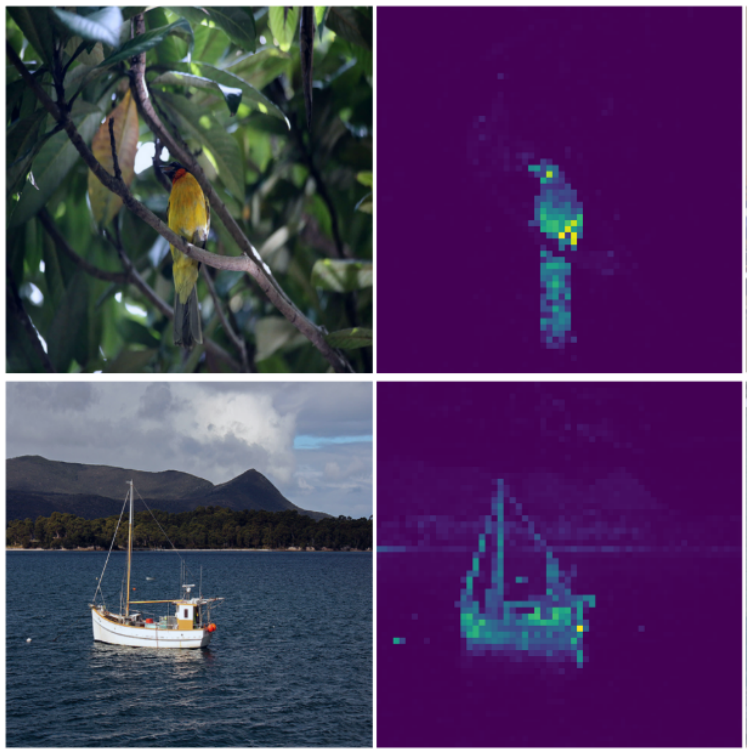


Figure 2.8: Example of DINO attention maps from [49]. The attention maps are interpretable and one can see the outlines of an object in them. Since the descriptors are dense, one would need to create a clustered representation to use them in our system. This is difficult due to similarities between the same parts of different objects, such as the wheels of a car and the wheels of a bicycle.

Chapter 3

System

Next, we will describe the system in detail. We will first describe the measurement sources, then the matching algorithm, and finally the optimization algorithm. We will also describe the language-aligned functions that leverage the underlying visual-language model to add additional constraints to the system such as landmark searching and filtering.

3.1 Overview

Our system follows the structure of a traditional sparse object-based landmark SLAM system. The system can be broken down into the frontend and the backend, which are responsible for the data association (aside from multi-hypothesis data association) and the optimization respectively. The front end is further broken down into the measurement sources, the matching algorithm, and the object initialization. The system diagram is shown in Figure 3.1. In the following sections, we will describe each block in the system diagram in detail, as well as point out some of the challenges an open-set system imposes on the system design that is not present in a closed-set system.

The input to our system is an odometry source that provides the pose of the robot at each time step and an object detection source that provides the object detections at each time step. The output of our system is a set of object landmarks and their poses in the world frame. The object landmarks are initialized at the first time step they are detected and optimized as the robot moves through the environment. For the odometry source, we extract the difference in pose from the previous timestep, $T_{X_n}^{X_n+1}$ [Block 1]. For object detection, we extract the bounding box, feature descriptor, and region proposal score [Block 2]. Depending on the configuration of the system, we may also extract a depth estimate. The odometry measurement is directly added to the optimization problem [Block 9]. The object detection is then filtered and matched to the object landmarks [Block 4, 5]. If the object detection is matched to an existing object landmark, the measurement is added to the optimization problem. If the object detection is not matched to an existing object landmark, the detection will then be matched to a set of candidate objects [Block 6]. Candidate objects are objects that are not in the optimization problem until they are successfully initialized based on one of two initialization methods. However, they may also be discarded if the objects are not consistently observed or the observations contain outliers. Candidates are initialized through either multi-view triangulation [Block 7] or depth-based initialization [Block 8], and if successful, added to the optimization problem [Block 9]. The optimization problem is then solved, and the object landmarks are updated.

One important aspect of a SLAM system is loop closure. Loop closure is the process of detecting when the robot has returned to a previously visited location. In sparse systems, it is often accomplished by having a visual place recognition (VPR) system that triggers a matching process when the likelihood of a match is above a threshold. In ORBSLAM, a visual-bag-of-words approach was used to initialize the loop closure. Some systems also include an outlier detection system that filters out bad loop closures via robust cost functions, which is useful in situations with visual aliasing such as in hallways or in front of doors. In our system, we do not have a VPR system and instead rely on matching between landmarks and current object detections. VPR systems using foundational models are an active area of work, and AnyLoc[81] has made significant progress in this area. However, we believe that the current state of the art is not yet robust enough for our system, and we leave this as

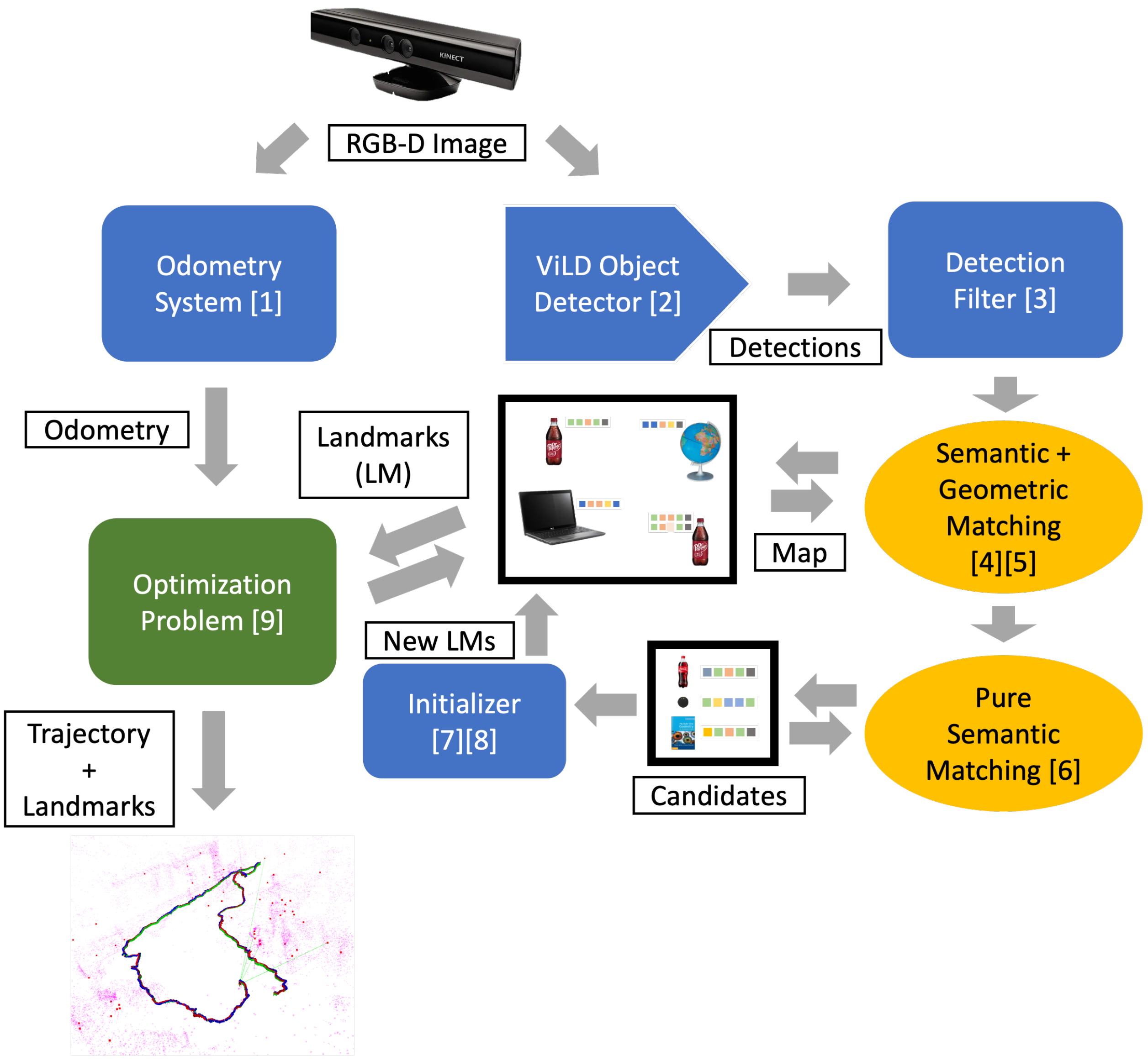


Figure 3.1: OSODA System Block Diagram. This diagram provides an overview of the different modules within our system and the flow of information from the input to the output. The input is provided by the RGB-D camera and the output is the optimized landmark positions and trajectory of the robot.

future work. We also do not have a robust outlier detection system and instead rely on the robust cost functions and max-mixture in the optimization problem to filter out bad loop closures. This is also an area of future work.

3.1.1 Robot Operating System (ROS)

Note that this system is written in the ROS[82] framework. Multiple nodes are run depending on the configuration. Figure 3.2 shows the nodes that are run for the system described in this paper and the message types associated with them. Some nodes are created for visualization purposes and are not necessary for the system to run. The nodes are described in detail in Table 3.1

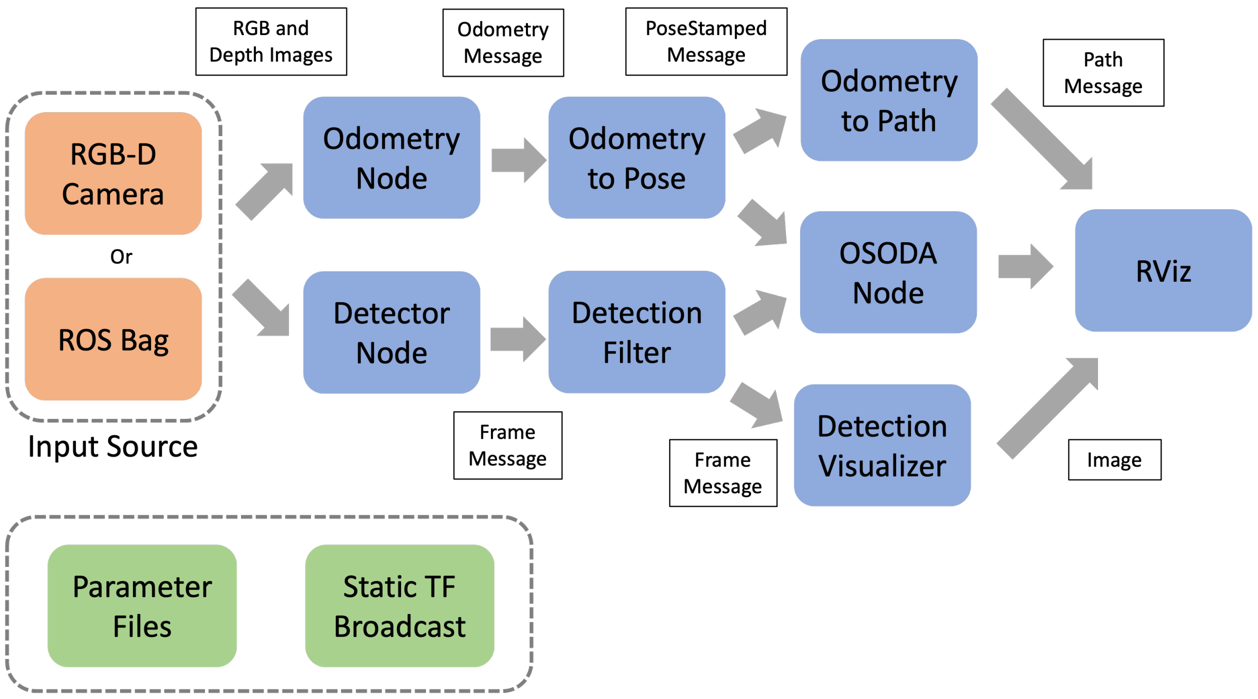


Figure 3.2: Graph of ROS nodes in the system. The input sources are in a box on the left, and the messages are passed between nodes as shown. A more detailed description of the nodes and message frequencies can be found in Table 3.1.

Node Name	**Input Type**	**Frequency**	**Output Type**	**Frequency**
Input Source	None	None	RGB Image Depth Image	30 Hz
Odometry Node	RGB Image Depth Image	30 Hz	Odometry	30 Hz
Odometry to Pose	Odometry	30 Hz	PoseStamped	30 Hz
Odometry to Path	PoseStamped	30 Hz	Path	30 Hz
Detector Node	RGB Image Depth Image	30 Hz	Frame	5 Hz
Detection Filter	Frame	5 Hz	Frame	5 Hz
Detection Visualizer	Frame	5 Hz	Image	5 Hz
OSODA Node	PoseStamped Frame	30 Hz 5 Hz	Path Line Markers Point Markers	30 Hz 5 Hz 5 Hz

Table 3.1: Detailed description of ROS nodes, and the messages passed between them as well as the frequencies.

3.2 Measurements

3.2.1 Odometry Source

A wide range of odometry sources can be used with our system. In our testing, we were able to use RTAB-Map RGB-D odometry[83], OpenVINS[84], and ORB-SLAM2[26] with loop closure disabled as our odometry source. The odometry source runs in its node and publishes the odometry with respect to the world frame at each time step. Only some of the odometry sources publish the covariance of the odometry, so we use a constant covariance set by a configuration file to simplify the problem. Intermediate nodes then convert the odometry or PoseWithCovarianceStamped message to a PoseStamped message, which is used by the system. For some datasets, only select odometry sources can be used due to the sensing modality. For example, the TUM RGBD dataset only has ground truth, RGBD, and an accelerometer at 10Hz. In this case, we can either use the ground truth or the RGBD odometry. The datasets collected in Stata used a Realsense D455 camera, which recorded stereo IR images, RGBD, and 6 DOF IMU at 400Hz, but we do not have a full ground truth trajectory. Therefore, RGBD Odometry or Visual-Inertial Odometry was used. There are also a couple of TUM RGBD datasets with wheel odometry, which we can use as well. The wheel odometry worked best when the environment was featureless or beyond the range of the RGBD sensor.

	Sensor Modality			**Compatible With**	
Odometry Source	**RGB**	**Depth**	**IMU**	**TUM**	**D455**
RTAB-Map RGB-D	✓	✓		⋆	✓
OpenVINS	✓		✓		✓
ORBSLAM2	✓	✓		⋆	✓
Wheel Odometry				⋆	

Table 3.2: Compatibility between odometry sources and sensor modalities. We also indicate if the source is compatible with sensors used in TUM and the Intel RealSense D455. ⋆ indicates that only some of the sequences are compatible.

RTAB-Map RGB-D Odometry

This method uses depth images from the camera to perform incremental closest point (ICP) point cloud registration. While simple, this method can reliably produce odometry for most trajectories tested. Due to the nature of ICP algorithms, the discrepancy between two adjacent frames must not exceed some threshold for the odometry to be accurate. This is not a problem for most trajectories but can be a problem for trajectories with fast motion or large rotations. This method also does not produce a covariance, so we use a constant covariance set by a configuration file to simplify the problem. Experimentally we determine that frame-to-frame ICP, where the current frame is matched to only the immediate previous point, is sufficient for our case, although RTAB-Map also offers a frame-to-map ICP method where a local pointcloud map is kept to be matched against. The latter is more robust to fast motion and large rotations but is also more computationally expensive.

OpenVINS

This odometry source uses a visual-inertial odometry (VIO) algorithm to produce odometry. The VIO algorithm uses a sliding window of IMU and camera measurements to produce a pose estimate. The VIO algorithm can produce a covariance for the pose estimate, which we do not use in our system for simplicity. One advantage of this odometry source is that it does not get lost in featureless environments or when there are large amounts of motion blur. The IMU is able to provide short-term estimates of odometry and aid in scale estimation. However, using VIO requires a carefully calibrated camera and IMU, as well as the calibrated extrinsics between the two. We used Kalibr[85] to perform all of those calibrations. The instructions on their repository were sufficient. The only adjustment to the calibration we made was to increase the IMU noise figures by a factor of 100. This odometry source also requires a certain magnitude of jerk in order to initialize and perform scale estimation, the magnitude was changed to 1.5 in the configuration files, and the duration was reduced to 1 second.

ORB-SLAM2

In some datasets where the RGB-D data does not allow for reliable odometry from RTAB-MAP, we resort to using a modified version of ORB-SLAM2[26] as our odometry source. ORB-SLAM2 can work on RGB-D images to provide a scale-accurate trajectory based on tracking point landmarks in the environment with ORB feature descriptors. As a full SLAM system, ORB-SLAM2 also performs local and global bundle adjustment, detects and performs loop closures, and maintains a map of the environment. However, these functions would disqualify it from being used as an odometry system as it is able to correct for drift. Therefore, we disabled the loop-closure module and severely limited its mapping capabilities to have it act as a visual odometry system. Local bundle adjustment is still enabled, which makes it a fixed-lag smoother. To confirm the efficacy of our modifications, we noticed that it does accumulate drift and does not perform any loop closures on the dataset that we tested on.

3.2.2 Object Detection

Since this is an open-set system, our definition of an object is much more general than closed-set methods. We define an object to be any closed shape that can be repeatably detected in the environment and holds some semantic meaning. For example, the back of an office chair could be counted as an object by our definition, but will not likely be its own class in a closed-set detector. For each object, we require a bounding box, a segmentation mask, a confidence score to describe the likelihood of it being what we consider an object, what we call objectness, and a feature descriptor. This is very similar to a closed set descriptor output, except that a closed set descriptor would output a class label instead of a feature vector. In the following sections, we will describe the detection pipeline and give examples of what the output looks like at each stage. We will also perform a qualitative comparison between the output of our open-set detection pipeline and a closed-set detector.

ViLD

The open-set object detector we use is ViLD[63], which is an object detector that uses a region proposal network (Mask R-CNN with ResNet FPN backbone) and a CLIP-aligned descriptor generating head. When given an image, it will first create masks that could describe an object, and feed the masked images into the CLIP aligned head to generate a 512-dimensional feature vector. The top 1000 masks and features are returned. Due to the way this model is trained, the descriptor it outputs resides in the same latent space as descriptors generated with the CLIP model, but it also has the ability to generate the descriptor for only the object of interest, making it useful for our purposes. Figure 3.3 shows an example of the output from the network. To avoid clutter, we only visualize 200 detections. The output from the detector is then fed into the detection filtering module.

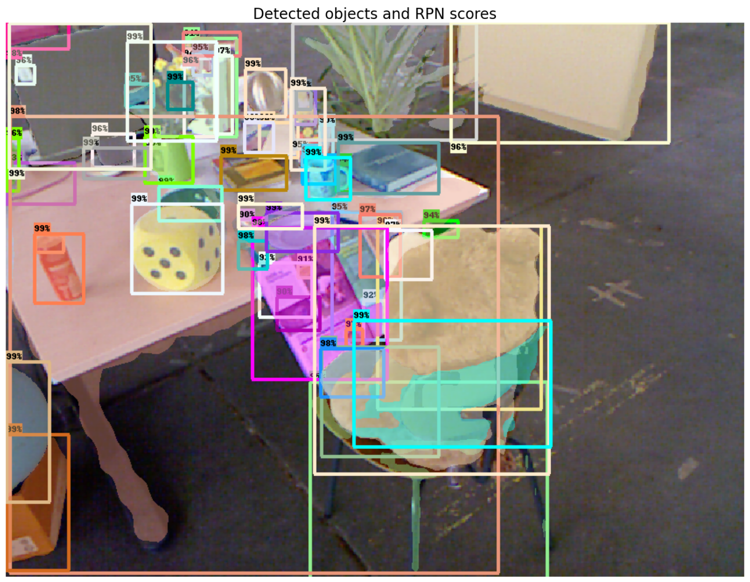


Figure 3.3: An example of the output from ViLD. We selected 100 masks and bounding boxes to be displayed, we have already performed filtering by region proposal network score and non-maximal suppression. Note that this model accurately segmented the objects but suffers from a granularity issue: the top of the Coke can is detected as a separate object from the whole can, the back of the chair is separate from the legs, and some paragraphs on the magazine is detected as its own object instead of a part of the magazine.

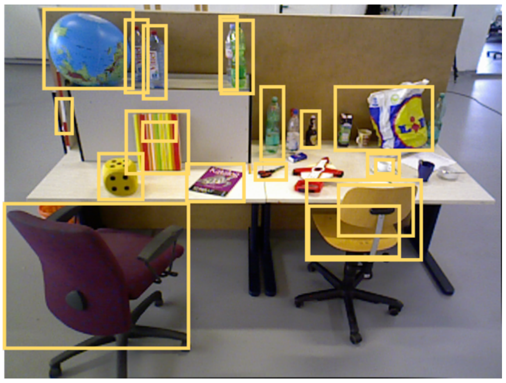
Using Closed Set Descriptors

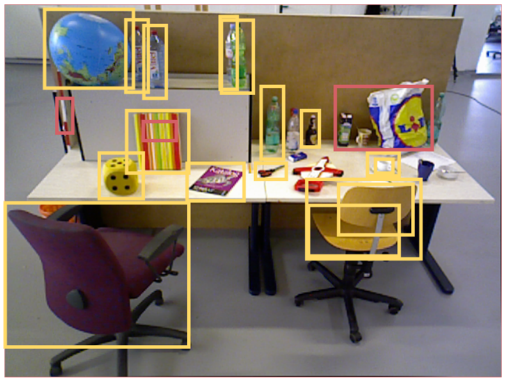
One interesting property of our system is that it can also use closed-set detectors. We convert the detected class to a feature vector using a one-hot encoding, such that two objects of the same class have the same feature vector and high cosine similarity. This allows us to use closed-set detectors such as YOLOv8[10] with our system. This is useful when we want to compare the performance of our system with a closed-set system, or when we want to compare the performance of different closed-set detectors.

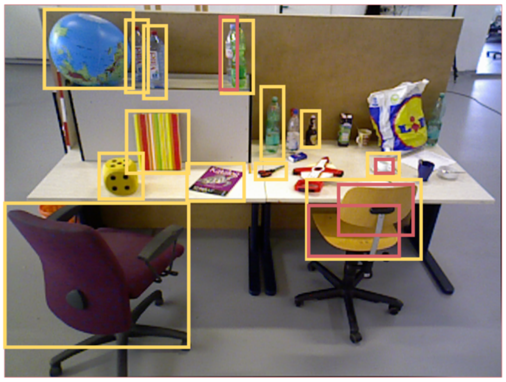
3.2.3 Detection Filtering

Objectness Threshold

All objects where the region proposal score is below a threshold are discarded. This is to remove objects that are not likely to be objects. The threshold is set by a configuration file


(a) Raw detections from the network.


(b) Detections after filtering by region proposal network score.


(c) Detections after non-maximal suppression.

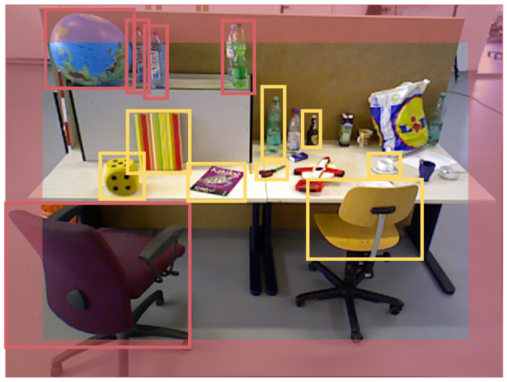

(d) Detections after edge filtering.

Figure 3.4: An example of detection filtering for the outputs of the object detector. All items filtered out during a step are outlined in red. The raw detections from the network are shown in 3.4a. The detections are then filtered by region proposal network score, and detections with a score below 0.95 are discarded. The remaining detections are shown in 3.4b. The detections are then filtered by non-maximal suppression, and detections with an IoU greater than 0.4 are discarded. The remaining detections are shown in 3.4c. The detections are then filtered by edge detections, and detections within 10 pixels of the edge of the image are discarded, the zone is shaded in red in the example image. The remaining detections are shown in 3.4d.

and is set to 0.95 by default. This threshold is high to ensure high precision in detections, it will need to be adjusted depending on the distinctness of objects in the environment and image quality. For example, the TUM RGB-D dataset has lots of high-contrast household objects, so the threshold can be set higher. In contrast, other datasets may have significantly higher motion blur and uncommon objects which require a lower threshold to ensure high recall. An example is shown in 3.4b.

Non-Maximal Suppression

Like closed-set detectors, we need to remove overlapping detections that may represent the same underlying object. We use a non-maximal suppression (NMS) algorithm to remove overlapping detections. The NMS algorithm is a greedy algorithm that iterates through the detections in order of confidence score and removes detections that have an IoU greater than a threshold. The threshold is set by a configuration file and is set to 0.4 by default. An example is shown in Figure 3.4c.

Edge Detections

We also remove detections that are too close to the edge of the image. Since we are using centroid representation for our objects, a partial observation will introduce a large amount of error in the centroid estimation. Therefore object detections that are within 10 pixels of the edge of the image are removed. There are alternative approaches to object representations that can allow for partial observations, such as Quadrics[37], but they are not explored in this work. An example of this is shown in Figure 3.4d.

Persistence

For unknown reasons, the ViLD detector produces occasional detections with high RPN scores but does not last for more than one frame, even if the camera view remains similar. To remove these spurious detections, we require that a detection have an IoU of greater than

0.5 with another detection within the last 5 frames to be considered valid. This is a heuristic that is not robust, and it would be good to remove it in the future.

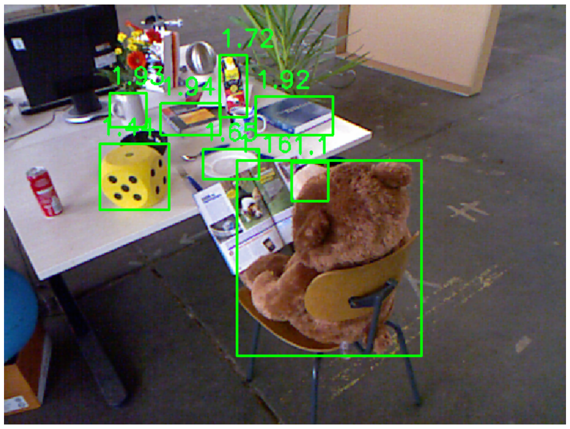
3.2.4 Filtered Detector Output

After the filtering step, we are left with a set of detections that are likely to be objects. This set of detections is then used in the matching step. To evaluate the quality of these open-set object detections, we compare them to the output of a closed-set detector.

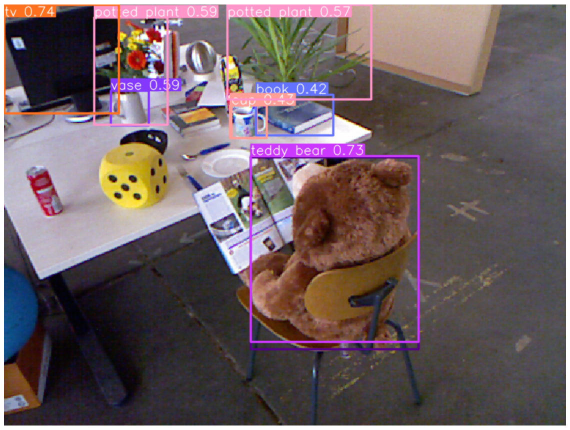
Figure 3.5 shows a comparison between open and closed-set detection results. The closed set detector was trained on Microsoft's COCO dataset, with labels including common office and household items as well as animals and vehicles. In these examples, we can notice some strengths and weaknesses of the open-set approach. The open-set detector is able to detect distinct items that may not have been explicitly trained on, such as the tape mark on the floor in 3.5c, the metallic sphere in 3.5e, and the large die in 3.5a. However, it also missed some obvious items like the keyboard in 3.5f or the potted plant in 3.5b and 3.5f. Finally, the issue of object granularity becomes apparent in this example, as the leg of the desk was detected as an individual object in 3.5c, which is not what a human would usually choose to detect. Our view on this issue is that as long as an object is able to be reliably detected across a variety of viewing angles, it should be included in the map since it will work well as a landmark for the robot.

3.2.5 Depth Estimation

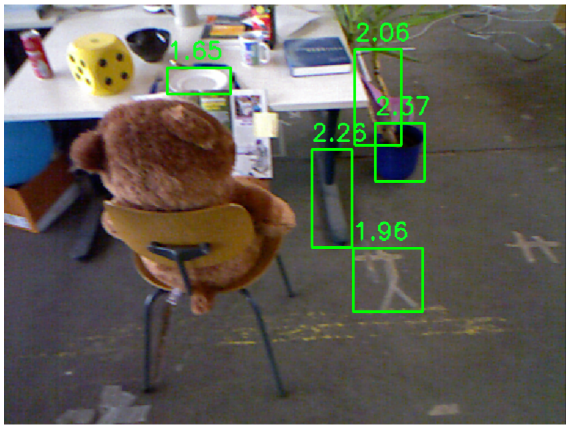
The segmentation mask is applied to the depth image to estimate the object's depth. This is only necessary if the object initializer uses depth instead of multi-view triangulation. The depth estimation is done by taking the mean of the depth values within the segmentation mask. This is a simple heuristic that works well enough for our purposes, however, it does not take into account the thickness of the object and could cause issues with observations from both the front and side of the objects. Ideally, this will be resolved by using a volumetric



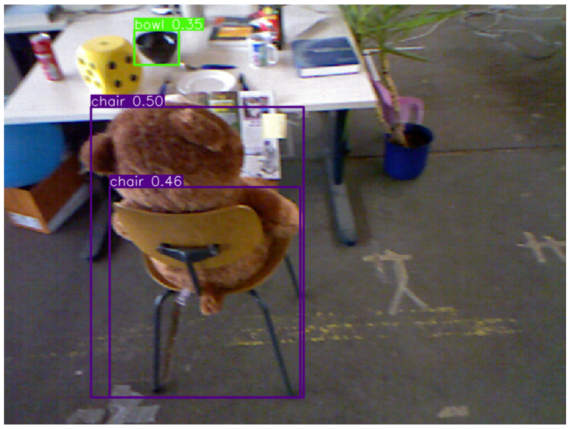
(a)



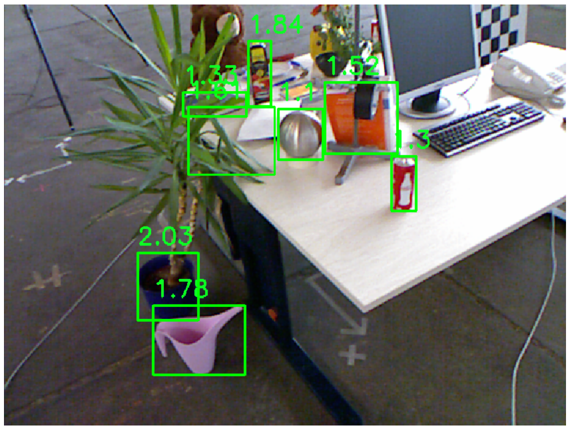
(b)



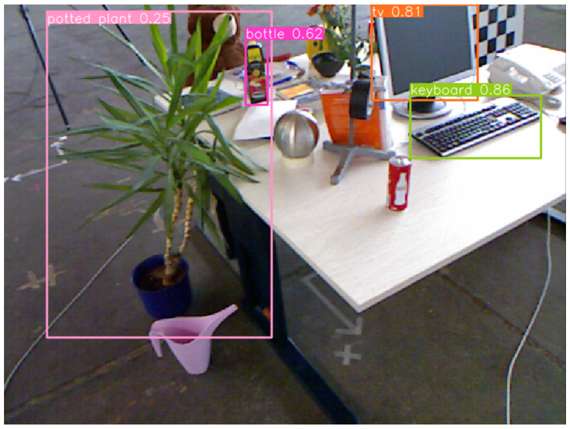
(c)



(d)



(e)



(f)

Figure 3.5: Comparison of ViLD and YOLOv8 on the TUM RGBD dataset. The left column shows the ViLD detections and the right column shows the YOLOv8 detections.

representation of the object, such as quadrics or cuboids, and the depth measurement is used as a measurement of the distance to the front surface of the object. Object detections without valid depths are discarded. Figure 3.6 shows an example of an RGB image and its corresponding depth image. The brightness of the depth image corresponds to the depth, with brighter pixels being further away from the camera. The upper left-hand corner of the depth image contains an incorrect depth that makes the detection have an erroneous 2.53 meter depth estimate. Much of the image is black, which means that they do not have any valid depth measurements. As a result, the depth estimation is not very reliable.

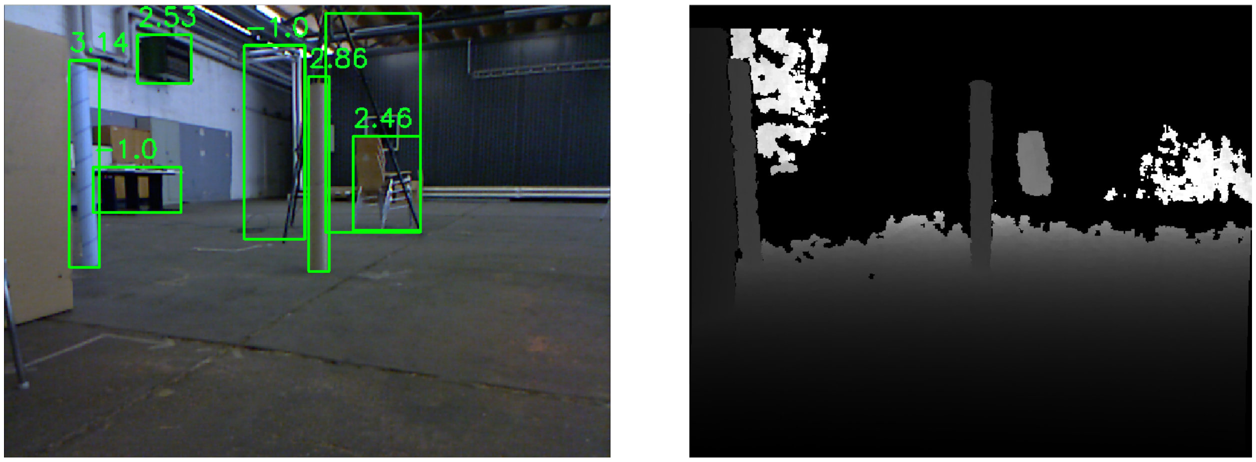


Figure 3.6: Example of an RGB image (left) and its corresponding depth image (right) from the TUM RGBD dataset. The depth image contains numerous incorrect depths and missing depths which makes it difficult to use for object depth estimation. This is due to the limitations of the Kinect sensor used to collect the dataset.

3.3 Objects

For object-based SLAM systems, we must accurately represent the object such that we can perform data association with high precision and recall. In a closed set system, the object is represented by a class label, which is a unique identifier for the object. However, in an open set system, we do not have a unique identifier for the object and therefore must use other methods to represent the object. We use a combination of a centroid and multiple feature descriptors to represent the object. The centroid is the average of the bounding box coordinates, and the feature descriptors are the CLIP embeddings of each detection associated with the object. The centroid is used to represent the position of the object, and the feature descriptor is used to represent the object's semantic information.

For an open-set system, it is essential to update an object's semantic information due to variations in feature descriptors across different views. This is a problem unique to open-set systems, as closed-set systems can rely on a viewpoint-independent class label.

3.3.1 Feature Descriptors

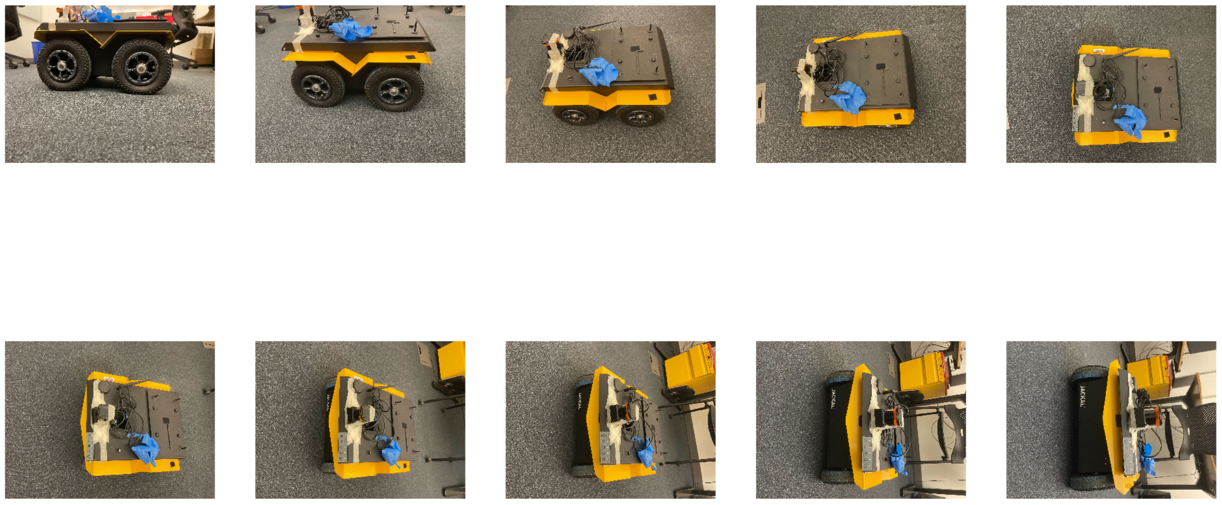
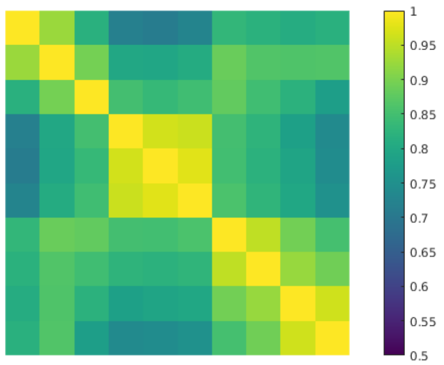


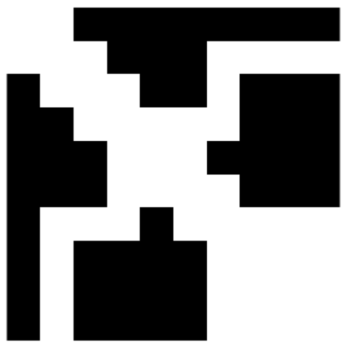
Figure 3.7: A collection of images of a robot on a gray-blue carpet. The robot is viewed from different angles, starting from the front and rotating to the top and the left side. This is an example of how different views of the same object can be observed during the course of traversing an unknown environment.

To illustrate the importance of maintaining a set of feature descriptors for each object, consider the following example.

Figure 3.7 shows a collection of views of a robot that could be encountered during a trajectory. We extracted the CLIP embedding from each image and calculated the pairwise cosine similarity between all images. Figure 3.8a shows the cosine similarity as a color map, where adjacent images are also adjacent in the plot. This shows how different observations of the same object can be significantly different in their feature descriptor depending on the viewpoint and lighting condition. To further illustrate this effect, Figure 3.8b shows a binarized version of that matrix with a threshold of 0.85. Black squares are instances where the two images would not be classified as the same object and white squares are instances where they are. It is the case that not all images will be correctly identified as the same object, which will lead to duplicate object detections and overall worse performance in our data association algorithm. However, we can use spatial-temporal information to capture all different views of the object and incrementally create a database of descriptors: since each adjacent image will be correctly identified, we can incrementally associate the new views to the previous view and therefore associate all views to the same object if we observe the views continuously.



(a) Cosine similarity between CLIP embeddings of different views of the same object. The images are ordered from left to right, top to bottom. The color map shows the cosine similarity between the CLIP embeddings of the images, with the color bar ranging from 0.5 to 1.



(b) Binary similarity between CLIP embeddings of different views of the same object. We used a threshold of 0.85 to determine if the two images are of the same object.

Figure 3.8: Cosine similarity between different viewpoints. CLIP embeddings are extracted from the images in 3.7 and the cosine similarity is calculated between all pairs of images.

3.3.2 Appending Observations to Objects

Once an association is made using one of the algorithms outlined in 3.4, we need to append the observation to the object and create the appropriate costs in the backend. The observation is appended to the object's list of observations, and a cost is created between the variable representing the camera position during the observation and the object's position in the world. The cost is calculated in one of two ways, depending on if the observation contains ambiguous associations.

Non-Ambiguous Associations

If the observation does not contain ambiguous associations, we create a projection factor between the camera pose $\boldsymbol{x}$ and the landmark position $\boldsymbol{\ell}$. The cost is calculated as the projective residual on the image plane between the bounding box center p and the projected landmark position. The cost takes into account a fixed camera intrinsics matrix, which is given by the calibration of the camera. The cost also takes into account the covariance, which is set to 10 pixels by default. In the following equation, we use π to represent the camera projection equation.

$$\phi(\boldsymbol{x}, \boldsymbol{\ell}, p) = \|\pi(\boldsymbol{x}, \boldsymbol{\ell}) - p\|_{\Sigma}^{2} \tag{3.1}$$

Ambiguous Associations

If the observation contains ambiguous associations, we create a max-mixture factor between the camera pose and the landmark position. For each potential match to a landmark, we calculate projective factors as described above. We then create a max-mixture factor between the camera pose and the landmark position, with the projective factors as the mixture components. One important thing to note is that we will not add the feature descriptor to the objects, as the descriptor is not uniquely associated with the object. If we were to add it, then it would lead to more ambiguity in the future as now multiple objects have the exact same descriptor.

$$\phi(\mathbf{X}, \boldsymbol{\ell}, p) = \min_{\boldsymbol{x} \in \mathbf{X}} \|\pi(\boldsymbol{x}, \boldsymbol{\ell}) - p\|^2_\Sigma \tag{3.2}$$

3.3.3 Descriptor Compression

Since our system appends descriptors to its corresponding objects after every matched observation, we potentially have an infinite memory footprint for the descriptors. This is not desirable as SLAM systems may be required to operate for an extended amount of time and on limited hardware. Therefore, we need to compress the descriptors to a fixed size. We do this by using a SVD decomposition of the descriptors, and only keeping the top k singular values. This technique is used in data compression, including image compression, and while it is fairly basic and does not offer the best compression, it is sufficient for our purposes. The number of singular values to keep is set depending on the desired level of descriptiveness, and there exists a tradeoff between descriptiveness and maximum memory footprint per object.

Using SVD for Descriptor Compression

The SVD decomposition of a matrix A is defined as follows:

$$A = U \Sigma V^T \tag{3.3}$$

where A is the original n by p matrix representing n observations each with a descriptor with p length. U is a n by n matrix, Σ is a n by p matrix, and V is a p by p matrix. Both U and V are orthogonal matrices, and Σ's diagonal entries, which are the singular values, are sorted in descending order. The first k singular values are kept. This will give us a truncated singular value matrix $\tilde{\Sigma}$ with shape n by p, and nonzero entries only on the first k diagonal entries. We can then use this matrix to create a reduced-rank approximation of the original matrix A:

$$\tilde{A} = U\tilde{\Sigma}V^T \tag{3.4}$$

where $\tilde{A}$ has shape n by p, but only the first k rows will be non-zero. Since we start with $n > k$, we have reduced the number of rows in the matrix and performed a dimensionality reduction. This preserves the k most important feature descriptors of this object.

One may recognize that we can decrease the computational complexity of this operation by only computing the first k by k submatrix of U and k by p submatrix of V^T. This will allow us to use a k by k diagonal matrix to represent $\tilde{\Sigma}$, and therefore only $O(k)$ matrix-vector products are required to compute $\tilde{A}$. This is a significant improvement over the $O(n)$ matrix-vector products required to compute $\tilde{A}$ without this optimization, especially when n is large. This optimization is used in our implementation.

$$\begin{aligned}
\tilde{A} &= U_{1:k,1:k}\Sigma_{1:k,1:k}V^T_{1:k,:} && (3.5)\\
&= \begin{bmatrix} u_{11} & \cdots & u_{1k} \\ \vdots & \ddots & \vdots \\ u_{k1} & \cdots & u_{kk} \end{bmatrix} \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{1k} \\ \vdots & \ddots & \vdots \\ v_{p1} & \cdots & v_{pk} \end{bmatrix}^T && (3.6)
\end{aligned}$$

A further optimization is to use a thin SVD algorithm to only calculate the first k entries of the singular values and vectors. We use `Eigen`'s implementation for our work[86].

After calculating $\tilde{A}$, we replace the descriptor of the object with rows or $\tilde{A}$ and proceed with the matching algorithm in the next iteration. The purpose of descriptor compression is to preserve the most distinct feature descriptors of the system while bounding the memory usage. One can estimate the number of descriptors needed for any given object by looking for distinct views of the object. A safe estimate is to have at least 6 views, one of each face of a cube that encapsulates the object. We used a k of 30 as a conservative estimate for the number of descriptors.

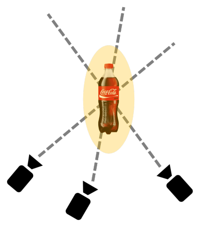
3.3.4 Object Initialization

Every filtered observation is either matched to an existing landmark in the map or added to a candidate object to potentially be initialized as a new landmark. However, since one of the modalities of object detection is monocular RGB, we may not have observability over the depth of the object. Therefore, we have two different methods of initializing objects, multi-view triangulation and depth-based initialization. Multi-view triangulation is used when we do not have observability over the depth of the object, and depth-based initialization is used when we do have observability over the depth of the object. To ensure the quality of our observations, we have several criteria outlined as follows: the object must be observed at least 3 times, the object must have a residual of less than 10 pixels during the initialization process, and the object must also not have any observations where the initialized position is behind the camera. The first two parameters are determined experimentally, and the last parameter is to ensure that all observations are valid when initialized. The last parameter is not necessary for depth-based initialization as it can be done after a single observation and does not require any triangulation. However, the depth of the estimate does not account for the thickness of the object and therefore is not as accurate.

Multi-View Triangulation

If we have multiple observations of an object from different perspectives, the object's position in space will be fully observable, assuming the views are not colinear. We can then use triangulation to estimate the object's position in the world frame by a variety of methods. For our case, we use the direct linear transform (DLT) [87] to perform the estimation, but one can use other methods such as nonlinear optimization.

To ensure the object has at least one pair of views that are not colinear, we calculate the angle between all vectors from the camera to the observed centroid in the world frame.


(a) A good multi-view initialization of an object.

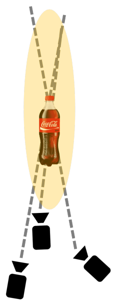

(b) A bad multi-view initialization of an object.

Figure 3.9: An example of a good and bad multi-view initialization. The dotted lines are the ray from the camera to the object, and the Coke bottle represents the initialized position of the object. The lines are less colinear in 3.9a, and they are more colinear in 3.9b. This means that the uncertainty along the Z direction for the second scenario is worse than the first.

If the pairwise angle is less than a threshold (10 degrees in our case), we consider the observation to be colinear. If the object has at least one pair of views that are not colinear, we proceed with the triangulation. Otherwise, we will wait until further observations are made. Figure 3.9 shows an example of a good and bad multi-view initialization. If the observations are arranged in a line, then the uncertainty along the Z direction will be very high, and the object will not be initialized in our system.

The object's position in the world frame is estimated by solving the following equation given n observations:

$$\begin{bmatrix} v_1\mathbf{p}_{13} - \mathbf{p}_{12} \\ \mathbf{p}_{11} - u_1\mathbf{p}_{13} \\ u_1\mathbf{p}_{12} - v_1\mathbf{p}_{11} \\ \vdots \\ v_n\mathbf{p}_{13} - \mathbf{p}_{n2} \\ \mathbf{p}_{n1} - u_n\mathbf{p}_{n3} \\ u_n\mathbf{p}_{n2} - v_n\mathbf{p}_{n1} \end{bmatrix} \mathbf{x} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.7}$$

for each of the observations and projection matrices, where $(u_i, v_i, 1)$ is the i^{th} projected point in homogeneous coordinates, and $\mathbf{p}_{ir}$ is the r^{th} row of the i^{th} projection matrix P. The solution is the point in $\mathbb{R}^3$ that minimizes the reprojection residual, which we denote as x in the equation.

We then check if the solution satisfies all of the criteria outlined above. If it does, we create a new object and add it to the optimization problem. If it does not, we discard the object and all observations associated with it.

Depth Based Initialization

This initialization method is comparatively simpler than multi-view triangulation but requires a depth estimate of the object. The depth estimate can be obtained from the depth image as shown in 3.2.5. The object's position is converted into the world frame by applying the following transformation:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = P^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{3.8}$$

where P is the camera projection matrix, and (u, v) is the centroid of the object in the camera frame, with z corresponding to the depth.

We then check if the solution satisfies all of the criteria outlined above and proceed accordingly.

Comparison of Initialization Methods

There exists a tradeoff between the speed and accuracy of initialization methods. With a higher threshold for consistent detection before an initialization attempt, we are able to have more confidence that the object being initialized will be reliably detected across different views and have no outliers in the series of detections. However, it also means that the observations made before the initialization attempt do not contribute to correcting the pose estimate of the robot and it can lead to increased drift and higher real-time error. On the other hand, with a lower threshold for consistent detection, we can initialize objects faster and have more observations to correct the pose estimate of the robot. However, it also means that the object being initialized may not be reliably detected across different views and may have outliers in the series of detections. This can lead to a higher chance of failure during the initialization process, and therefore a higher chance of introducing errors into the system. To make matters more complicated, the threshold for consistent detection depends on the environment and detector recall and accuracy. Better detectors will allow for a lower threshold as we will be able to initialize sooner based on fewer observations.

In our system, we use a threshold of 3 observations before an initialization attempt, which is a balance between the two extremes. We also use a threshold for the residual of the observations during the initialization process, which is a heuristic that is not robust and should be replaced with a more robust method in the future.

Another difference between the two methods is the maximum distance of an object from the camera. Since the depth sensor has a maximum range, and the quality of the depth estimation decreases with distance, we need to limit the maximum valid range to a constant. In our experiments, we found 5 meters to be a good threshold for a D455 camera. By contrast, the multi-view triangulation method does not have a maximum range and can initialize objects at any distance as long as the views are not colinear and there exists a large enough baseline.

Initializing with Ambiguous Associations

Another challenge that arises from open-set systems is that we may have ambiguous associations between multiple observations from previous frames to the current frame. Currently, all observations with multiple possible candidate objects are removed. However, it would be interesting to explore a hierarchy of associations where the most likely association is used for the first initialization attempt, and then if it is not successful, the next most likely association is used for the next initialization attempt. This method will ensure that any observation is only used for one object and that the object is initialized with the most likely association. However, it will also increase the initialization time.

3.4 Matching

Data association is the process of creating matches between previous observations with current observations. We approach this similarly to other landmark-based systems, where objects are stored as point landmarks on the map. This approach is sparse, since it stores each landmark as a single point, but is not descriptive of the size or shape of the object. To supplement this, our objects also store a collection of feature descriptors. One unique challenge with open-set systems is that the observed descriptor will not exactly match landmarks in the map, as in the case of a closed-set detector with class labels. We will have to use a distance metric to determine associations, much like how it is done for feature-based matching with ORB feature descriptors. However, unlike with ORB descriptors, the number of landmarks in the map is drastically lower, and therefore our data association method must have high recall. To accomplish this, we outline several types of matching algorithms that can be used depending on the amount of information available. The pure-semantic (PS) matching algorithm is best used when we don't have a strong position prior on the landmarks, and thus can only work off of their semantic similarity. The bipartite nearest-neighbor (BNN) matching uses bipartite matching between observations and landmarks to find the set of associations such that the sum of projective error is minimized. The ambiguity-aware matching (AAM) algorithm allows for non-bipartite matching such that observations can be matched to multiple landmarks at the same time. Only one association will be correct, and that is formalized using max-mixture or sum-mixture noise models in the backend.

3.4.1 Pure Semantic Matching

We define the cost $c_{o,l}$ to be $1 - \max_{f_l \in F_l}(\cos(f_o, f_l))$ and populate the cost matrix as such. F_l is the set of feature descriptors associated with the landmark l, and f_o is the observed features descriptor. There is also a threshold similarity θ_s such that all costs greater than this will be assigned to MAXCOST to denote the lack of an edge. Figure 3.10 shows an example of this algorithm. We use this algorithm to match observations with candidate objects because we have yet to initialize the object position. Alternatively, we can use optical flow or other forms of object tracking algorithm to track across adjacent frames, but we have not explored this method in our work.

The Hungarian algorithm[88] is then used to perform bipartite matching. The overall objective becomes:

$$d^* = \arg\min_d \sum_{i,j} c_{i,j} d_{i,j} \tag{3.9}$$

$$\text{subject to} \qquad c_{i,j} = 1 - \max_{f \in F_j}(\cos(f_o, f)) \tag{3.10}$$

$$\sum_i d_{i,j} <= 1 \tag{3.11}$$

$$\sum_j d_{i,j} <= 1 \tag{3.12}$$

$$\sum_{i,j} d_{i,j} = \min(n, m) \tag{3.13}$$

Where d is the association matrix between observations and landmarks, and n, m are the number of observations and landmarks respectively.

3.4.2 Bipartite Nearest Neighbor

We define a similarity threshold S_{th} and find all observation-landmark pairs above this threshold. We then add each of those pairs into the cost matrix with the projective error as cost. We then filter by a distance threshold d_{th} so that pairs that are too far away are not going

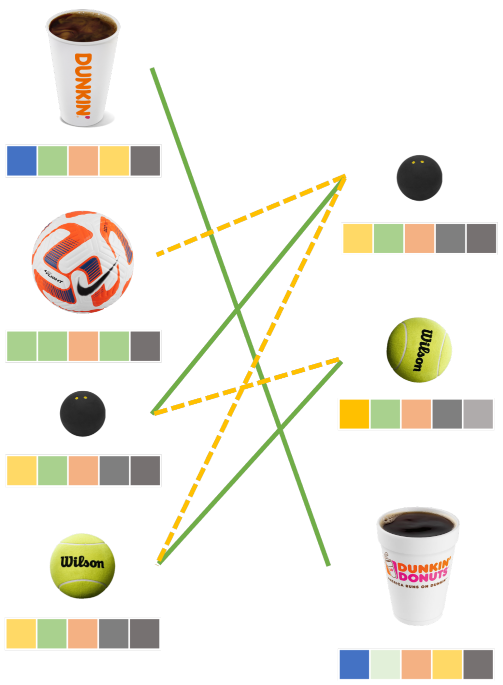


Figure 3.10: An example of the pure semantic matching algorithm. The cost matrix is composed of the cosine similarity between the observation (left) and landmark (right) feature descriptors, which are visualized as colored blocks. The Hungarian algorithm is used to find the optimal matching. The solid green lines in this figure show the matched observations and the dotted lines show observations that are not matched even though they are below the similarity threshold.

to be matched. This is to reduce the chance of an incorrect correspondence based on two similar objects in the scene.

Similar to PSM, the Hungarian algorithm is used to perform bipartite matching. The objective function is

$$d^* = \arg\min_{d} \sum_{i,j} c_{i,j} d_{i,j} \tag{3.14}$$

$$\text{subject to} \qquad c_{i,j} = \|p_i - p_j\|^2 \tag{3.15}$$

$$\sum_{i} d_{i,j} <= 1 \tag{3.16}$$

$$\sum_{j} d_{i,j} <= 1 \tag{3.17}$$

$$\sum_{i,j} d_{i,j} = \min(n, m) \tag{3.18}$$

3.4.3 Ambiguity Aware Matching

Much like BNN, we find pairs of observations and landmarks and filter by both semantic similarity and distance. Instead of using the Hungarian algorithm, we find ambiguous associations of multiple observations to a single landmark or multiple landmarks to a single observation and create a special factor that includes ambiguity. The constraint enforced is that only one of those potential associations can be active at a time, although the active constraint may change over time depending on information after the moment of association.

The objective function is similar to BNN but we are allowed to make multiple associations.

$$d^* = \arg\min_d \sum_{i,j} c_{i,j} d_{i,j} \tag{3.19}$$

$$\text{subject to} \qquad c_{i,j} = \|p_i - p_j\|^2 \tag{3.20}$$

$$\sum_i d_{i,j} <= k \tag{3.21}$$

$$\sum_j d_{i,j} <= k \tag{3.22}$$

$$\sum_{i,j} d_{i,j} = k \min(n, m) \tag{3.23}$$

Where k is the maximum number of associations made per object.

3.5 Optimization

The optimization part of our open-set object-based SLAM system is similar to other object SLAM systems. We have a combination of single and mixture-based factors, optimizing over projective and depth residuals. To reduce the chance that a false data association creates a degenerate solution, we used a Cauchy noise model to reduce the effects of outliers.

3.5.1 Max and Sum Mixture Models

The mixture models are used to model the ambiguity in the data association between landmarks and observations. We create mixture factors in 3.4 when we have observations that can correspond to multiple existing landmarks, or vice versa. The mixture factor ensures that only a single association can be active at once during the optimization process, thus fulfilling the bijective property of data association.

The implementation used in our system comes in the form of a custom factor that takes in a vector of factors and a values variable. During error evaluation, the max-mixture factor evaluates all errors and returns the error with the lowest magnitude, thus choosing the most likely factor to be active. During the Jacobian evaluation, the factor evaluates the Jacobian of the most likely factor and returns it. This is a simple implementation that works well for our purposes. Likewise, with the sum-mixture factor, we return the sum of all errors and the sum of all Jacobians.

Algorithm 1 Max-mixture factor error evaluation

```
Require: $\phi_1 \dots \phi_n, \hat{X}$
Ensure: $\mathbf{e}$ is the minimum error of all $\phi_i$
1: $i \leftarrow 1$
2: $\mathbf{e} \leftarrow \phi_1(\hat{X})$
3: while $i \leq n$ do
4:     $\mathbf{e}_i \leftarrow \phi_i(\hat{X})$
5:     if $\|\mathbf{e}_i\| < \|\mathbf{e}\|$ then
6:         $\mathbf{e} \leftarrow \mathbf{e}_i$
7:     end if
8:     $i \leftarrow i + 1$
9: end while
```

3.5.2 Robust Cost Functions

We use a Cauchy cost function[89] to model the residuals of the projective and depth factors. The Cauchy cost function is a robust cost function that is less sensitive to outliers than the traditional L2-norm cost function. The Cauchy cost function is defined as follows:

$$\rho(x) = \frac{1}{2}k^2 \log(1 + \frac{x^2}{k^2}) \tag{3.24}$$

where x is the residual and k is a positive scale parameter. The scaling parameter is set to 4 such that all measurements above 4 are considered outliers. Since our factor uses the covariance of the measurement to weigh the residual, we do not need to scale the error. This is equivalent to counting all measurements outside of 4σ of any given Gaussian distribution as an outlier. The Cauchy cost function is used in both the projective and depth factors, but not the odometry factors. Figure 3.11 shows the Cauchy cost function compared to the standard L2-norm cost function.

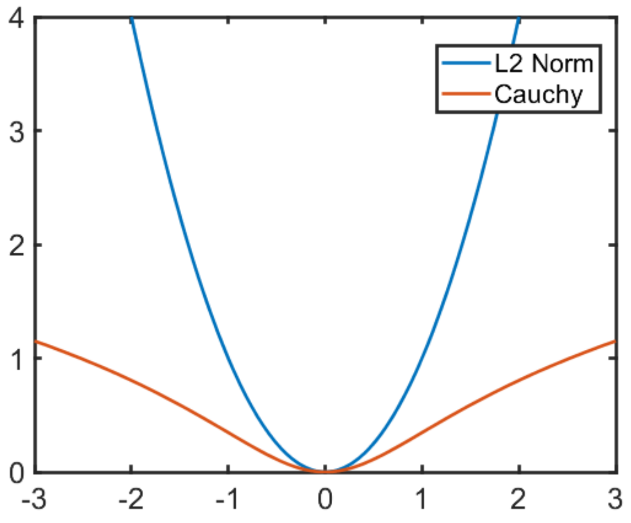


Figure 3.11: Plot of the Cauchy cost function compared to the standard L2-norm cost function. The Cauchy cost function is less sensitive to outliers than the L2-norm cost function and thus prevents incorrect data associations from dominating the error term of our objective function.

3.5.3 Overall Objective Function

The entire optimization problem consists of the following objective function:

$$\min_{\mathbf{R},t,\boldsymbol{\ell}} \quad \sum_{i,j} \|(t_j - \mathbf{R}_{ij} t_i) - \tilde{t}_{ij}\|_{\Sigma}^{2} + \|\mathbf{R}_j^{\mathsf{T}} - \mathbf{R}_i \tilde{\mathbf{R}}_{ij}\|_{\Sigma}^{F} \tag{3.25}$$

$$+ \sum_{i,j} \rho\left(\|\pi(\mathbf{R}_i, t_i, \boldsymbol{\ell}_j) - \tilde{p}_{ij}\|_{\Sigma}^{2}\right) \tag{3.26}$$

$$+ \sum_{i,j} \min_{\boldsymbol{\ell} \in \boldsymbol{\ell}_{\mathcal{H}}} \rho\left(\|\pi(\mathbf{R}_i, t_i, \boldsymbol{\ell}) - \tilde{p}_{ij}\|_{\Sigma}^{2}\right) \tag{3.27}$$

$$+ \sum_{i,j} \rho\left(\|d_{ij} - \tilde{d}_{ij}\|_{\Sigma}^{2}\right) \tag{3.28}$$

$$\text{subject to} \quad \mathbf{R}_i \in \mathrm{SO}(3) \quad \forall i \tag{3.29}$$

$$t_i \in \mathbb{R}^3 \quad \forall i \tag{3.30}$$

$$\boldsymbol{\ell}_j \in \mathbb{R}^3 \quad \forall j \tag{3.31}$$

$$d_{ij} = \|[0, 0, 1] \cdot \mathbf{R}_i^{-1}(\boldsymbol{\ell}_j - t_i)\| \tag{3.32}$$

We can see that there is a cost for odometry using Lie algebra 3.25, a robustified cost for projection of unambiguous landmark associations onto the image plane 3.26, as well as ambiguous landmark associations 3.27, and a cost to the depth measurement between the camera and landmark, if applicable 3.32 3.28.

3.5.4 Incremental Smoothing and Mapping

When the problem is constructed and ready to be solved, we have two optimizer options to choose from. The first is a batched optimizer using the Levenberg-Marquardt algorithm, where the entire factor graph is converted into a nonlinear least-squares problem and all variables are solved. The second is an incremental optimizer using the iSAM2 algorithm, where only a part of the factor graph is solved as new measurements are added. The incremental optimizer drastically improves the runtime by only selectively updating parts of the problem that are affected by new measurements and is, therefore, the preferred method for large-scale problems. However, the method using the incremental optimizer requires a few modifications to be used with max-mixture factors.

In the error evaluation function of the mixture factors, it takes in a series of values which are the current solution, and evaluates multiple factors to determine the composition of the mixture. However, in the incremental optimizer, not all values are included in the problem when it is solved. To evaluate the mixture factors, we need to manually keep track of the relevant values in the factor and update them as necessary. This is done by adding a shared reference to the factor containing the previous best estimate of the values, and using this previous estimate if the values are not in the current problem. This is a simple modification that allows us to use the incremental optimizer with max-mixture factors.

In this chapter we described the system in detail, including the object detection system, data association algorithms, and the optimization backend with ambiguous data association. We also reviewed how strengths and weaknesses of odometry systems. In the next chapter, we will evaluate our system.

Chapter 4

Results and Discussion

For our experiments, we measure the effectiveness of the system based on two criteria: one is the accuracy of the trajectory, and another is its ability to provide a semantically meaningful sparse map that could be used for localization and downstream tasks such as task and motion planning. In terms of accuracy, we compare our system to the ground truth trajectory provided by the dataset and calculate the average pose error. In terms of semantic map quality, we rely on visual inspection and querying of the map to determine whether the map is semantically meaningful. In the future, synthetic datasets with ground truth semantic maps could be used to quantitatively evaluate the semantic map.

4.1 Experiment Setup

4.1.1 Datasets

To evaluate this system's effectiveness in real-world scenarios, we tested it on the TUM-RGBD dataset [90].

TUM-RGBD

The TUM-RGBD dataset is a standard benchmark for visual odometry and SLAM algorithms. It consists of multiple sequences of RGB-D images, each with ground truth poses. The dataset contains unsynchronized RGB images and depth images, from a monocular camera and a Microsoft Kinect respectively. We chose this dataset because it allows us to use an RGB-D odometry source, and some of its sequences contain loop closures. This dataset is also suitable for our purpose because it contains cluttered office environments, where there exist distinct objects that we can recognize and use as landmarks in our system.

The TUM-RGBD dataset contains the following data stream:

- RGB images 640x480 at 30 Hz
- Aligend Depth images 640x480 at 30 Hz
- Ground truth poses at 30 Hz

4.1.2 Computation and Communication

The system uses ROS to handle the communication between different parts, which also allows the distribution of computation across multiple machines. This is especially beneficial for our application due to the specialized computation requirements that deep neural networks possess, which we process on a desktop GPU. The system uses an RTX3090 to run the open set detection node, and an AMD Ryzen Threadripper 3960X to run the rest of the system. This allows us to achieve good performance with a 30 Hz input stream, and the GPU can process the input images at 5 Hz. To increase the frame-to-frame associativity, the input is played at $1/4$ speed so that most frames are processed through the object detector. Note that the system is limited by the object detector and not by the data association and optimization, even when we do not use incremental solvers.

4.2 Quantitative Evaluation

4.2.1 Evaluation Metrics

We use a few common metrics to evaluate the trajectory accuracy of the system. In this section, we will review the formulation of the metrics and show the trajectories obtained by the system.

Absolute Translation Error (ATE)

This error measures the magnitude of the translational component of two poses in a common parent frame. We use the L2-norm which gives us the Euclidean distance in meters. ATE $= \|t_2 - t_1\|$, where t is the translational component of the pose.

Absolute Rotation Error (ARE)

This error measures the magnitude of the rotation required to align two poses. There are many ways to quantify the size of a rotation, we chose to measure it in the degree of rotation in the axis-angle formulation.

Given a rotation matrix R, we can find its axis-angle representation using the following:

$$\theta = \arccos\left(\frac{\text{Trace}(R) - 1}{2}\right) \tag{4.1}$$

$$\omega = \frac{1}{2\sin\theta}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} \tag{4.2}$$

Our metric is then ARE $= \|\theta\|$.

Absolute Pose Error (APE)

This error measures the combination of translation and rotation errors in the form of a Frobenius norm of the matrix.

$$\text{APE} = \|T_i^{w^{-1}} \tilde{T}_i^w - I^4\|_F \tag{4.3}$$

where T_i^w is the estimated pose and $\tilde{T}_i^w$ is the ground truth pose.

Trajectory Alignment

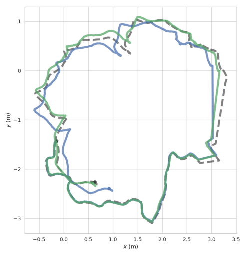
The trajectories are compared using **Evo** [91], which can perform alignment and interpolation of trajectories. Our trajectories are aligned using the Kabsch-Umeyama Algorithm[92][93] and corrected for scale. While we use RGBD odometry which should have a correct scale, it is observed that we often run into issues where the resulting trajectory has a noticeable scale difference from the ground truth. Alignment of the poses is done using the first 30 percent of the trajectory, which corrects for origin alignment and scale. Since the TUM format also includes timestamps for poses, we interpolate the ground truth to find the corresponding pose for our trajectory. Note that the ground truth poses are not time-aligned with the camera frames.

4.2.2 TUM-RGBD

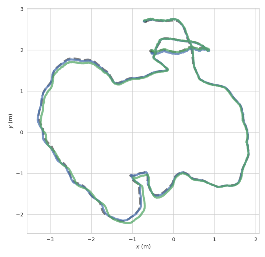
For each dataset, we plot the optimized trajectory, the odometry trajectory, and the ground truth trajectory. The ground truth is provided in dashed gray for comparison purposes. We also plot the error in translation and rotation over time for each dataset for quantitative evaluation.

Overview of All Trajectories

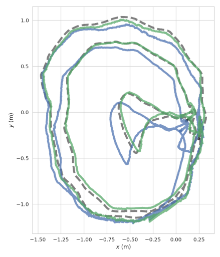
Figure 4.1 shows both the odometry and the optimized trajectory plotted against the ground truth trajectory. The odometry trajectory is in blue, the optimized trajectory is in green, and the ground truth is in gray. In most cases, the optimized trajectory can correct for odometry drift and lower overall translation errors.



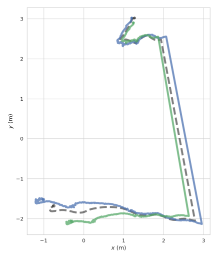
(a) Freiburg2 Desk



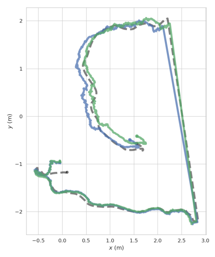
(b) Freiburg3 Long Office



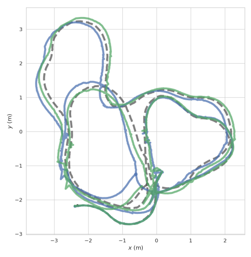
(c) Freiburg2 Dishes



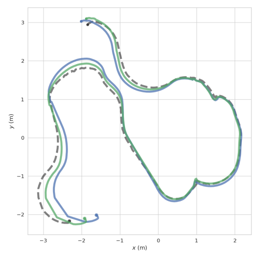
(d) Freiburg2 Large No Loop



(e) Freiburg2 Large With Loop



(f) Freiburg2 Pioneer Slam

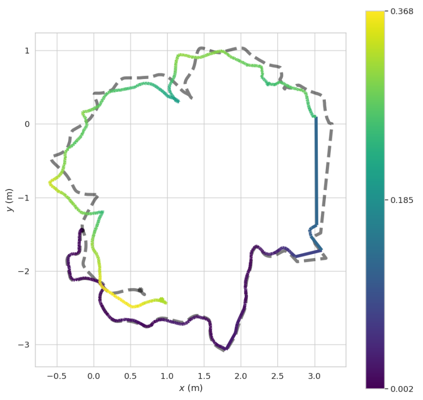


(g) Freiburg2 Pioneer Slam3

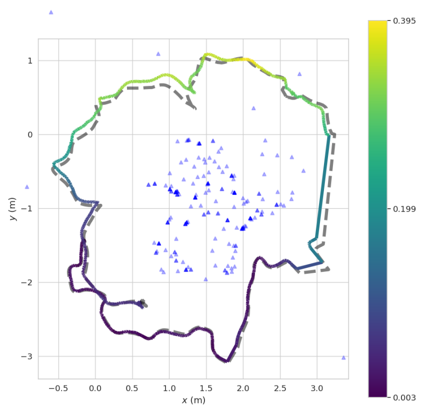
Figure 4.1: Overview of all trajectories.

Freiburg2 Desk

This dataset consists of a trajectory around a desk with various objects on it in a cluttered office environment. The camera points toward the center of the desk and observes the objects from multiple perspectives. The odometry does well in this dataset but it is apparent that odometry drift is accumulated as the camera completes the loop and comes back to the original location. As seen in Figure 4.2, the system is able to correct for the drift at the end of the trajectory and lower the error, and the correction is also propagated towards



(a) Color mapped translation error on odom trajectory.

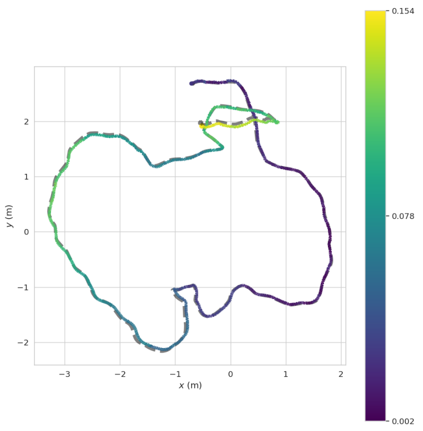


(b) Color mapped translation error on the optimized trajectory. The blue triangles are landmarks detected by the system.

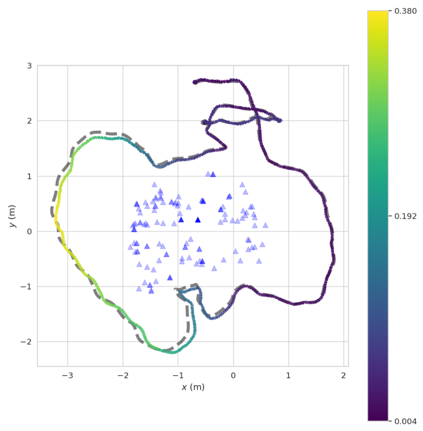
Figure 4.2: Freiburg2 Desk Results.

the middle of the trajectory. We see a dense collection of objects in the center of the plot, with some objects being recognized multiple times as duplicate objects that occupy the same space.

Freiburg3 Long Office Household



(a) Color mapped translation error on odom trajectory.

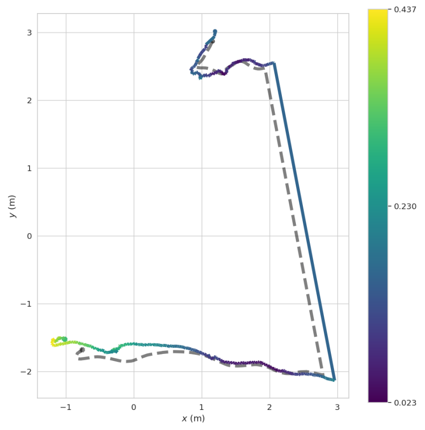


(b) Color mapped translation error on the optimized trajectory. The blue triangles are landmarks detected by the system.

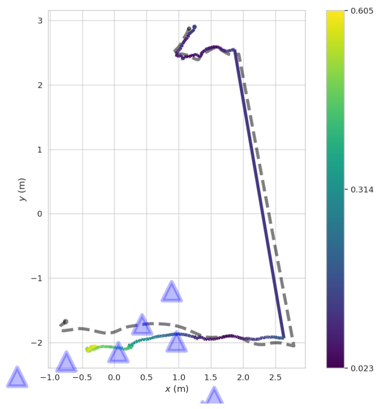
Figure 4.3: Freiburg3 Long Office Results.

Very similar to Freiburg2 Desk, this office scene has objects scattered around the desks and creates many opportunities for the system to recognize and create landmarks. The odometry does well in this dataset as well, due to its distinct environment and objects' close range to the sensor. One thing we noticed in this trial is that the system created incorrect data associations in the middle of the trajectory and resulted in greater errors when compared to the odometry. However, the end of the trajectory is noticeably more aligned with the ground truth and improves on the odometry.

Freiburg2 Large No Loop



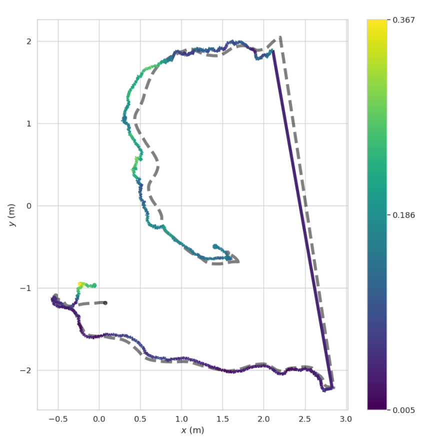
(a) Color mapped translation error on odom trajectory.



(b) Color mapped translation error on the optimized trajectory. The blue triangles are landmarks detected by the system.

Figure 4.4: Freiburg2 Large No Loop Results.

This dataset contains a trajectory around a larger floor space without an opportunity for loop closure and repeated observations of items. As expected, our system performed on par with odometry and did not create any landmarks that helped with reducing drift. The landmarks are scattered far from the trajectory. The ground truth is also truncated so we do not see any landmarks in the right half of the plot.


(a) Color mapped translation error on odom trajectory.

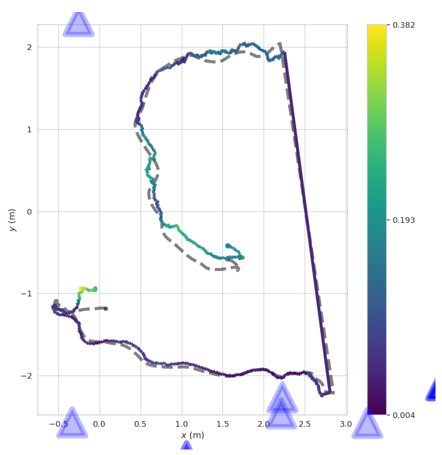

(b) Color mapped translation error on the optimized trajectory. The blue triangles are landmarks detected by the system.

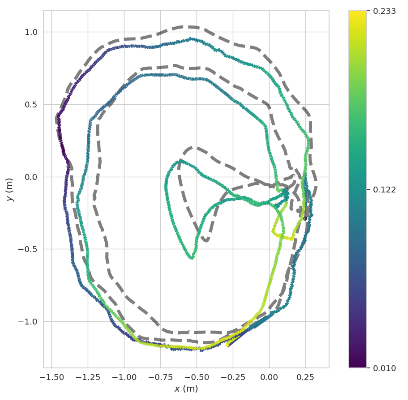
Figure 4.5: Freiburg2 Large With Loop Results.

Freiburg2 Large With Loop

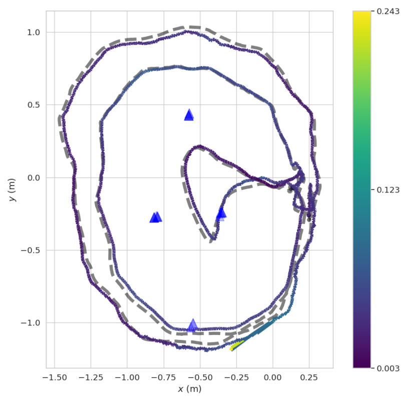
This dataset is similar to the previous one but contains a section where the camera is brought to the start of the trajectory. This allows the system to create loop closures and correct for drift. As seen in Figure 4.5, the system is able to correct for drift and lower the error. Due to the size of the scene, ground truth was provided for only the start and end of the trajectory, so we are unable to determine whether the reduction in drift propagated corrected throughout the rest of the trajectory.

Freiburg2 Dishes

This scene is a small table with four objects on it. The camera traversed around the table, pointing inward towards the dishes. We believe this example showcases the ability of this system to navigate around ambiguous objects and create associations that leverage the max-mixture model. The system is able to significantly reduce drift across the entire trajectory because it maintains associations with the bowls the entire time. The association is not perfect, however, so we can see duplicate items on the map shown as very dark overlapping



(a) Color mapped translation error on odom trajectory.

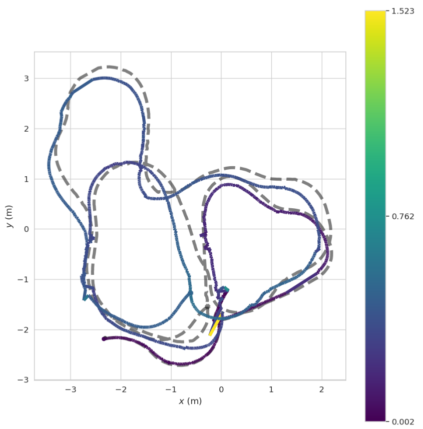


(b) Color mapped translation error on the optimized trajectory. The blue triangles are landmarks detected by the system.

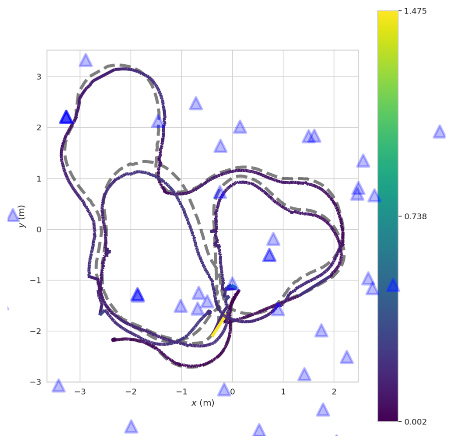
Figure 4.6: Freiburg2 Dishes Results.

triangles. This is a result of the system not recognizing the previously discovered object and creating a new one. In the future it may be helpful to perform duplicate object detection and pruning.

Freiburg2 Pioneer SLAM



(a) Color mapped translation error on odom trajectory.

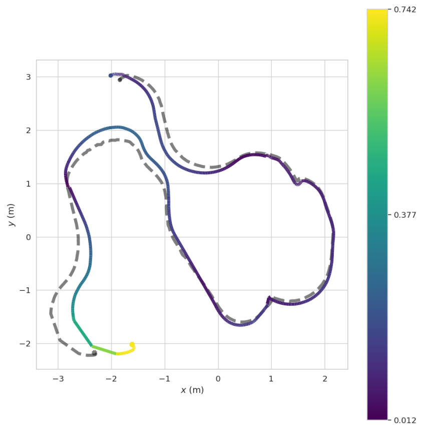


(b) Color mapped translation error on the optimized trajectory. The blue triangles are landmarks detected by the system.

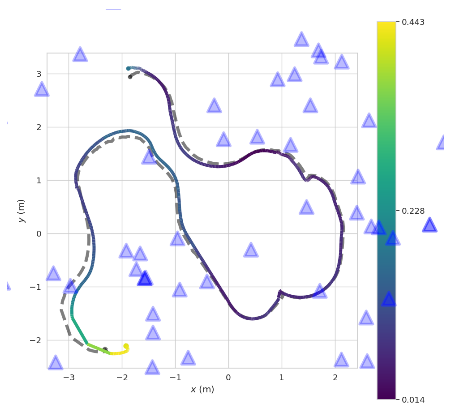
Figure 4.7: Freiburg2 Pioneer SLAM Results.

This dataset consists of a forward-facing camera on a wheeled robot that traverses around a large room. The path of the robot makes it very difficult to initialize objects using multi-view triangulation. We observed that most of the objects were initialized using the depth initialization method. Even so, we were able to correct drift in multiple parts of the trajectory as we created loop closures. In particular, the loop in the upper left and the last part of the trajectory was much closer to ground truth when compared to odometry. The map created by the system shows items scattered throughout the scene, with some duplicate landmarks that show up as dark triangles.

Freiburg2 Pioneer SLAM3



(a) Color mapped translation error on odom trajectory.



(b) Color mapped translation error on the optimized trajectory. The blue triangles are landmarks detected by the system.

Figure 4.8: Freiburg2 Pioneer SLAM3 Results.

This dataset is similar to the previous one but has less number of loops and opportunities for loop closure. This dataset also had instances where we did not receive images for a significant period of time, which caused RTAB-MAP Odometry to fail. Instead, we used the wheel odometry provided by the robot base, which incurs much more drift than the RTAB-MAP Odometry. Nevertheless, the system can correct for a small amount of drift both at the beginning and the end of the trajectory.

4.2.3 Calculated Mean Errors for Trajectory

Table 4.1 shows the mean ATE, ARE, and APE for each of the seven trajectories

Dataset	APE	ATE (m)	ARE (deg)
Freiburg2 Desk	0.228	0.072	8.690
Freiburg3 Long Office	0.168	0.122	4.390
Freiburg2 Dishes	0.061	0.040	1.820
Freiburg2 Large No Loop	0.391	0.294	8.645
Freiburg2 Large With Loop	0.199	0.105	5.863
Freiburg2 Pioneer SLAM	0.154	0.119	3.311
Freiburg2 Pioneer SLAM3	0.198	0.111	6.032

Table 4.1: Mean ATE, ARE, and APE for each dataset.

4.2.4 Quantitative Examples of Loop Closure

One can observe instances of loop closure when the system corrects for drift. We created plots of mean error over time for the optimized trajectory to visualize instances where this happens. The first example is from Freiburg2 Desk, which is shown in Figure 4.9 we can see that the mean error is reduced at the end of the trajectory, which is an instance of loop closure. This is also visible in the map, where the system can create associations between objects that were observed at the start and the end of the trajectory.

Another example can be seen in Figure 4.10, which is from Freiburg2 Pioneer SLAM. We can see that the mean error decreases abruptly multiple times toward the end of the trajectory, which is an instance of loop closure.

In contrast, a trajectory that didn't have a loop closure was Freiburg2 Large No Loop, which is shown in Figure 4.11. We can see that the mean error almost monotonically increases due to unbounded drift.

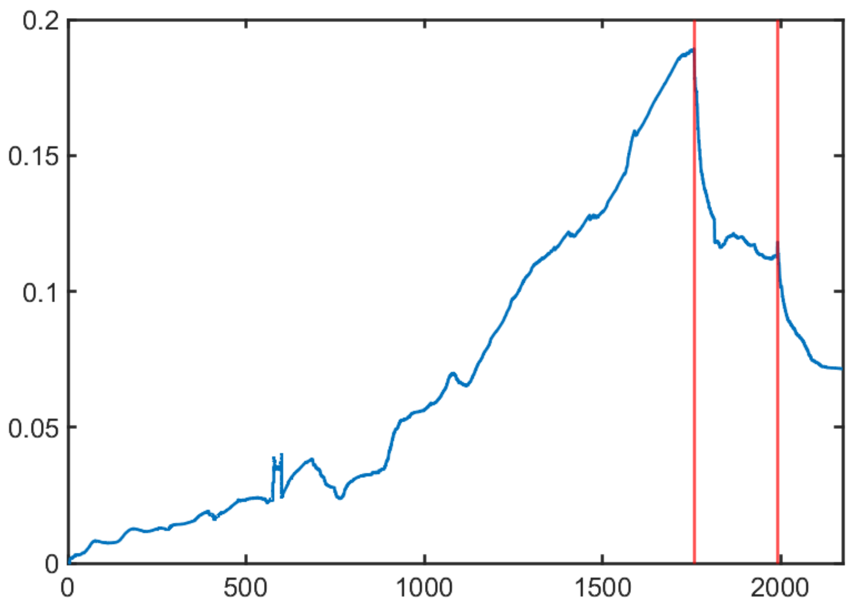


Figure 4.9: Mean translation error over frame number plot for Freiburg2 Desk. The red lines indicate times of loop closure, which occur towards the end of the trajectory when the camera returns to the start of the trajectory and creates data associations with previously observed objects.

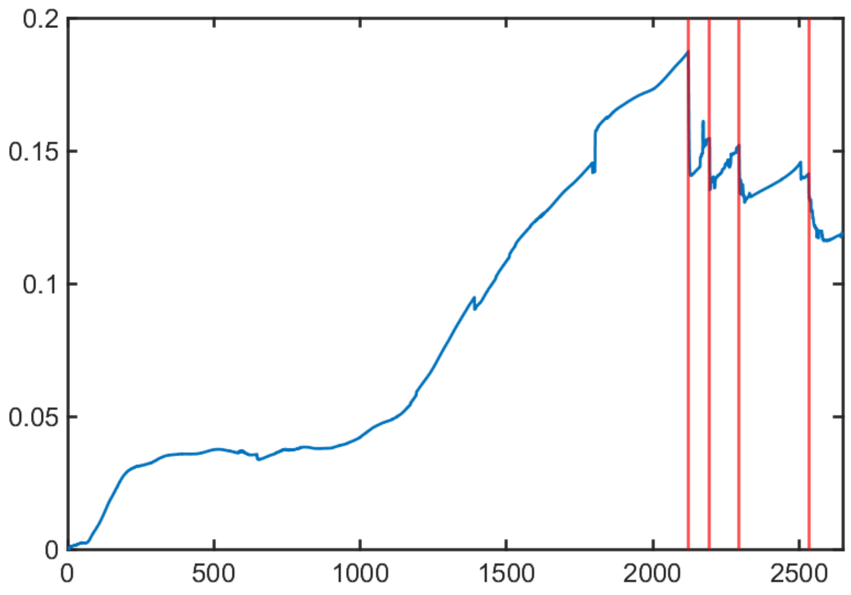


Figure 4.10: Mean translation error over frame number plot for Freiburg2 Pioneer SLAM. The red lines indicate times of loop closure, which occur towards the end of the trajectory when the camera returns to the start of the trajectory and creates data associations with previously observed objects.

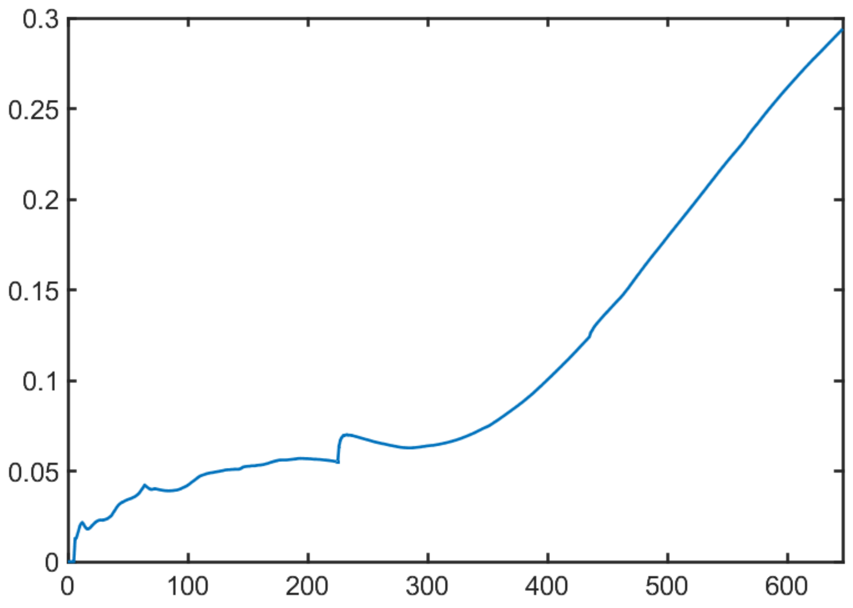


Figure 4.11: Mean translation error over frame number plot for Freiburg2 Large No Loop. The mean error almost monotonically increases due to unbounded drift. This trajectory did not revisit locations, thus we do not have loop closures.

4.2.5 Map Quality

To qualitatively evaluate our system, we presented maps generated by our system for some of the experiments in the figures from the previous section. The objects created by the map are shown in blue triangles, and the trajectory is shown in the background. Most of the objects in the map correspond to an object in the real world, however, some incorrectly initialized objects are also present. We also see instances of double initialization of the same object, which is a result of the system not recognizing a previously discovered object. A higher threshold on the object similarity score in the matching process could cause this behavior, which is an instance of the tradeoff between false positives and false negatives. In our case, we would prefer higher precision detections and are willing to accept a lower recall.

Dataset	Odom Source	Odom	Ours	ORB-SLAM2
Freiburg2 Desk	RTAB-MAP	0.123	**0.072**	**0.012**
Freiburg3 Long Office	RTAB-MAP	**0.062**	0.122	**0.009**
Freiburg2 Dishes	RTAB-MAP	0.130	**0.040**	**0.070**
Freiburg2 Large No Loop	ORB-SLAM*	**0.240**	0.294	**0.125**
Freiburg2 Large With Loop	ORB-SLAM*	0.110	**0.105**	**0.109**
Freiburg2 Pioneer SLAM	RTAB-MAP	0.280	**0.119**	**0.171**
Freiburg2 Pioneer SLAM3	Wheel	0.148	**0.111**	**0.046**

* Indicates modified version of ORB-SLAM2 described in 3.2.1. This is different from the last column which is unmodified ORB-SLAM2.

Table 4.2: Comparison of trajectory ATE between odometry, our system, and ORB-SLAM2. The top two results are in bold.

4.3 Comparison with ORB-SLAM2

The system is able to correct drift in the odometry by creating loop closures with previously traversed portions of the world. Through the maps generated by the system shown in the previous section, we can see that objects are recognized and associated multiple times across the duration of the trials. The system is also able to create associations with objects across a variety of viewpoints, which is a difficult task due to the possible changes in the appearance of the items. By comparing the optimized trajectory with the ground truth, we can see that the average translation and rotation error is reduced by a significant amount and is lower than the error of the odometry, which reinforces the idea that the system can correct for drift.

By comparing the results in Table 4.2, we can see an average reduction of 11.2 percent in the mean absolute translation error when compared to odometry.

4.3.1 Resource Usage

Our system uses much more CPU and GPU computation, partially due to the ViLD detector using a large amount of resources, and also partly due to this system being unpolished software. However, we use much less memory than ORB-SLAM2 because we have fewer landmarks. Table 4.3 shows a comparison between the size of the saved map from ORB-SLAM2 and the map representation of our system calculated based on the number of landmarks and worst-case observation. We can see that our system uses much less memory than ORB-SLAM2, which is a significant advantage for use in lower-resource platforms. The memory usage of our system is calculated with the following formula:

$$\text{Memory Usage} = \#\text{ LMs } \times 30 \text{ Descriptors Per LM } \times 1024 \text{ Bytes per Descriptor} \tag{4.4}$$

Dataset	ORB-SLAM2 (MB)	Ours (MB)
Freiburg2 Desk	48	6.7
Freiburg3 Long Office	52	4.2
Freiburg2 Dishes	194	0.7
Freiburg2 Large No Loop	153	2.2
Freiburg2 Large With Loop	214	2.6
Freiburg2 Pioneer SLAM	217	1.8
Freiburg2 Pioneer SLAM3	123	2.2

Table 4.3: Comparison of map size between ORB-SLAM2 and our system.

We can see that our system uses much less memory than ORB-SLAM2, due to the choice of objects as our representation. For environments with a lower density of objects and more empty space, we are able to see a 99 percent reduction in map size. For environments with a higher density of objects, we are still able to see a 85 percent reduction in size. This comes at a cost of the descriptiveness of the map, as we are only able to represent objects that are detected and properly initialized, but we have shown that we are still able to use those objects for loop closure.

In this chapter, we described the experiments performed to evaluate the system. We showed that the system is able to correct for drift and create loop closures. We also compared our system to ORB-SLAM2 and showed that our system is able to reduce the mean absolute translation error by 11.2 percent when compared to odometry. In the next chapter, we will conclude this thesis and discuss future work.

Chapter 5

Conclusion

5.1 Summary

In this thesis, we explored the problem of using open-set objects in a SLAM system, and how to perform object detection, association, and optimization in a way that works on open-set objects. We presented a method for using a max-mixture model to represent ambiguous data associations and showed how to use this model in a SLAM system. We also presented three different matching algorithms used in the case of candidate, ambiguous, and non-ambiguous data associations, which improves the robustness of the system to multiple instances of the same object, or objects that are similar to each other in feature descriptors. We showed that our system can use multiple sources of odometry and sensor models, and initialize objects with both multi-view triangulation and depth. After testing on several experiments in the TUB RGB-D dataset, we showed that our system is able to successfully create a sparse map of the environment and use it to improve the localization accuracy of the robot. We also showed that our system can represent the world using a smaller number of landmarks than a non-object-based sparse system (ORB-SLAM2), without sacrificing the ability to perform loop closures.

5.2 Future Work

5.2.1 Improving Localization Accuracy

It is clear from Table 4.2 that our system can improve upon the localization estimate of the odometry sources. We can compete with state-of-the-art methods in terms of accuracy for some trials, but we are not able to do so for most. This is due to a variety of factors, but one of the contributing factors is the choice of object representation in our system. By choosing to use a centroid representation, we sacrificed a lot of information about the object, such as size and orientation, which could be used to improve the localization accuracy. In the future, we would like to explore using a more complex object representation, such as a 3D bounding box or a quadric, to improve the localization accuracy of our system. Additionally, the extra bits of information provided by a more complex object representation could be used to improve the robustness of the system to ambiguous data associations.

5.2.2 Dynamically Determining Thresholds

Due to the reasons outlined in 3.2.2, we currently use hand-tuned thresholds for determining valid object detections and various parts of the matching algorithms. If we can use information from the map and previous observations, such as the closeness of observations in feature space, we can potentially dynamically determine the appropriate clustering parameters. This would reduce the amount of hand-tuning required to use our system and allow for the deployment of our system in a wider variety of environments without expert knowledge of its operation.

5.2.3 Undoing Data Association

While our method uses max-mixture to describe ambiguous data associations, it cannot fully undo the data association once it is made. If we create an incorrect data association with a non-mixture-based representation, our system relies on robust cost functions to avoid incorrect solutions. However, the observation will have been permanently added to the object in the map, and we currently do not have a way to determine that an association is incorrect, or a way to remove that association.

5.2.4 Ablation Study

In the future, we would like to perform an ablation study over the different components of our system to determine their effects on the performance of the system. We are especially interested in seeing the effect of different object detectors and different parameters for the matching algorithms, as they directly affect the precision and recall of landmarks.

References

[1] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM): Part I," *Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, Jun. 2006, ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1638022.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM): Part II," *Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, Sep. 2006, ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1678144.

[3] C. D. C. Lerma, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Trans. Robotics*, 2016.

[4] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proceedings of Robotics: Science and Systems*, Jul. 2014.

[5] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 5135–5142.

[6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Intl. Conf. on Computer Vision (ICCV)*, Los Alamitos, CA, USA: IEEE Computer Society, 2011, pp. 2564–2571, ISBN: 978-1-4577-1101-5. DOI: http://doi.ieeecomputersociety.org/10.1109/ICCV.2011.6126544.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[8] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008, ISSN: 1077-3142. DOI: 10.1016/j.cviu.2007.09.014.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Eur. Conf. on Computer Vision (ECCV)*, 2016.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," arXiv, 2015. DOI: 10.48550/ARXIV.1506.02640. [Online]. Available: https://arxiv.org/abs/1506.02640.

[11] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, *Microsoft COCO: Common objects in context*, 2014. DOI: 10.48550/ARXIV.1405.0312. [Online]. Available: https://arxiv.org/abs/1405.0312.

[12] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015. DOI: 10.1109/MRA.2015.2448951.

[13] S. Shao, Z. Li, T. Zhang, C. Peng, G. Yu, X. Zhang, J. Li, and J. Sun, "Objects365: A large-scale, high-quality dataset for object detection," in *Intl. Conf. on Computer Vision (ICCV)*, Oct. 2019.

[14] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factored solution to the simultaneous localization and mapping problem,” in *AAAI Conf. on Artificial Intelligence*, Edmonton, Canada: AAAI, 2002.

[15] D. G. Lowe, “Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks,” *Intl. J. of Robotics Research*, vol. 21, no. 8, Aug. 2002.

[16] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A. Ng, “Simultaneous mapping and localization with sparse extended information filters,” in *Proc. of the Fifth Intl. Workshop on Algorithmic Foundations of Robotics*, J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, Eds., Nice, France, 2002.

[17] J. Folkesson and H. Christensen, “Outdoor exploration and SLAM using a compressed filter,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003, pp. 419–427.

[18] E. Eade, P. Fong, and M. Munich, “Monocular graph SLAM with complexity reduction,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 3017–3024. DOI: 10.1109/IROS.2010.5649205.

[19] J. Folkesson and H. Christensen, “Graphical SLAM - a self-correcting map,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 1, 2004, pp. 383–390.

[20] J. Folkesson and H. Christensen, “Closing the loop with graphical SLAM,” *IEEE Trans. Robotics*, vol. 23, no. 4, pp. 731–741, Aug. 2007, ISSN: 1552-3098. DOI: 10.1109/TRO.2007.900608.

[21] F. Dellaert and G. Contributors, *Borglab/gtsam*, version 4.2a8, May 2022. DOI: 10.5281/zenodo.5794541. [Online]. Available: https://github.com/borglab/gtsam.

[22] M. Kaess, A. Ranganathan, and F. Dellaert, “iSAM: Incremental smoothing and mapping,” *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[23] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, “iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[24] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, Nov. 2007, pp. 225–234.

[25] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A versatile and accurate monocular slam system,” *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[26] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017. DOI: 10.1109/TRO.2017.2705103.

[27] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in *Computer Vision–ECCV 2014*, Springer, 2014, pp. 834–849.

[28] J. Mccormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, “Fusion++: Volumetric Object-Level SLAM,” in *2018 International Conference on 3D Vision (3DV)*, Sep. 2018. DOI: 10.1109/3DV.2018.00015.

[29] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, “SuMa++: Efficient LiDAR-based Semantic SLAM,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Nov. 2019, pp. 4530–4537.

[30] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, “MID-Fusion: Octree-based object-level multi-instance dynamic slam,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[31] M. Runz, M. Buffier, and L. Agapito, “Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects,” in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Oct. 2018, pp. 10–20. DOI: 10.1109/ISMAR.2018.00024.

[32] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, “ElasticFusion: Dense SLAM without a pose graph,” in *Robotics: Science and Systems (RSS)*, Rome, Italy, Jul. 2015.

[33] M. Rünz and L. Agapito, “Co-Fusion: Real-time segmentation, tracking and fusion of multiple objects,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4471–4478.

[34] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, “Kimera: From SLAM to spatial perception with 3D dynamic scene graphs,” *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1510–1546, 2021.

[35] S. Yang and S. Scherer, “Monocular object and plane SLAM in structured environments,” *IEEE Robotics and Automation Letters*, Oct. 2019.

[36] S. Yang and S. Scherer, “CubeSLAM: Monocular 3D object SLAM,” *IEEE Trans. Robotics*, Aug. 2019.

[37] L. Nicholson, M. Milford, and N. Sünderhauf, “QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM,” *IEEE Robotics and Automation Letters*, 2019.

[38] Z. Qian, K. Patath, J. Fu, and J. Xiao, “Semantic SLAM with Autonomous Object-Level Data Association,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2021.

[39] J. Zhang, M. Henein, R. Mahony, and V. Ila, “VDO-SLAM: A Visual Dynamic Object-aware SLAM System,” Dec. 2021.

[40] N. Merrill, Y. Guo, X. Zuo, X. Huang, S. Leutenegger, X. Peng, L. Ren, and G. Huang, "Symmetry and uncertainty-aware object SLAM for 6DoF object pose estimation," in *2022 Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, USA, Jun. 2022.

[41] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Portland, Oregon, Jun. 2013.

[42] M. Hosseinzadeh, K. Li, Y. Latif, and I. Reid, "Real-time monocular object-model aware sparse SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 7123–7129. DOI: 10.1109/ICRA.2019.8793728.

[43] A. Rosinol, J. J. Leonard, and L. Carlone, "NeRF-SLAM: Real-time dense monocular SLAM with neural radiance fields," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Detroit, MI, 2023.

[44] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "NICE-SLAM: Neural implicit scalable encoding for SLAM," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Jun. 2022.

[45] J. Fu, Y. Du, K. Singh, J. B. Tenenbaum, and J. J. Leonard, "NeuSE: Neural SE (3)-Equivariant Embedding for Consistent Spatial Understanding with Objects," *arXiv preprint arXiv:2303.07308*, 2023.

[46] C. Kassab, M. Mattamala, L. Zhang, and M. Fallon, "Language-EXtended Indoor SLAM (LEXIS): A versatile system for real-time visual scene understanding," 2023. arXiv: 2309.15065 [cs.RO].

[47] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," 2022. arXiv: 2201.13360 [cs.RO].

[48] A. Radford, J. W. Kim, C. Hallacy, *et al.*, “Learning Transferable Visual Models From Natural Language Supervision,” in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 8748–8763.

[49] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging Properties in Self-Supervised Vision Transformers,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9630–9640, 2021, ISSN: 15505499. DOI: 10.1109/ICCV48922.2021.00951. arXiv: 2104.14294.

[50] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, “DINOv2: Learning robust visual features without supervision,” 2023. arXiv: 2304.07193 [cs.CV].

[51] K. Jatavallabhula, A. Kuwajerwala, Q. Gu, *et al.*, “ConceptFusion: Open-set multimodal 3d mapping,” *arXiv*, 2023.

[52] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, “Open-vocabulary queryable scene representations for real world planning,” in *arXiv preprint*, 2022.

[53] S. Peng, K. Genova, C. Jiang, A. Tagliasacchi, M. Pollefeys, and T. Funkhouser, “OpenScene: 3D scene understanding with open vocabularies,” in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2023.

[54] K. Yamazaki, T. Hanyu, K. Vo, T. Pham, M. Tran, G. Doretto, A. Nguyen, and N. Le, “Open-Fusion: Real-time Open-Vocabulary 3D mapping and queryable scene representation,” *arXiv preprint*, 2023.

[55] Y. Liu, I. E. Zulfikar, J. Luiten, A. Dave, D. Ramanan, B. Leibe, A. Ošep, and L. Leal-Taixé, “Opening up open-world tracking,” 2022.

[56] Y. Wang, J.-W. Hsieh, P.-Y. Chen, and M.-C. Chang, “SMILEtrack: SiMIlarity LEarning for Multiple Object Tracking,” *ArXiv*, vol. abs/2211.08824, 2022.

[57] B. Wang, T. Li, J. Wu, Y. Jiang, H. Lu, and Y. He, "A simple baseline for open-world tracking via self-training," in *Proceedings of the 31st ACM International Conference on Multimedia*, New York, NY, USA: Association for Computing Machinery, 2023, pp. 2765–2774. DOI: 10.1145/3581783.3611695. [Online]. Available: https://doi.org/10.1145/3581783.3611695.

[58] D. Kim, T.-Y. Lin, A. Angelova, I. S. Kweon, and W. Kuo, "Learning open-world object proposals without learning to classify," *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[59] M. Maaz, H. Rasheed, S. Khan, F. S. Khan, R. M. Anwer, and M.-H. Yang, "Class-agnostic object detection with multi-modal transformer," in *17th European Conference on Computer Vision (ECCV)*, Springer, 2022.

[60] K. J. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, "Towards open world object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2021)*, 2021. arXiv: 2103.02603.

[61] O. Zohar, K.-C. Wang, and S. Yeung, "PROB: Probabilistic objectness for open world object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 11 444–11 453.

[62] H. Huang, A. Geiger, and D. Zhang, "GOOD: Exploring geometric cues for detecting objects in an open world," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=W-nZDQyuy8D.

[63] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary detection via vision and language knowledge distillation," *arXiv preprint arXiv:2104.13921*, 2021.

[64] A. Kirillov, E. Mintun, N. Ravi, *et al.*, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV].

[65] Y. Cheng, L. Li, Y. Xu, X. Li, Z. Yang, W. Wang, and Y. Yang, "Segment and track anything," *arXiv preprint arXiv:2305.06558*, 2023.

[66] F. Li, H. Zhang, P. Sun, X. Zou, S. Liu, J. Yang, C. Li, L. Zhang, and J. Gao, "Semantic-SAM: Segment and recognize anything at any granularity," *arXiv preprint arXiv:2307.04767*, 2023.

[67] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong, "Faster segment anything: Towards lightweight SAM for mobile applications," *arXiv preprint arXiv:2306.14289*, 2023.

[68] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," 2023. arXiv: 2306.12156 `[cs.CV]`.

[69] F. Dellaert, M. Kaess, *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[70] T. Davis, J. Gilbert, S. Larimore, and E. Ng, "A column approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.*, vol. 30, no. 3, pp. 353–376, 2004, ISSN: 0098-3500.

[71] K. Levenberg, "A method for the solution of certain nonlinear problems in least squares," *Quart. Appl. Math*, vol. 2, no. 2, pp. 164–168, 1944.

[72] M. J. D. Powell, "A hybrid method for nonlinear equations," in *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, Ed., Gordon and Breach, 1970.

[73] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *CoRR*, vol. abs/1812.01537, 2018. arXiv: 1812.01537. [Online]. Available: http://arxiv.org/abs/1812.01537.

[74] L. Ortiz, L. Gonçalves, and E. Cabrera, "A generic approach for error estimation of depth data from (stereo and RGB-D) 3D sensors," May 2017. DOI: 10.20944/preprints201705.0170.v1.

[75] *Intel RealSense D400 series product family datasheet*, 2023.

[76] K. J. Doherty, Z. Lu, K. Singh, and J. J. Leonard, "Discrete-continuous smoothing and mapping," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 1–8, 2022. DOI: 10.1109/LRA.2022.3216938.

[77] K. Doherty, D. Baxter, E. Schneeweiss, and J. Leonard, "Probabilistic data association via mixture models for robust semantic SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, 2020.

[78] K. Doherty, D. Fourie, and J. Leonard, "Multimodal semantic SLAM with probabilistic data association," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 2419–2425.

[79] S. Liu, Z. Zeng, T. Ren, *et al.*, "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection," Mar. 2023. [Online]. Available: http://arxiv.org/abs/2303.05499.

[80] F. Li, H. Zhang, H. Xu, S. Liu, L. Zhang, L. Ni, and H. Shum, "Mask DINO: Towards a unified transformer-based framework for object detection and segmentation," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2023.

[81] N. Keetha, A. Mishra, J. Karhade, K. M. Jatavallabhula, S. Scherer, M. Krishna, and S. Garg, "AnyLoc: Towards universal visual place recognition," *arXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2308.00688.

[82] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[83] M. Labbé and F. 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*, vol. 36, Oct. 2018. DOI: 10.1002/rob.21831.

[84] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 4666–4672.

[85] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1280–1286. DOI: 10.1109/IROS.2013.6696514.

[86] G. Guennebaud, B. Jacob, *et al.*, *Eigen v3*, http://eigen.tuxfamily.org, 2010.

[87] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2003, Second Edition.

[88] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics (NRL)*, vol. 52, 1955. [Online]. Available: https://api.semanticscholar.org/CorpusID:9426884.

[89] G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Robust pose-graph loop-closures with expectation-maximization," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 556–563. DOI: 10.1109/IROS.2013.6696406.

[90] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[91] M. Grupp, *Evo: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo, 2017.

[92] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991. DOI: 10.1109/34.88573.

[93] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," Sep. 1976. DOI: 10.1107/s0567739476001873.