2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

SemanticSLAM: Learning based Semantic Map Construction and Robust Camera Localization

Mingyang Li, Yue Ma, and Qinru Qiu
Department of Engineering and Computer Science, Syracuse University
{mli170, yma183, qiqiu}@syr.edu

Abstract— **Current techniques in Visual Simultaneous Localization and Mapping (VSLAM) estimate camera displacement by comparing image features of consecutive scenes. These algorithms depend on scene continuity, hence requires frequent camera inputs. However, processing images frequently can lead to significant memory usage and computation overhead. In this study, we introduce SemanticSLAM, an end-to-end visual-inertial odometry system that utilizes semantic features extracted from an RGB-D sensor. This approach enables the creation of a semantic map of the environment and ensures reliable camera localization. SemanticSLAM is scene-agnostic, which means it doesn't require retraining for different environments. It operates effectively in indoor settings, even with infrequent camera input, without prior knowledge. The strength of SemanticSLAM lies in its ability to gradually refine the semantic map and improve pose estimation. This is achieved by a convolutional long-short-term-memory (ConvLSTM) network, trained to correct errors during map construction. Compared to existing VSLAM algorithms, SemanticSLAM improves pose estimation by 17%. The resulting semantic map provides interpretable information about the environment and can be easily applied to various downstream tasks, such as path planning, obstacle avoidance, and robot navigation. The code will be publicly available at `https://github.com/Leomingyangli/SemanticSLAM`**

I. INTRODUCTION

Visual Simultaneous Localization and Mapping (VSLAM) is a challenging computational problem that has wide application areas in various fields such as robotics and autonomous vehicles. The goal of VSLAM is to construct a map of an unknown environment using visual input while simultaneously estimate the position and orientation of the camera.

Traditional VSLAM techniques, such as ORBSLAM [16], perform pose estimation by comparing matched keypoints of camera inputs with recent keyframes. The algorithm first extracts keypoints, which are image features selected either based on predesigned filters [16], [2] or machine learned models[3], from the camera input and track them in keyframes by matching the descriptors. The pose estimation is then performed by minimizing the reprojection error between the matched 3D map points in world coordinates and 2D keypoints. A considerable large number of matched keypoints on consecutive frames is required to reliably perform pose estimation, therefore, the scene continuity in adjacent frames is important. The camera images must be captured and processed frequently to ensure a high level of similarity among the consecutive frames. However, frequent image processing and keypoint comparison will cause significant computation overhead and to record all keyframes of the environment will result in a high storage complexity.

Rather than memorizing the image features of the visual input, human navigates in an environment by roughly estimating their position based on relative location of surrounding objects. Neuroscience research shows that the place cell, i.e., the set of neurons associated with locations, located in the hippocampus and involves in the function of episodic and semantic memory [7], [9]. In this work, we present SemanticSLAM. The algorithm performs localization and map construction using extracted visual semantic information. The map is a 2D array of neuron clusters, each representing a grid area in the environment. The state of the neuron clusters, which can be written as a vector, represents the semantic information of the corresponding grid area. By comparing the observation and the semantic map, a rough estimation can be made about the camera pose. With the estimated pose and the observed semantic information of the surrounding area, the algorithm will update the map to include the new observations.

The pose estimation described above may have large error at the early phase of a mission when the map has not been constructed. Because the robot has no prior knowledge of the environment, no matching could be found by comparing the observation with the map. In addition, the observation is not always perfect. For example, the most common observation error is due to obstruction. Using imperfect observation to update the map will also introduce errors. In this work, a convolutional long-short-term-memory (ConvLSTM) is trained to correct errors during map update such that the errors will eventually converge instead of being magnified with the iteration. To improve the accuracy of pose estimation at the beginning of the mission, we will leverage the reading from a low cost Inertial Measurement Unit (IMU) to cross check the pose and narrow down the region of the map update. Solely rely on the IMU is not an option for pose estimation as the intrinsic error will accumulate and eventually become unbearable. However, using the IMU input to bootstrap the pose estimation at the beginning of the mission is viable.

Compared to traditional SLAM, the SemanticSLAM's advantage is two folds. Firstly, it does not require high frequency observation and image processing. While different distance, view angle and luminance level may change image features in the observation, after the semantic extraction,

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

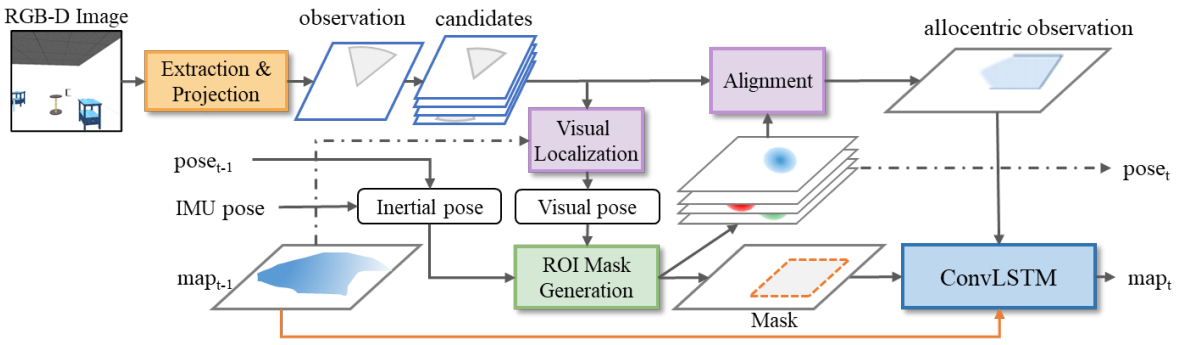

Fig. 1: **System Overview of SemanticSLAM**

such low level variance will be filtered out. Therefore, continuity in the view point is not a necessary condition to find a semantic match. Secondly, compared with maps storing image features and keyframes, a map with semantic information requires much less memory and is more human interpretable. The semantic map can be directly used for mission and navigation planning and can be easily shared among robots as well.

We evaluate the effectiveness of SemanticSLAM, on a dataset obtained from the Gazebo [13] simulation platform using RGB-D cameras and IMU data. Our experimental results demonstrate that SemanticSLAM outperforms other representative methods in terms of accuracy and adaptability to new environments.

II. Related Work

Existing VSLAM techniques, such as ORB-SLAM [16] and LIFT-SLAM [3], employ parallel threads for tracking, mapping, relocalization, and loop closing. Traditionally, feature points are extracted from images using predefined descriptors such as SIFT [14]. The map is a collection of the frames consists of extracted feature points. Camera pose is estimated by minimizing the reprojection error between the query frame and the stored frames that have the largest number of matched feature points. To ensure that an abundance of map points can be found, large amount of keyframes must be stored in the map, and the observation must have a significant amount of continuity. Hence these methods often require significant computational and memory resources. After pose estimation, some recent works [18], [5] also extract semantic features from the camera image and built a semantic map. However, the semantic map is not involved in the process of pose estimation.

There is also a class of research work that performs visual localization without map. One of the techniques is relative camera pose regression (RPR). The algorithm directly predict the displacement of the camera position between a reference image and a query image using neural networks [19], [15]. Such method typically requires highly correlated images, hence needs to be performed frequently. Under low frame rate, the RPR will suffer from accumulated error and eventually lose tracking. Because the RPR does not perform loop closure, additional processing is needed to prevent errors from accumulating. A lightly different technique is absolute pose regression(APR), which approximates the relationship between a query image and reference scene using a neural network. [11], [20]. APR is scene-specific. A new neural network model must be trained for a new application scenario.

To improve the accuracy and robustness of visual localization, the Visual-Inertial Localization method has been introduced. Classical feature-based methods, such as ORB-SLAM3 [4], utilize IMU data to provide a metric pose estimate, which acts as a constraint for visual localization. On the other hand, there are deep learning-based approaches that combine[6] or fuse[1] learned features from both sequential images and IMU data to perform visual inertial odometry. These methods leverage the power of deep learning techniques to effectively integrate visual and inertial information, resulting in improved visual inertial odometry performance. However, these methods are susceptible to various IMU sensor noises that accumulate drift errors, leading to inaccurate pose estimation as the agent moves.

Instead of relying on the matching of keyframes or comparing between the query and reference frames, we project the observed image to build an observation map and try to correlate it with an allocentric "global" map that has been constructed from past observations for pose estimation. To the best of our knowledge, the only work that adopts a similar approach is the MapNet[8]. However, MapNet stores image features instead of semantic features of the environment in the global map. Hence the map cannot be directly used to guide the mission planning. Furthermore, varying image features for the same object may be received when observed from different angles, which makes it difficult to find a match between the current observation and past observations stored in the map.

III. Method

In this section, we present the SemanticSLAM. The structure of the framework is illustrated in Figure 1. It receives the RGB-D observation of the environment and provides an estimated pose of the camera and a semantic map.

We consider the camera localization during navigation. The environment is considered a $H \times W$ grid map, each grid is a possible camera location. The potential camera orientations are discretized into R levels. The robot observes the environment and estimates its location at discrete time t, the time intervals between adjacent observations and estimations

does not have to be equal. For each step t, the output of SemanticSLAM is an array of probability values, denoted as $\mathbf{p_t} \in [0,1]^{R\times H\times W}$, and $\sum_{r,x,y}^{R,H,W} p_t^{r,x,y} = 1,\ \forall t$. The (r,x,y)th entry of the array gives the probability that the camera is at location (x,y) and its orientation falls into the rth level. The index of the entry with the largest value is the estimated pose $\bar{p}_t = argmax_{x,y,r} p_t^{r,x,y}$. The framework also constructs and maintains a neural-symbolic map $m_t \in [0,1]^{L\times H\times W}$ of the environment, where L is the dimension of the semantic feature vector.

The input of the system includes the information (I_t, d_t) from RGB-D camera reading, where $I_t \in \mathbb{R}^{s_1\times s_2\times 3}$ is the set of pixels in RGB image and $d_t \in \mathbb{R}^{s_1\times s_2\times 3}$ gives the 3D point cloud of each pixel in the RGB image. The IMU sensor reading is represented as a triplet, $u_t = [\Delta x_t, \Delta y_t, \Delta\theta_t]$, which describes IMU measured position and rotation changes from last observation.

A. Semantic Feature Extraction and Projection

In this work, we use the class labels of the foreground objects as the semantic feature of the environment. These semantic features are detected from the RGB image, then projected onto a 2D observation map based on the depth information.

Object Extraction. We detect the foreground objects from the received RGB image I_t using a pre-trained Yolo[17] model, which gives a bounding box for each detected object. Next, the object is separated from its background using a semantic segmentation model SAM [12]. After object extraction, we received a group of N foreground objects, $\{(A_i, C_i), 0 \le i \le N-1\}$, which A_i is the set of pixels in the RGB image, and C is its class label, which will be used as the semantic feature in our map.

Feature Projection. Using the one-to-one pixel correspondence between the depth image (d_t) and the RGB image (I_t), the semantic features of the foreground object will be projected onto an egocentric(i.e., using camera coordinate system) 2D observation map $o_t \in \mathbb{R}^{L\times h\times h}$. The observation map represents a top-down view of an $h\times h$ region centered at the camera location with orientation 0. Associated to each map location is an L dimensional vector, storing the semantic features at that location. For each pixel $a \in A_i$ of the ith foreground object, we first identify its corresponding (x,y) location is using the depth image,then increment the value $o(C_i,x,y)$ by $1/|A_i|$, where C_i is the class label of the object and $|A_i|$ is the cardinally of the pixel set A_i. At the end of the feature projection process, an entry $o(l,x,y)$ can be written as the following:

$$o(l,x,y) = \sum_{i,C_i=l} \frac{|a, a\in A_i \& d_a(x,y) = (x,y)|}{|A_i|}, \tag{1}$$

where $d_a(x,y)$ gives the (x,y) position of the pixel a in the depth image.

As we can see, the $o(l,x,y)$ describes how different classes of objects are distributed across the 2D area. The feature value lower than a threshold β (e.g., 0.02) is considered as noise and will be filtered. This projection is by no means perfect. There are many potential sources of errors, the object detection and classification may be wrong, the segmentation may not be perfect, and the feature projection may contain errors. Even if all of these steps are correct, due to obstructions, objects may not be observed or only partially observed. Hence the observation map contains a significant amount of errors. We leverage a ConvLSTM network to correct the errors during map construction.

B. Visual Pose Estimation

The semantic observation map $o_t \in \mathbb{R}^{L\times h\times h}$ is egocentric. It then undergoes a set of rotations at multiple viewing angles $2\pi r/R$ for $r \in \{1,...,R\}$ using Spatial Transformation[10], which generates a set of observations $\acute{o}_t \in \mathbb{R}^{R\times L\times h\times h}$, where R is the number of orientation levels. Each candidate represents a hypothesis of the observation map relative to allocentric(world coordinate system) view angle r. The correlation between each observation candidate and the global map m_{t-1} constructed from previous time step is calculated by applying a 2D convolution on m_{t-1} with kernel $\acute{o}_t$ and stride 1:

$$v_t = \sigma(m_{t-1} * \acute{o}_t) \tag{2}$$

where $\sigma()$ represents the softmax function. We pad the global map m_{t-1} such that the output v_t has the dimension $R\times H\times W$. v_t is a visual pose probability field. Its (r,h,w)th entry indicates the degree of correlation between the map and the observation if the camera is located at position (h,w) with orientation r relative to the world coordinate system, which also gives the probability distribution of the camera's location and pose. The entry with the largest value gives the visual pose estimation: $\bar{v}_t = argmax_{x,y,r} v_t^{r,x,y}$.

C. Cross-check with Inertial Pose Estimation

At the beginning of the mission, before the map is constructed, the observation o_t and the global map m_{t-1} correlates poorly and the aforementioned visual pose estimation will have low performance. We propose to bootstrap the pose estimation using IMU sensor data to mitigate the uncertainty. An alternative inertial pose estimation, $\bar{u}_t \in \mathbb{R}^3$, is obtained. Given the pose estimation in previous step $\bar{p}_{t-1} = (r_p, x_p, y_p)$, the location and orientation changes detected by the IMU sensor over the last observation interval $u_t = [\Delta x_t, \Delta y_t, \Delta\theta_t]$, the inertial pose estimation $\bar{u}_t = (r_i, x_i, y_i)$ can be calculated as the following:

$$\bar{u}_t = T(\bar{p}_{t-1},\ u_t) \tag{3}$$

where $\bar{p}_{t-1} \in \mathbb{R}^3$ is the allocentric pose estimated in last time step and $u_t \in \mathbb{R}^3$ is the egocentric pose displacement sensed by IMU. The transformation function $T(\cdot)$combines these two to calculate the resulting allocentric pose.

It should be noted that the reading of IMU consists of Gaussian noise and fixed biased noise, which results in drift error in the inertial pose estimation. The error in the IMU reading is relatively small, however, after several steps of accumulation, it will become quite significant. On the other hand, when the visual pose estimation is wrong, the error is usually drastically large because we are guessing randomly without evidence of matched features. However,

when matched features are found, the visual pose estimation tends to provide very accuracy results.

Based on the above observation, the final pose estimation $\bar{p}_t$ will be chosen from either visual estimation $\bar{v}_t$ or innertial estimation $\bar{u}_t$ based on the following threshold function:

$$\bar{p}_t = \begin{cases} \bar{v}_t & \text{if } \|\bar{v}_{t,k} - \bar{u}_{t,k}\|^{k=2,3} < \gamma_1 \\ & \text{and } |\bar{v}_{t,k} - \bar{u}_{t,k}|^{k=1} < \gamma_2 \\ \bar{u}_t & \text{otherwise} \end{cases} \tag{4}$$

where γ_1, γ_2 are the maximum noise that the IMU may have during one sample interval. If the difference between the visual and inertial estimations is greater than the maximum possible noise level, we assume that the visual estimation is wrong and accept the inertial estimation. Otherwise, we accept the visual estimation result. The visual-inertial cross check imposes a bound to the pose estimation error, particularly at the onset of the training phase. Hence it accelerates the convergence of the training. It is also easy to implement as the visual and inertial pose estimation are two parallel and independent threads.

D. Map Update

Observation Projection. To update the global map using the observed information, first we need to project the egocentric observations to the allocentric global map at the estimated pose. In visual pose estimation, the pose has a probability distribution p_t, which is a 3D tensor with dimensions $R \times H \times W$. However, the output of the inertial pose estimation $\bar{u}_t$ is a vector (r, x, y) of size 3. To be consistent, we encode $\bar{u}_t$ as a one-hot tensor $E(u_t)$ with dimensions $R \times W \times H$. Combining the visual and inertial estimation, the probabilistic distribution of the pose estimation is chosen as the following:

$$p_t = \begin{cases} v_t & \text{if } \bar{p}_t = \bar{v}_t \\ E(u_t) & \text{otherwise} \end{cases} \tag{5}$$

Subsequently, projecting the egocentric observation onto the map coordinate is achieved using transposed convolution over p_t with kernel $\acute{o}_t$ and stride 1:

$$\tilde{o}_t = p_t * \acute{o}_t \tag{6}$$

The output $\tilde{o}_t \in \mathbb{R}^{R \times H \times W}$ represents the allocentric map that contains observation information

ROI. We create an $h \times h$ Region of Interest (ROI) around the estimated pose $\bar{p}_t$. The global map within the ROI undergoes updates while the areas outside the ROI remain unchanged. This strategy aims to mitigate errors, especially when visual pose estimation v_t contains high levels of uncertainty. To separate the ROI from the rest of the map, an ROI mask, M_t is created as the following:

$$M_t^{x,y} = \begin{cases} 1 & \text{if } x \in [\bar{p}_t^x - \frac{h}{2}, \bar{p}_t^x + \frac{h}{2}],\ y \in [\bar{p}_t^y - \frac{h}{2}, \bar{p}_t^y + \frac{h}{2}] \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Update. Given the projected observation $\tilde{o}_t$ and the ROI mask, we update the global map m_t using a convolutional LSTM model [21]. The model learns how to "remember" or "forget" the information in the current map, and whether the information in the incoming observation can be trusted and stored in the map. The update of the global map can be expressed in the following way

$$m_t = \sigma((1 - M_t) \odot m_{t-1} + M_t \odot ConvLSTM(m_{t-1}, \tilde{o}_t)) \tag{8}$$

where $\odot$ denotes the Hadamard Product. The entry of the map is a vector of size L, which represents probabilities of L class labels of the foreground object located at this position. We use the ROI mask M_t to select the region that needs to be updated, and retain the original value of entries outside the ROI.

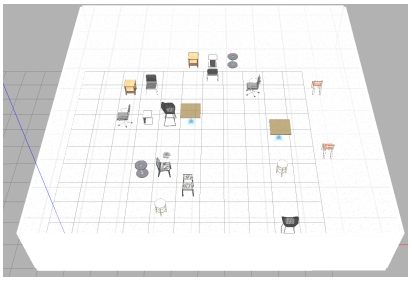

Fig. 2: **IndoorScenes Dataset**

E. Loss function

In the proposed SemanticSLAM, the accuracy of the constructed semantic map plays a critical role in pose estimation. An accurate global map ensures correct pose estimation. Therefore, the ConvLSTM model is trained to improve the quality of the constructed map. We design a loss function that quantifies the accumulated disparity between the constructed map m_t and the ground truth map $\overline{m}_t$ over T time steps as the following:

$$\mathscr{L} = \frac{1}{T} \sum_{t}^{T} \sum_{x,y}^{H,W} \mathscr{D}_{KL}(\overline{m}_{t,x,y} || m_{t,x,y}) \tag{9}$$

where $\mathscr{D}_{KL}(\overline{m}_{t,x,y} || m_{t,x,y})$ represents the Kullback–Leibler divergence(KL) between the ground truth label and the estimated label of the map location (x, y) at time t.

IV. Experiments

A. Experiment Setup

Our evaluation dataset, "IndoorScenes", is a simulated indoor localization dataset. An example of the indoor scene is shown in Figure2. The environment is the indoor environment, featuring various objects randomly placed within it, generated by Gazebo[13], an open-source multi-robot simulator. To simulate low-frequency sensor input, a ground robot simulator, TurtleBot3[22], captures RGB and Depth images with 640x480 resolution and a horizontal filed of view(FOV) of 90 degrees as it traverses the environment. The images are sampled at a rate slower than 1Hz and the IMU sensor contained Gaussian noise and biased noise. We ran the simulator to generate 30 different scenes. Each scene contains 3 different trajectories.

The map size is 33×33, which means the grid size is 300 millimeters. The dimension of the observation map is 11×11 and the orientation of the camera is discretized into $R = 360$ levels.

TABLE I: **Performance comparison.** (Average Position Error(APE), Direction Error(DE))

	SM	DC	Allo	Intra-Scene	Cross-Scene
DeepVO[15]				2.29, 87.31	3.97, 103.7
MapNet[8]			✓	2.40, 106.1	3.90, 92.42
ORBSLAM2[16]		✓	✓	N/A	N/A
our(visual)	✓	✓	✓	2.14, 49.37	3.24, 67.73
our(visual-inertial)	✓	✓	✓	**1.61, 34.64**	**2.39, 57.11**

SM: semantic map, DC: drift correction, Allo: allocentric pose estimation

TABLE II: **Path From base model to SemanticSLAM**

	Mapnet	+Semantic	+ConvLSTM	SemanticSLAM
result	3.90, 92.42	3.80, 80.50	3.24, 67.73	2.39, 57.11

We conducted our experiment under two different settings, "Intra-Scene" and "Cross-Scene". Under the "Intra-Scene" setting, we train the model with two of the three trajectories from all 30 scenes, while test the model using the remaining trajectory from each scene. Under the "Cross-Scene" setting, the dataset was divided into two distinct sets with different scenes to ensure that the testing trajectory and training trajectory are collected from different scenes. The aim of "Cross-Scene" testing is to analyze whether our model can generalize to a new training environments.

Two evaluation metrics are used to measure the quality of the model. The average position error (APE) is determined by calculating the average of the Euclidean distances between the predicted and actual positions over time. The average direction error (ADE) is computed by calculating the mean orientation difference.

For comparison, we implemented and trained several existing models that utilize different localization techniques.

- DeepVO: This is a typical deep learning-based RPR model that estimates relative camera pose through regression. The model takes in a pair of RGB images as input and produces predictions for the relative changes in translational motion $(\Delta x, \Delta y, \Delta\theta)$.
- ORB-SLAM2: This is a keypoint-based VSLAM model. It leverages manually crafted feature descriptors to represent the images, which are then stored in a map.
- MapNet: This model is very similar to ours except that it extracts image features from RGB images and projects them to the observation map. Its global map also stores image features instead of semantic features.

B. Results and Analysis

1) Performance Comparison: Table I compares the average position error and direction error of these SLAM models under both Intra-Scene and Cross-Scene settings. Each test sequence is a trajectory with 30 steps. At each step the robot observes the environment and performs pose estimation. For each test, the robot begins with no prior information of the environment. We can see that the SemanticSLAM with both visual and IMU input has the best performance and the second best model is the SemanticSLAM with only visual sensors. Although DeepVO also performs well in Intra-Scene setting, the model is overly customized to the scenes in the training set, therefore does not have good performance

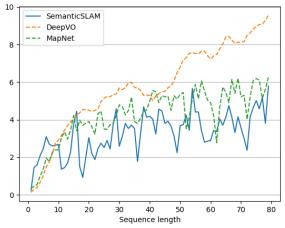

Fig. 3: **Localization performance over time.** Our model (green solid) vs. reference models (dashed lines)

in generalization when applied to the Cross-Scene setting. In contrast, our algorithm learns how to construct the map instead of the map itself, hence it can adapt to environment with different scenes. Finally, we also applied ORB-SLAM2 to our dataset, however, the algorithm could not generate any pose estimation because it could not find sufficient number of matched key points between frames.

Figure 3 illustrates the localization error over time for various methods. It shows that with SemanticSLAM(visual) the error accumulation is significantly slowed down.

2) Ablation Study: The SemanticSLAM is inspired by MapNet. Compared to MapNet, the following enhancements were adopted, (1) semantic features instead of image features were extracted from the observation and stored in the global map, a new feature projection algorithm was developed; (2) the map update is achieved using a ConvLSTM instead of a normal LSTM; (3) we bootstrap the pose estimation by cross-checking the visual and inertial information. In Table II, we show how much improvement each of these enhancement techniques could bring. As we can see, by using semantic features and adopting new feature projection algorithms, we can reduce the APE by 2% and DE by 14%. By adopting the ConvLSTM model, we can further reduce the APE and DE by 17% and 19% respectively. Finally by utilizing the inertial information, the APE and DE can be further reduced by 35% and 18% respectively.

3) Map Construction: Next we will demonstrate the error correction capability of SemanticSLAM during map construction. The baseline map construction algorithm is a heuristic that update each grid of the global map in a leaky-integrate manner:

$$m_t(x,y) = (1-\alpha) \times m_{t-1}(x,y) + \alpha \times \tilde{o}_t(x,y) \quad \text{if } \tilde{o}_t(x,y) > 0 \tag{10}$$

where α varies from 0.1, 0.3 to 0.7 and 1. It controls the leakage of the old information in the map and map update speed. Table III compares the MSE error of the maps constructed using our method and using the heuristic method. To have a fair comparison, we assume that we have perfect knowledge of the camera pose, and the only error is from the observation. The errors in the observation may come from 3 sources. The first type of error is the feature extraction and projection error. The object detection, classification and segmentation may not be perfect and we may project the feature to a wrong position in the observation map o_t. The second type of error is the obstruction error. Even if the feature extraction and projection are perfect, we still may

TABLE III: **Map construction loss of our model compared to heuristic methods.** MSE error and standard deviation(x100)

Condition	Real camera	Obstructed	Ideal
our	**2.51±0.21**	**2.42±0.21**	**2.05±0.29**
heuristic(0.1)	2.90±0.06	3.21±0.04	3.17±0.05
heuristic(0.3)	2.57±0.12	3.18±0.10	3.26±0.15
heuristic(0.7)	2.57±0.28	3.65±0.26	4.77±0.41
heuristic(1.0)	2.94±0.45	4.35±0.41	6.65±0.59

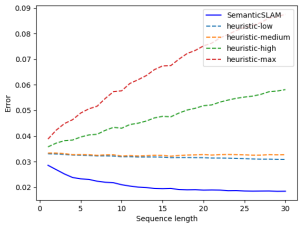

Fig. 4: **Map construction loss over time**

miss some objects in the scene due to obstruction by other objects. The third type of error is rotation error. Even if there is no objection, after rotating the o_t to generate the allocentric observation candidate $\acute{o}_t$, we may introduce interpolation error. To feature these different errors, we created three different type of observation inputs. The "Real Camera" input contains all 3 types of errors, the "Obstructed" input contains the rotation error and obstruction error, and the "Ideal" input contains only the rotation error. As shown in Table III, the map constructed by our algorithm has much lower MSE error compared to the heuristic algorithms.

Figure 4 shows how the map construction error changes as the mission goes on. Using our map construction, the map error reduces with the increase of time steps. Such trend is not observed with the heuristic algorithms.

Figure5 illustrate the map construction process. We pick one of the L channels of the global map m_t and egocentric observation map o_t and show them in the figure. The updated map and the ground truth map is also given. The brightness of the pixel indicate the confidence that this grid cell is occupied.

V. Conclusions

This work presents a pioneering semantic learning-based SLAM system that significantly enhances localization accuracy, particularly for a system with infrequent observation environment. The constructed map offers a comprehensive semantic-level representation of the environment that could be utilized for navigation planning or shared among robots. It also proved interpretable information for human users. Our research demonstrates the practicality and potential of incorporating semantic features into SLAM.

VI. Acknowledgements

This research was partially supported by NSF IUCRC ASIC center (CNS-1822165) and NSF award CNS-2148253.

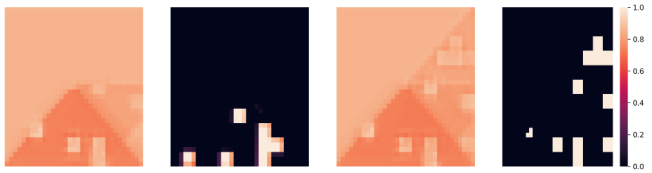

Fig. 5: **Visualization of the Map Update Process**. From left to right: Original global map m_{t-1}, Semantic observation o_t, Updated global map m_t, and Ground-truth.

References

[1] Yasin Almalioglu et al. Selfvio: Self-supervised deep monocular visual-inertial odometry and depth estimation. *CoRR*, abs/1911.09968, 2019.

[2] Berta Bescós et al. Dynslam: Tracking, mapping and inpainting in dynamic scenes. *CoRR*, abs/1806.05620, 2018.

[3] Hudson Martins Silva Bruno and Esther Luna Colombini. LIFT-SLAM: a deep-learning feature-based monocular visual SLAM method. *CoRR*, abs/2104.00099, 2021.

[4] Carlos Campos et al. ORB-SLAM3: an accurate open-source library for visual, visual-inertial and multi-map SLAM. *CoRR*, abs/2007.11898, 2020.

[5] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural SLAM. *CoRR*, abs/2004.05155, 2020.

[6] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. *CoRR*, abs/1701.08376, 2017.

[7] Howard Eichenbaum, Paul Dudchenko, Emma Wood, Matthew Shapiro, and Heikki Tanila. The hippocampus, memory, and place cells: is it spatial memory or a memory space? *Neuron*, 23(2):209–226, 1999.

[8] Joao F. Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8476–8484, 2018.

[9] Nora A Herweg and Michael J Kahana. Spatial representations in the human brain. *Frontiers in human neuroscience*, 12:297, 2018.

[10] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015.

[11] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. *IEEE international conference on computer vision*, pages 2938–2946, 2015.

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[13] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[14] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[15] Vikram Mohanty, Shubh Agrawal, Shaswat Datta, Arna Ghosh, Vishnu Dutt Sharma, and Debashish Chakravarty. Deepvo: A deep learning approach for monocular visual odometry. *arXiv preprint arXiv:1611.06069*, 2016.

[16] Raul Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*, abs/1610.06475, 2016.

[17] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[18] Martin Rünz and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. *CoRR*, abs/1804.09194, 2018.

[19] Paul-Edouard Sarlin et al. Superglue: Learning feature matching with graph neural networks. *CoRR*, abs/1911.11763, 2019.

[20] Greg Schohn and David A. Cohn. Less is more: Active learning with support vector machines. In *ICML*, 2000.

[21] Xingjian Shi et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214, 2015.

[22] Tully Foote, Melonee Wise. Turtlebot, 2011.