TOWARDS ROBUST SLAM WITH NEURAL IMPLICIT REPRESENTATION

Haoming Li

A THESIS

in

Department of Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Master of Science in Engineering

2024

Supervisor of Thesis

Nadia Figueroa

Shalini and Rajeev Misra Presidential Assistant Professor, Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania

Co-Supervisor of Thesis

Pratik Chaudhari

Assistant Professor, Department of Electrical and Systems Engineering, University of Pennsylvania

Program Director

Tania Khanna

Electrical Engineering Master's Program Director, Senior Lecturer Department of Electrical and Systems Engineering, University of Pennsylvania

ABSTRACT

TOWARDS ROBUST SLAM WITH NEURAL IMPLICIT REPRESENTATION

Haoming Li

Nadia Figueroa

Pratik Chaudhari

In this thesis, I investigate the advantages of integrating Lipschitz continuity into neural Simultaneous Localization and Mapping (SLAM) to enhance its robustness against challenges such as catastrophic forgetting and noise sensitivity, which are prevalent in traditional SLAM methods. I have implemented Lipschitz regularization within neural RGBD SLAM systems to bolster their generalization capabilities, thereby improving performance in reconstructing unobserved regions. Additionally, I propose a semantic-informed neural SLAM trying to further enhance the generalization capabilities of these systems and provide more precise and detailed scene reconstructions and robust camera tracking. While the method does not currently surpass state-of-the-art approaches, potential improvements and future enhancements have been identified. Significant contributions of this work include the development of novel weight normalization and regularization techniques to increase network robustness, as well as the exploration of the benefits of semantic information to refine both mapping and tracking processes. The Lipschitz regularized and semantic-informed neural SLAM systems have been evaluated on the Replica and Neural RGBD datasets, demonstrating its potential to impact the fields of robotic navigation and autonomous vehicle guidance.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1. Motivation

Dense Visual Simultaneous Localization and Mapping (SLAM) is a fundamental problem in the field of computer vision and plays a crucial role in various applications such as autonomous driving [35], indoor robotics [34], virtual reality [20], augmented reality [38], robot navigation [17], collision [25], planning tasks [30], and detailed occlusion reasoning [9], which is vital for scene understanding and perception. Its primary goal is to construct a three dimensional dense map of an unknown environment while simultaneously approximating the camera pose in real-time. It is divided into initialization, tracking, and mapping. During initialization, the system establishes an initial state, while in the tracking stage, it estimates poses for mapping stage, while in the tracking stage, it estimates poses for the mapping stage to update the reconstructed scene continuously. Traditional SLAM methods [15], [11], [29] use either feature matching, nonlinear energy minimization, or a combination of both to solve the camera poses. These poses are then coupled with their corresponding input point clouds to update a global map represented by geometric primitives such as cost volumes, surfels, or voxels. Camera tracking is done with sparse point clouds or depth maps, e.g. via frame-to-model tracking and incorporated loop closures. The most common scene representations for dense mapping are voxel grids [6], voxel hashing [16], octrees [26], or point/surfel clouds [20]. Traditional SLAM solutions exist that can robustly track the position of the camera while fusing depth and/or color measurements into a single high-fidelity map. However, they rely on hand-crafted loss terms and do not exploit data-driven priors. Traditional approaches rely on multi-view geometry and mostly focus on localization accuracy. They are incapable of rendering novel views as they cannot hallucinate the unseen parts of the scene. Storing and distributing the maps can be also challenging due to the requirement of large video memory.

Different from the explicit representation methods, a neural SLAM focuses on implicitly representing the scene, which is compact and can be extended to unobserved regions. iMAP [25] is a pioneering work that utilizes a single multilayer perceptron (MLP) to represent the 3D scene efficiently and

continuously, extending even to unobserved regions. Nevertheless, when scaling up to larger scenes, e.g. an apartment consisting of multiple rooms, significant performance drops are observed in both the dense reconstruction and camera tracking accuracy. The key limiting factor of iMAP stems from its use of a single MLP to represent the entire scene, which only be updated globally with every new, potentially partial RGB-D observations. NICE-SLAM [37] employs hierarchical voxel grids and pre-trained decoders to generalize to larger scenes. ESLAM [10] proposes to replace the voxel grids with compact feature planes, significantly improving the processing speed. They estimate poses from a scene representation, prioritizing world-space geometry over image-space geometry, showing promising results on synthetic indoor datasets. However, these methods tend to oversmooth details in the reconstruction, which causes additional tracking errors. To address this challenge, Co-SLAM [27] uses parametric embeddings, while Point-SLAM [20] combines point cloud data with neural implicit representations to enhance reconstruction details. However, their remarkable performance heavily relies on accurate pixel-perfect depth supervision. But in the real world, the depth is usually noisy, and these methods degrade drastically in these scenarios.

A neural SLAM is continual learning system. An effective continual learning system should demonstrate both plasticity (the ability to acquire new knowledge) and stability (preserving old knowledge). Catastrophic forgetting is a well-known property of neural networks, and is failure of stability, where new experiences overwrite memories. One line of work on alleviating catastrophic forgetting has focused on protecting representations against new data using keyframe database. iMAP direct our attention towards the replay-based approach to continual learning, where previous knowledge is stored either directly in a buffer, or compressed in a generative model. iMAP use a straightforward method where keyframes are automatically selected to store and compress past memories. One can sample these keyframes in the continually running map update process to periodically replay and strengthen previously-observed scene regions, while continuing to add information via new keyframes.

In conclusion, in terms of a neural SLAM system, there are three major aspects to be improved. 1. Catastrophic forgetting. 2. Robust against noisy data. 3. Hole-filling ability. The key to resolve the problems is to improve the generalization of neural networks. The theory of generalization in

machine learning is concerned with understanding how well a trained model can perform on unseen data, i.e., data that was not used during training. It aims to answer the fundamental question: "How does a model generalize from the training data to new, unseen data?" In terms of neural SLAM, the unseen data refers to the unobserved region of the environment. Improving the generalization of neural networks can help the neural SLAM to have a better hole-filling ability. A generalizable model can also be robust against overfitting to noises in the data.

One of the methods to improve the generalization of neural networks is to increase the size and diversity of the training data [19]. It can help the model learn a broader range of patterns and relationships, leading to better generalization. In terms of a neural SLAM, one solution is to introduce heterogeneous data to the model. Using heterogeneous data can improve generalization in the model for several reasons: 1. Comprehensive representation. Heterogeneous data often comes from various sources or modalities. Combining these diverse sources of information can provide a more comprehensive representation of the underlying problem domain. By leveraging different types of data, the model can capture a broader range of features and patterns, leading to better generalization. [19] 2. Robustness to variability. Different data modalities may exhibit different forms of variability or noise. By incorporating heterogeneous data, the model learns to generalize across these variations more effectively. [3] 3. Reduced overfitting. Incorporating heterogeneous data can help mitigate overfitting by providing complementary information that regularizes the learning process. If one modality contains noisy or irrelevant features, other modalities may compensate for this noise, leading to a more robust model that generalizes better to unseen data. [3] 4. Robustness to missing data. In real-world scenarios, certain data modalities may be missing or incomplete. [23] By leveraging heterogeneous data, the model can still make accurate predictions even when some modalities are unavailable. This robustness to missing data can improve generalization performance in practical applications where data may be incomplete or noisy.

Another method to improve the generalization of neural networks is to improve the Lipschitz continuity of them. The relationship between generalization and Lipschitz continuity is an important aspect of understanding the behavior and performance of machine learning models, particularly in deep learning [21]. Lipschitz continuity is a mathematical property that describes how "smooth" or

"well-behaved" a function is. In the context of machine learning, Lipschitz continuity can have implications for the generalization performance of models. Lipschitz continuity is related to generalization in the sense that functions that are Lipschitz continuous tend to generalize better. Intuitively, Lipschitz continuity imposes a smoothness constraint on the function, preventing it from exhibiting rapid and unpredictable changes in its output as the input varies slightly. This smoothness property can lead to more stable and reliable predictions, which in turn can improve generalization performance [31]. In the context of deep learning, recent research [13], [4], [5] has shown that enforcing Lipschitz continuity constraints on neural networks can help improve their generalization performance. By limiting the magnitude of the gradients and ensuring smoother transitions between input-output mappings, Lipschitz regularization techniques can mitigate issues such as overfitting and adversarial vulnerability, leading to better generalization [12]. Lipschitz regularization techniques [5], such as weight clipping, gradient clipping, spectral normalization, and Lipschitz regularization penalties, are commonly used to enforce Lipschitz continuity in neural networks. These techniques aim to limit the Lipschitz constant of the network's layers or parameters, thereby promoting smoother function behaviors and improving generalization performance. In summary, Lipschitz continuity is closely related to the generalization performance of machine learning models. Enforcing Lipschitz continuity constraints, either through regularization techniques or architectural choices, can help improve the stability and robustness of models, leading to better generalization to unseen data.

Based on the three major problems and two solutions mentioned above, I would like to introduce my thesis goals and approaches.

1.2. Thesis Goals and Approaches

The primary objective of this thesis is to develop a comprehensive solution for enhancing the generalization capabilities of neural SLAM systems. This goal will be pursued through two main avenues: Lipschitz continuity and semantic-informed 3D reconstruction.

The first part of the thesis focuses on methodologies for estimating and regulating the Lipschitz constant through regularization and weight normalization. By effectively approximating the Lipschitz constant, the smoothness and robustness of neural networks (specifically, MLPs) can be assessed [13]. This approach also considers the impact of the activation function on the estimation of the

Lipschitz constant. A Lipschitz loss function, derived from the estimated Lipschitz constant, will be employed to globally constrain the Lipschitz constant of the neural network, thereby enhancing its generalization. Additionally, the proposed weight normalization technique will further improve Lipschitz continuity, leading to better reconstruction and localization performance.

The second part of the thesis explores the integration of semantic information into neural SLAM systems. The underlying premise is that combining diverse information can lead to improved overall performance. Semantic labels are utilized to categorize pixels, and multiple MLPs are employed to process the color and depth information of pixels from different classes, allowing each MLP to specialize in learning the features of a specific class. This specialization is expected to enhance the generalization of the overall architecture. Semantic labels also guide keyframe selection and pixel sampling, with frames exhibiting high pixel diversity being chosen for the keyframe database to mitigate the forgetting catastrophe. For each selected frame, pixels from each semantic class are sampled to ensure efficient feature learning by each MLP. Consequently, the MLPs are well-equipped to reconstruct unobserved regions and minimize redundancy in overlapping parts. Additionally, another set of MLPs is tasked with predicting semantic labels for each class, which are then used to calculate the semantic loss based on ground truth. By fully leveraging semantic information, the generalization performance of the neural SLAM system is further enhanced.

In conclusion, the thesis proposes a novel semantic-informed RGB-D neural SLAM system that combines the advantages of Lipschitz continuity and semantic integration to improve the generalization and overall performance of neural SLAM systems.

1.3. Contributions and Thesis Outline

The thesis presents three main contributions:

1) The introduction of weight normalization and Lipschitz regularization techniques significantly enhances the robustness and generalization of MLPs, leading to improved reconstruction and localization performance, particularly in noisy environments.

2) The proposed method maximizes the use of semantic information to guide keyframe selection, pixel sampling, and the training process. A semantic loss function is introduced to supervise the entire mapping and tracking process, further boosting the generalization capabilities of the MLPs.

3) The integration of the aforementioned approaches into a neural SLAM system results in a novel semantic-informed RGB-D neural SLAM that demonstrates state-of-the-art performance on the Replica [24] and Synthetic RGBD [2].

The structure of the thesis is as follows: Chapter one provides a review of the related literature on neural SLAM, Lipschitz continuity, generalization ability, and outlines the motivation and goals of this thesis. In chapter two, the approaches related to Lipschitz continuity are detailed. Chapter three introduces the semantic-informed RGB-D neural SLAM. The final chapter concludes the thesis, highlighting its limitations and suggesting directions for future work.

CHAPTER 2

LIPSCHITZ REGULARIZED NEURAL SLAM

2.1. Background in Lipschitz Networks

In the field of mathematical analysis, the concept of Lipschitz continuity is named after the German mathematician Rudolf Lipschitz. It represents a stringent form of uniform continuity for functions. Essentially, a function is Lipschitz continuous if there is a real number that serves as a cap on how quickly the function can change: for any two points on the function's graph, the absolute value of the slope of the line connecting these points does not exceed this real number. This smallest bound is known as the function's Lipschitz constant and is associated with the modulus of uniform continuity. For example, any function defined on an interval with a bounded first derivative is Lipschitz continuous [13].

In the realm of differential equations, Lipschitz continuity is a crucial condition in the Picard–Lindelöf theorem, which ensures the existence and uniqueness of a solution to an initial value problem. A specific form of Lipschitz continuity, called contraction, plays a key role in the Banach fixed-point theorem [14]. Here we give the definition of Lipschitz continuity.

Definition 1. A function $f : X \to Y$ between two metric spaces X and Y is said to be Lipschitz continuous if there exists a real number $L \geq 0$ (called the Lipschitz constant) such that for all points $x_1, x_2 \in X$, the following inequality holds:

$$|f(x_1) - f(x_2)| \leq L \cdot |x_1 - x_2|.$$

Any value of L that satisfies the Lipschitz continuity condition is known as a Lipschitz constant for the function f, and f can also be referred to as L-Lipschitz. The smallest such constant is often termed the (best) Lipschitz constant of f or alternatively, the dilation or dilatation of f. If $L = 1$, the function is described as a short map, and if $0 \leq L < 1$ with f mapping a metric space onto itself, the function is termed a contraction.

For functions of several real variables that map to real values, the condition is met if and only if

the absolute values of the slopes of all secant lines are bounded by L. The collection of lines with slope L passing through a point on the graph of the function forms a circular cone. A function is Lipschitz if and only if its graph lies entirely outside of this cone at all points.

A function is termed locally Lipschitz continuous [33] if, for every point x in the domain X, there exists a neighborhood U of x such that the function restricted to U is Lipschitz continuous. Alternatively, if X is a locally compact metric space, then the function is locally Lipschitz if and only if it is Lipschitz continuous on every compact subset of X. In spaces that are not locally compact, this condition is necessary but not sufficient for local Lipschitz continuity.

Lipschitz continuity plays a significant role in the generalization of neural networks. Generalization refers to the ability of a neural network to perform well on unseen data, and it is a crucial aspect of machine learning. Here's how Lipschitz continuity influences the generalization of neural networks:

1. Robustness to Adversarial Examples [22]: Neural networks with Lipschitz continuity constraints are more robust to adversarial examples. Adversarial examples are inputs that are slightly perturbed to deceive the network into making incorrect predictions. By enforcing Lipschitz continuity, the sensitivity of the network to small perturbations in the input is reduced, making it harder for adversaries to craft successful attacks.

2. Stability of Training [8]: Lipschitz continuity can lead to more stable training of neural networks. When the Lipschitz constant is bounded, the gradients used for training are also bounded, which can prevent issues like exploding gradients. This stability can lead to better convergence and more reliable training outcomes.

3. Regularization Effect [32]: Enforcing Lipschitz continuity can act as a form of regularization, which helps to prevent overfitting. By limiting the rate of change of the function, the network is encouraged to learn smoother, more generalizable functions that perform better on unseen data.

4. Implicit Bias Towards Simplicity [31]: Neural networks with Lipschitz constraints tend to have an implicit bias towards simpler functions, which often generalize better. This is related to the concept of Occam's razor, which suggests that simpler models are more likely to generalize well.

5. Influence on Loss Landscape [12]: Lipschitz continuity can affect the loss landscape of neural networks. A smoother loss landscape, resulting from Lipschitz constraints, can make optimization

easier and lead to better generalization performance.

In practice, enforcing Lipschitz continuity in neural networks can be challenging, especially in deep networks. Techniques such as weight normalization [1], gradient clipping [18], and spectral normalization [28] are often used to control the Lipschitz constant of the network or its layers. Overall, considering Lipschitz continuity in the design and training of neural networks can lead to models that are more robust, stable, and capable of generalizing well to new data.

Weight normalization, gradient clipping, and spectral normalization are techniques used to control the Lipschitz constant of a multi-layer perceptron (MLP), each with its advantages and disadvantages:

Weight normalization is advantageous in that it improves the conditioning of the optimization problem, leading to faster convergence, and can stabilize the training process by ensuring that the scale of the weights does not grow too large. However, it has disadvantages as well, such as not directly controlling the Lipschitz constant since it only normalizes the weights without considering the overall effect on the function's Lipschitz constant, and it may not be sufficient to prevent issues like exploding gradients in deep networks.

Gradient clipping, on the other hand, has the advantage of directly addressing the issue of exploding gradients by limiting the maximum norm of the gradient, which indirectly helps control the Lipschitz constant. It is also simple to implement and can be applied to any gradient-based optimization algorithm. However, its disadvantages include the somewhat arbitrary choice of clipping threshold, which may require tuning for different problems, and the fact that it does not directly control the Lipschitz constant of the function, only the gradients during training.

Spectral normalization directly controls the Lipschitz constant by normalizing the weight matrices based on their spectral norm, which is the largest singular value. This has been shown to improve the stability and generalization of deep networks, especially in adversarial settings. However, spectral normalization is computationally more expensive than weight normalization, as it requires computing the spectral norm of the weight matrices, and it may lead to slower convergence in some cases due to the constraint on the weight matrices.

Estimating the Lipschitz constant of an MLP can be challenging due to the non-linearity and depth

of the network [21]. However, some approaches can be used, such as empirical estimation, where the Lipschitz constant is estimated by computing the maximum ratio of the change in the output to the change in the input over a sample of input pairs. Another approach is layer-wise estimation, where the Lipschitz constant of an MLP is estimated as the product of the Lipschitz constants of individual layers, which corresponds to the spectral norm of the weight matrix for linear layers and depends on the properties of the function for non-linear activation functions. Spectral norm bounds also provide an upper bound on the Lipschitz constant by controlling the spectral norm of the weight matrices. In practice, the exact Lipschitz constant may be difficult to compute, and these methods provide approximate estimates or upper bounds that can be useful for controlling the behavior of the network during training and inference.

2.2. Method

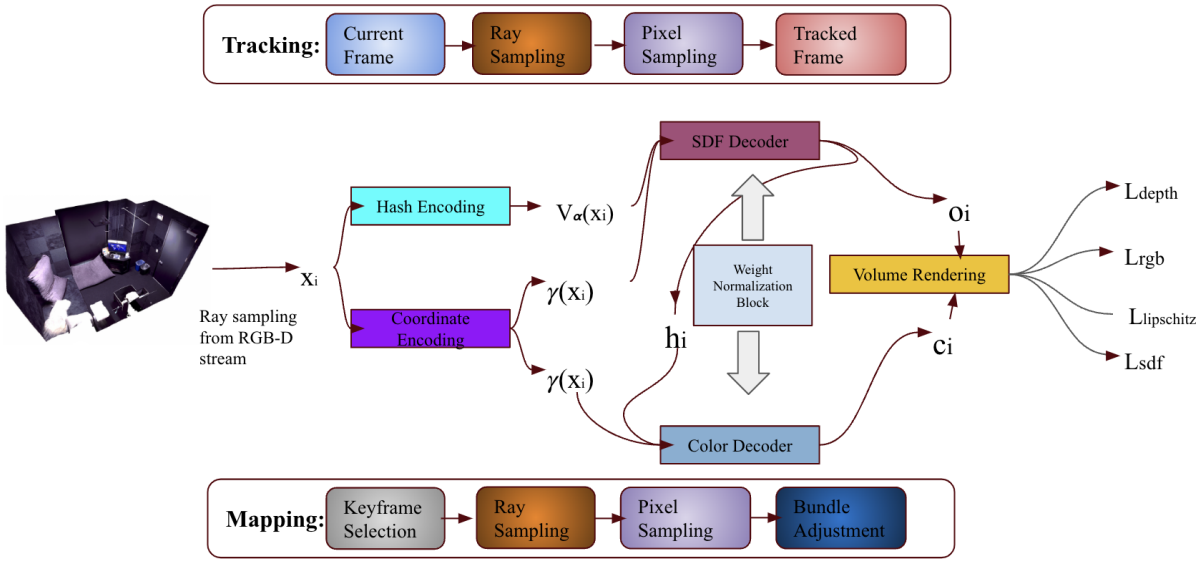


Figure 2.1: **Overview of Lipschitz regularized Neural SLAM.** For each point along the ray, I query a 3D feature from the hash lookup table and coordinate embedding. The outputs of the encoders are sent to the SDF decoder and the color decoder respectively to obtain RGB and SDF values. Meanwhile, the proposed weight normalization is performed in the decoders. Compute the color loss, the depth loss, the SDF loss, and the Lipschitz loss with the respective ground-truth maps and estimated Lipschitz constants. The scene representation and camera parameters are optimized during mapping, tracking, and bundle adjustment.

The proposed Lipschitz regularization is formulated as follows:

Given a neural network Φ with parameters $\theta = \{W_1, W_2, \ldots, W_L\}$ representing the weight matrices of each layer L, the regularization term is defined as the product of the per-layer Lipschitz bounds, which is then added to the original loss function $L(\theta)$ to form the augmented loss function $J(\theta)$:

$$J(\theta) = L(\theta) + \lambda \prod_{l=1}^{L} \text{softplus}(\gamma_l)$$

Here, γ_l is a learnable parameter representing the logarithm of the Lipschitz bound for layer l, and λ is a regularization coefficient. The softplus function is used to ensure that the Lipschitz bounds are positive. The weight matrices W_l are normalized in each layer according to the corresponding Lipschitz bound $\text{softplus}(\gamma_l)$ to ensure that the network satisfies the desired Lipschitz continuity.

The product of the per-layer Lipschitz bounds in the Lipschitz regularization proposed by the authors [5] is given by:

$$\prod_{l=1}^{L} \text{softplus}(\gamma_l)$$

where L is the total number of layers in the neural network. γ_l is the learnable parameter representing the logarithm of the Lipschitz bound for layer l. $\text{softplus}(\cdot)$ is the softplus function, defined as $\text{softplus}(x) = \ln(1 + e^x)$.

This product term represents the overall Lipschitz bound of the network, which the regularization term aims to minimize in order to encourage smoothness in the learned functions.

This regularization encourages the network to have a small overall Lipschitz constant, promoting smoothness in the learned functions.

The product of the per-layer Lipschitz bounds is a key component of the Lipschitz regularization technique proposed in the paper. Here's a detailed explanation:

Lipschitz continuity is a concept that ensures a function behaves smoothly by controlling its rate of change. In the context of neural networks, this means ensuring that the output of the network does not change too rapidly with respect to changes in the input. The Lipschitz constant, K, is a measure of this rate of change, and a function is Lipschitz continuous if there exists such a K that satisfies the inequality $\|f(x_1) - f(x_2)\| \leq K\|x_1 - x_2\|$ for all input pairs x_1, x_2.

To control the smoothness of a neural network, one approach is to control the Lipschitz constant

of each layer. This is achieved by introducing a learnable parameter γ_l for each layer l, which represents the logarithm of the layer's Lipschitz bound. The actual bound for the layer is obtained using the softplus function, ensuring that the bound is always positive, as the Lipschitz constant should not be negative.

The overall Lipschitz constant of the network can be bounded by the product of the per-layer bounds [13]. This is because the rate of change of each layer could compound through the layers in the worst case. By controlling the Lipschitz constant of each layer, the overall smoothness of the network can be controlled. The product of the per-layer bounds is given by $\prod_{l=1}^{L} \text{softplus}(\gamma_l)$, where L is the total number of layers, γ_l is the learnable parameter for layer l, and $\text{softplus}(\cdot)$ is the softplus function.

This product of per-layer bounds is used as a regularization term in the loss function of the network. By adding this term to the original loss function, the training process is encouraged to find parameters that not only minimize the original loss but also maintain a small overall Lipschitz constant, promoting smoothness in the learned function. This approach helps ensure that the neural network learns a function that behaves smoothly, which can be important for generalization and robustness. The structure of the Lipschitz regularized neural SLAM mirrors that of Co-SLAM. For each point along the ray, a 3D feature is retrieved using a hash lookup table and coordinate embedding. The outputs from the encoders are then fed into the SDF decoder and the color decoder to generate SDF and RGB values, respectively. Additionally, weight normalization is applied within the decoders. The color loss $\mathcal{L}_{\text{rgb}}$, depth loss $\mathcal{L}_{depth}$, SDF loss $\mathcal{L}_{\text{sdf}}$, $\mathcal{L}_{\text{fs}}$, and Lipschitz loss $\mathcal{L}_{\text{lipschitz}}$ are calculated using the corresponding ground-truth maps and estimated Lipschitz constants. During the processes of mapping, tracking, and bundle adjustment, the scene representation and camera parameters are refined.

The complete loss function is defined as follows:

$$\mathcal{L} = \lambda_{\text{lipschitz}}\mathcal{L}_{\text{lipschitz}} + \lambda_{\text{rgb}}\mathcal{L}_{\text{rgb}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{sdf}}\mathcal{L}_{\text{sdf}} + \lambda_{\text{fs}}\mathcal{L}_{\text{fs}}$$

Here, $\lambda_{\text{lipschitz}}$, λ_{rgb}, λ_{depth}, λ_{sdf}, and λ_{fs} represent the weighting factors for the Lipschitz, color, depth, truncated signed distance field (TSDF) near the surface, and TSDF in free space loss func-

tions, respectively.

2.3. Experiments

We first describe our experimental setup and then evaluate our method against state-of-the-art dense neural RGBD SLAM methods on synthetic and real-world datasets. In addition, we compare our method with concurrent work with released source code.

2.3.1. Implementation Details

All experiments are conducted using Python and the PyTorch framework on a desktop PC equipped with a 5.5GHz Intel Core i9-13900K CPU and NVIDIA RTX3080 GPU. For camera tracking, we set $N_t = 1024$ pixels and perform 10 iterations of tracking with $M_c = 32$ for regular sampling and $M_f = 11$ for depth-guided sampling per camera ray. In mapping and bundle adjustments, $N_g = 2048$ pixels are selected, utilizing 200 iterations for initial frame mapping and 10 iterations for bundle adjustment every 5 frames. Scene representations employ an $L = 16$ level HashGrid ranging from $R_{\text{min}} = 16$ to R_{max}, with the maximum voxel size of 2 cm determining R_{max}, and 16 bins for OneBlob encoding of each dimension. Two 2-layer shallow MLPs with 32 hidden units each decode color and SDF. The dimension of the geometric feature $\mathbf{h}$ is 15. Training of our scene representation uses learning rates of 1×10^{-3} for tracking and 1×10^{-2}, 1×10^{-2}, and 1×10^{-3} for the feature grid, decoder, and camera parameters during bundle adjustment, respectively. The weights of each loss are $\lambda_{\text{c}} = 5$, $\lambda_d = 0.1$, $\lambda_{\text{sdf}} = 1000$, $\lambda_{\text{fs}} = 10$, and $\lambda_{\text{lipschitz}} = 1 \times 10^{-6}$. The truncation distance tr is set to 10 cm.

2.3.2. Datasets

The Replica Dataset [24] is a large-scale indoor scene dataset that is commonly used for research in computer vision and robotics. It provides detailed 3D reconstructions of indoor scenes, including homes and offices, captured from various real-world locations. The dataset includes RGB images, depth images, surface normals, semantic segmentations, and instance segmentations, along with camera poses and calibration information. Replica is often used for tasks such as 3D scene understanding, navigation, and manipulation in indoor environments.

The Neural RGBD dataset [2] is created for the purpose of evaluating the performance of the proposed 3D surface reconstruction method. The dataset comprises 10 different scenes, each with its own unique characteristics in terms of size, complexity, and material properties. These scenes are designed to test the reconstruction algorithm's ability to handle various challenges, such as specular surfaces, mirrors, and textureless regions. The scenes in the dataset are obtained from sources like BlendSwap and the ICL-NUIM dataset [7], and they are rendered using BlenderProc to generate color and depth images for each camera pose in a predefined trajectory. The depth maps are further processed to simulate sensor noise, similar to what would be encountered with real depth sensors. For the evaluation, the dataset provides ground truth camera trajectories and meshes. The reconstruction quality is assessed using metrics such as Chamfer distance, intersection-over-union (IoU), normal consistency, and F-score. These metrics evaluate how well the reconstructed 3D models match the ground truth in terms of shape, completeness, and surface normal accuracy.

2.3.3. Evaluation Metrics

To assess tracking performance, we utilize the Absolute Trajectory Error (ATE) Root Mean Square Error (RMSE). For reconstruction performance, we compute Accuracy, Completion, Completion Ratio, and Depth L1. To evaluate mapping performance, we first generate meshes using the marching cubes algorithm. Accuracy is defined as the average distance from a point on the ground truth mesh to its nearest point on the reconstructed mesh; thus, smaller values indicate better performance. Completion measures the average distance from sampled points on the ground truth mesh to the nearest point on the reconstructed mesh, with smaller values being preferable. The Completion Ratio represents the percentage of points in the reconstructed mesh that have a completion distance of less than 5 centimeters, where larger percentages are better. Depth L1 is a 2D metric that quantifies the average L1 difference between rendered ground truth depth and reconstructed depth, with smaller values indicating improved accuracy.

2.3.4. Baseline Methods

I consider iMAP, NICE-SLAM, and Co-SLAM as my main baselines for reconstruction quality and camera tracking.

The iMAP system [25], while offering real-time SLAM capabilities with its innovative use of a multilayer perceptron for scene representation, has several shortcomings. It demands high computational resources for its continual training of a neural network, which may not be feasible on all hardware. The network's training on specific scenes in real time limits its ability to generalize to new environments without retraining. Sparse pixel sampling, used to manage computational load, might compromise the detail and accuracy of the reconstructed map. The reliance on keyframes for mapping and tracking means that the selection quality of these keyframes is crucial for overall performance. Handling dynamic objects in the scene is also a challenge for iMAP, as it can lead to tracking inaccuracies and mapping errors. Additionally, continuous map updates with new keyframes necessitate efficient memory management to avoid performance degradation over time.

NICE-SLAM [37], while promising in its scalability and efficiency for large-scale scenes, has some potential shortcomings. First, the reliance on hierarchical grid-based representations and neural implicit decoders might pose challenges in handling highly dynamic environments or scenes with significant occlusions. Second, the system's performance could be sensitive to the quality of the RGB-D input data, where low-quality or noisy data may affect the accuracy of the reconstruction and tracking. Third, the computational complexity of the method, although improved compared to previous approaches, might still be demanding for real-time applications on limited hardware. Additionally, the method's reliance on pre-trained geometric priors might limit its adaptability to novel or significantly different environments from those seen during training. Lastly, like many SLAM systems, NICE-SLAM might face difficulties in dealing with loop closure and relocalization in large-scale environments.

Co-SLAM [27], while offering robust camera tracking and high-fidelity surface reconstruction in real-time, might face some challenges. One potential shortcoming could be its reliance on a multi-resolution hash-grid and one-blob encoding for scene representation. While these features enable fast convergence and the ability to represent high-frequency local features, they may also limit the

system's adaptability to diverse environments or its ability to handle highly dynamic scenes. Additionally, while the system aims for real-time performance, the actual frame rate of 10-17Hz might still be insufficient for certain high-speed applications. Furthermore, like many SLAM systems, Co-SLAM may face challenges in dealing with large-scale environments, loop closure, and relocalization. Lastly, the system's performance could be dependent on the quality of the RGB-D input data, where low-quality or noisy data may affect the accuracy of the reconstruction and tracking.

2.3.5. Reconstruction and Localization Performance

In the comparative evaluation of my proposed approach against established baselines, namely iMAP, Nice-SLAM, and Co-SLAM, extensive analyses were conducted on two distinct datasets: the Replica Dataset and a synthetic dataset sourced from NeuralRGBD.

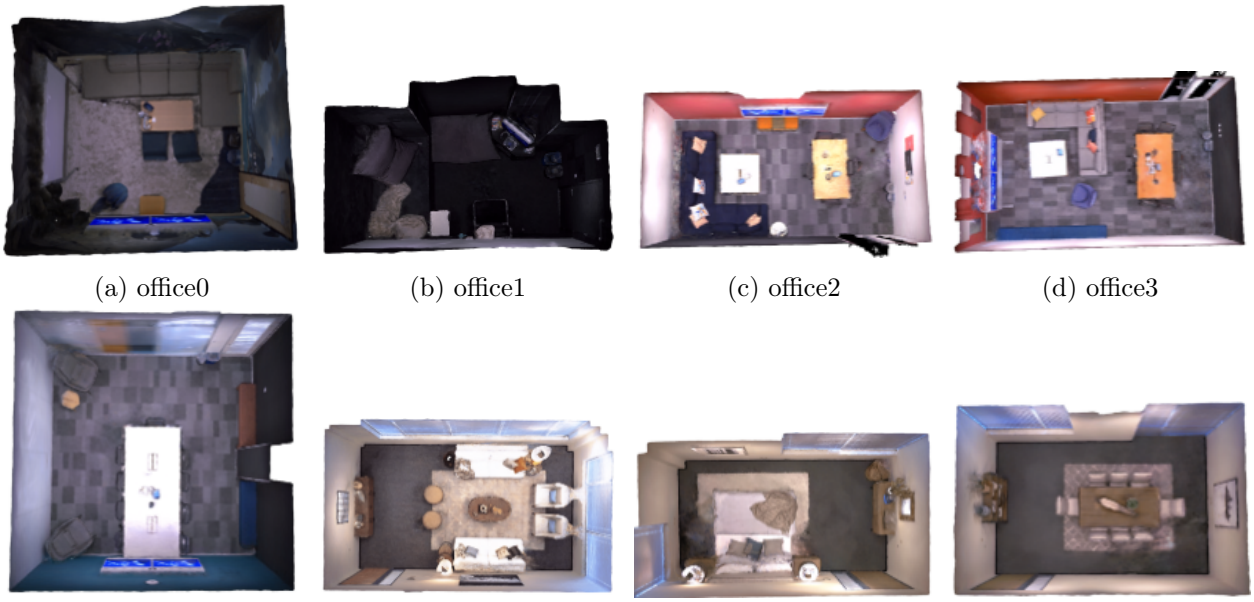
In the evaluation using the Replica Dataset, renowned for its pristine quality devoid of inherent noise, it was observed that my approach consistently outperforms the baseline methods. Despite this achievement, the magnitude of improvement may not be immediately striking. This nuanced observation can be attributed to the controlled environment of the Replica Dataset, which poses minimal challenges for traditional neural SLAM frameworks. Consequently, while my approach aims to enhance the robustness of neural SLAM systems, its performance metrics may not significantly surpass those of the baseline methods, such as Co-SLAM, particularly in certain evaluation metrics. However, my approach demonstrates notable strengths in augmenting the generalization capabilities of the underlying neural network architecture, particularly in scenarios involving previously unseen regions. Thus, even though some metrics may not exhibit substantial improvements over existing methods, my approach showcases commendable performance where generalization is paramount, offering promising avenues for further exploration and refinement.

In contrast, the synthetic dataset sourced from NeuralRGBD introduces a different set of challenges, characterized by the presence of noise within the depth maps. In this scenario, my approach's advantages become more pronounced, showcasing marked improvements over the baseline methods. This can be attributed to the inherent benefits stemming from the incorporation of restricted Lipschitz constants within my approach's framework, which endows it with heightened resilience against

Method	Metrics	rm0	rm1	rm2	off0	off1	off2	off3	off4
iMAP	Accuracy↓	4.01	3.04	3.04	3.34	2.10	4.06	4.20	4.34
	Completion↓	3.20	4.40	5.84	3.63	3.62	4.73	5.49	6.65
	Completion Ratio↑	78.34	85.85	85.85	83.59	88.45	79.73	73.93	74.77
	Depth L1↓	3.44	3.44	5.08	3.79	3.76	3.97	5.61	5.71
NICE-SLAM	Accuracy↓	2.44	2.10	2.17	1.85	1.56	3.28	3.01	2.54
	Completion↓	2.60	2.19	2.73	1.84	1.82	3.11	3.16	3.61
	Completion Ratio↑	91.81	93.56	91.48	94.93	94.11	88.27	87.68	87.23
	Depth L1↓	1.79	1.33	2.20	1.43	1.58	2.70	2.10	2.06
Co-SLAM	Accuracy↓	2.11	1.68	1.99	1.57	1.31	2.84	**3.06**	**2.23**
	Completion↓	**2.02**	**1.81**	1.96	1.56	1.59	**2.43**	2.72	2.52
	Completion Ratio↑	**95.26**	95.19	93.58	96.09	94.65	91.63	90.72	90.44
	Depth L1↓	1.05	**0.85**	2.37	1.24	1.48	1.86	1.66	1.54
Ours	Accuracy↓	**1.99**	**1.66**	**1.92**	**1.53**	**1.25**	**2.71**	3.10	**2.27**
	Completion↓	2.16	1.83	**1.95**	**1.55**	**1.32**	2.49	**2.69**	**2.49**
	Completion Ratio↑	94.43	**95.22**	**93.64**	**96.20**	**94.88**	**91.86**	**90.93**	**91.11**
	Depth L1↓	**0.97**	0.97	**2.35**	**1.24**	**1.35**	**1.85**	**1.59**	**1.51**

Table 2.1: Reconstruction performance on Replica dataset.

the detrimental effects of noise originating from depth sensors.



(a) office0 (b) office1 (c) office2 (d) office3

(e) office4 (f) room0 (g) room1 (h) room2

Figure 2.2: Qualitative reconstruction results on Replica dataset.

Method	Metrics	br	ck	gr	gwr	ma	tg	wr
iMAP	Accuracy↓	10.56	25.16	13.01	11.90	29.62	12.98	24.82
	Completion↓	11.27	31.09	19.17	20.39	49.22	21.07	32.63
	Completion Ratio↑	46.91	12.96	21.78	20.48	10.72	19.17	13.07
	Depth L1↓	24.03	63.59	26.22	21.32	61.29	29.16	81.71
NICE-SLAM	Accuracy↓	3.44	10.92	5.34	2.63	6.55	3.57	9.22
	Completion↓	3.69	12.00	4.94	3.15	3.13	5.28	4.89
	Completion Ratio↑	87.69	55.41	82.78	87.72	85.04	72.05	71.56
	Depth L1↓	3.66	12.08	10.88	2.57	1.72	7.74	5.59
Co-SLAM	Accuracy↓	1.97	4.68	2.10	1.89	1.60	3.38	5.03
	Completion↓	1.93	4.94	2.96	2.16	2.67	2.74	3.34
	Completion Ratio↑	94.75	68.91	90.80	95.04	86.98	86.74	84.94
	Depth L1↓	3.51	5.62	1.95	1.25	1.41	4.66	2.74
Ours	Accuracy↓	**1.43**	**4.12**	**2.01**	**1.43**	**1.36**	**2.99**	**4.66**
	Completion↓	**1.45**	**4.32**	**2.55**	**1.72**	**2.11**	**2.55**	**2.89**
	Completion Ratio↑	**95.72**	**74.78**	**91.22**	**96.53**	**90.42**	**88.67**	**86.58**
	Depth L1↓	**2.72**	**4.88**	**1.88**	**1.01**	**1.11**	**3.87**	**1.97**

Table 2.2: Reconstruction performance on Neural RGBD dataset.

Method	rm0	rm1	rm2	off0	off1	off2	off3	off4
iMAP	5.23	3.09	2.58	2.40	1.17	5.67	5.08	2.23
NICE-SLAM	1.69	2.13	1.87	1.26	0.84	1.71	3.98	2.82
Co-SLAM	0.69	**0.83**	**1.02**	0.58	0.55	2.22	**1.30**	0.87
Ours	**0.65**	1.01	1.17	**0.55**	**0.53**	**2.16**	1.40	**0.72**

Table 2.3: Tracking performance (ATE RMSE (cm)) on Replica dataset.

CHAPTER 3

SEMANTIC-INFORMED NEURAL DENSE VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING

3.1. Background in Semantic-informed Neural SLAM

Different types of data can provide complementary information about the same phenomenon [19]. For instance, in a multimodal learning task, images might provide visual information, while text provides contextual or semantic information. Combining these data types can lead to a more comprehensive understanding of the task at hand. Heterogeneous data can help the model become more robust to noise and variability in individual data types. For example, if one data source is noisy or incomplete, information from other sources can compensate for this deficiency. Using data from different sources or modalities can increase the diversity of training examples, which can help the model learn more general patterns and reduce overfitting to specific characteristics of a single data type [23]. Heterogeneous data allows for the transfer of knowledge across different domains or modalities [3]. This can be particularly beneficial in scenarios where data is scarce in one domain but abundant in another. By learning from heterogeneous data, a deep learning model can develop more sophisticated and higher-level feature representations that capture a broader range of characteristics of the input data. In tasks that inherently involve multiple data types, leveraging heterogeneous data is essential for achieving high performance. Real-world scenarios often involve complex, multifaceted data. Models trained on heterogeneous data are better equipped to handle this complexity and make more accurate predictions in diverse situations. In summary, heterogeneous data can enhance the generalization ability of deep learning models by providing a richer, more diverse set of information, which leads to more robust and accurate models that perform well across different scenarios and tasks.

In my case, I integrate semantic information in RGB-D neural SLAM to improve the generalization. Integrating semantic information into RGB-D data for 3D reconstruction using neural networks offers several benefits. Semantic information can guide the neural network to generate more accurate and detailed reconstructions, particularly for objects and surfaces with distinct semantic categories.

By understanding the context and nature of different objects, the network can better infer their shapes and appearances. With semantic information, the reconstruction process can be performed at the object level, allowing for the creation of semantically segmented 3D models. This is particularly useful for applications that require interaction with specific objects or detailed analysis of individual components within a scene. Semantic understanding can help the network infer the shape and appearance of partially occluded or missing parts of objects, leading to more complete and coherent reconstructions. By leveraging semantic information, the network can learn more effectively from limited data, as it can generalize from the semantic context of the scene [36]. This can be particularly beneficial when dealing with diverse environments and object types. Semantic 3D reconstruction provides not only geometric information but also semantic labels for each part of the scene, enabling a deeper understanding of the environment. This can be valuable for applications like robotic navigation, virtual reality, and augmented reality, where interaction with the environment is key.

3.2. Method

In this thesis, I introduce a novel approach to SLAM using a neural RGB-D semantic framework. The method leverages 2D semantic maps to construct an additional scene representation. I present a hybrid representation that leverages 2D semantic priors for class-wise scene representation, which enables stable camera tracking and improved appearance details in reconstructions.

Figure 3.1 presents a detailed view of the semantic-informed neural SLAM system. It starts by taking an input RGB-D stream $\{I_t\}_{t=1}^{N}\{D_t\}_{t=1}^{N}$ and a semantic map stream $\{S_t\}_{t=1}^{N}$, alongside known camera intrinsics $K \in \mathbb{R}^{3\times3}$. The system performs dense mapping and tracking by simultaneously optimizing the camera poses $\{\xi_t\}_{t=1}^{N}$. In this process, the implicit function f_θ translates world coordinates x into color c, truncated signed distance (TSDF) values o, and semantic categories s:

$$f_\theta(x) \rightarrow (c, o, s)$$

The procedure is divided into tracking and mapping phases. Initially, a few training iterations are conducted on the first frame for setup. For each following frame, camera pose optimization is prioritized, starting from an assumption of constant-speed motion. A select number of pixels

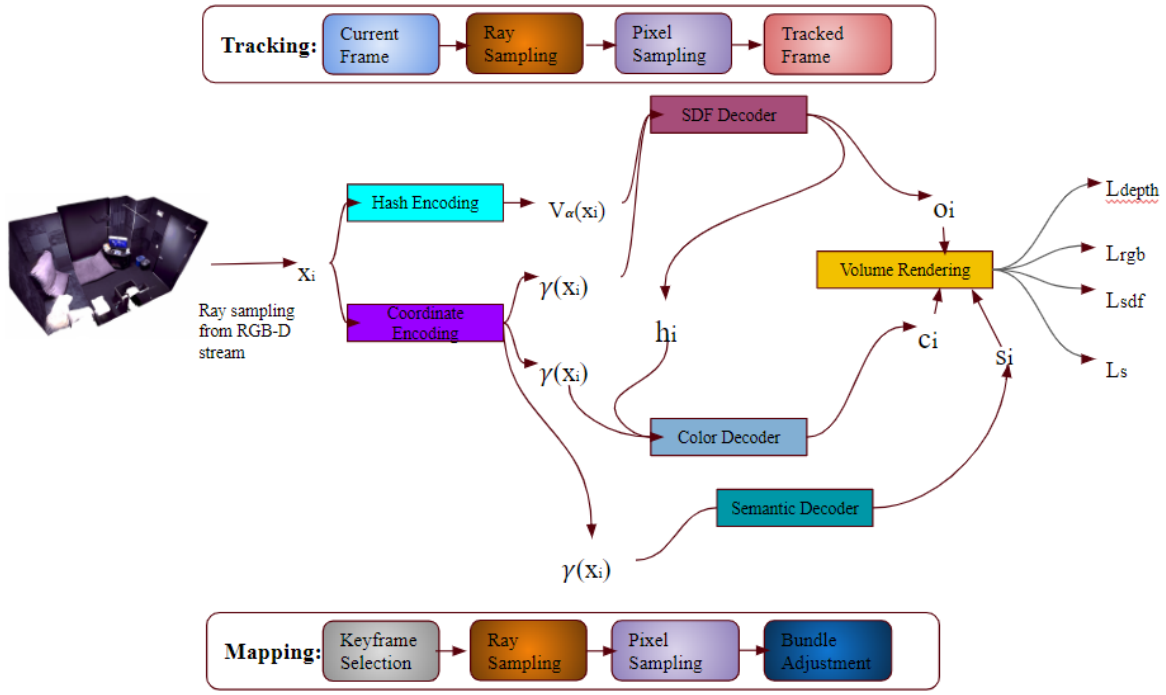


Figure 3.1: **Overview of Semantic-Informed Neural SLAM.** For each point along the ray, I query a 3D feature from the hash lookup table and coordinate embedding. The outputs of the encoders are sent to the SDF decoder, the color decoder, and the semantic decoder respectively to obtain RGB, SDF values, and semantic logits. Compute the color loss, the depth loss, the SDF loss, and the semantic loss with the respective ground-truth maps. The scene representation and camera parameters are optimized during mapping, tracking, and bundle adjustment.

are sampled and integrated into the global pixel set. During each mapping phase, a global bundle adjustment is carried out on randomly selected pixels from this set to concurrently refine the scene's representation θ and all camera poses $\{\xi_t\}$. The system incorporates OneBlob encoding $\gamma(x)$ and a multi-resolution hash grid $V_\alpha = \{\nu_\alpha^l\}_{l=1}^L$ to encode spatial points. The SDF decoder then computes the predicted SDF value o and a feature vector h:

$$f_\beta(\gamma(x), V_\alpha(x)) \to (h, o).$$

The color decoder predicts the RGB value:

$$f_\phi(\gamma(x), h) \to c.$$

The semantic decoder predicts the semantic logit:

$$f_\tau(\gamma(x), V_\alpha(x)) \rightarrow s$$

All three decoders are shallow MLPs, and $\theta = \{\alpha, \beta, \phi, \tau\}$. The rendering strategy for color and depth is the same as Co-SLAM. For the semantic value of each pixel in one frame, I simply sum up the value of the sample points along the ray to be the semantic prediction of the pixel.

$$\hat{\mathbf{s}} = \sum_{i=1}^{M} s_i$$

, where M is the number of the samples along the ray.

The loss functions for depth, color, and TSDF are the same as the ones of Co-SLAM. Here I introduce the multi-class cross-entropy loss function for the semantic information:

$$\mathcal{L}_s = \sum_{u \in |I|} s(u) log(\hat{s}(u))$$

, where u is the pixel in one frame. Thus, the complete loss function is

$$\mathcal{L} = \lambda_s \mathcal{L}_s + \lambda_{rgb} \mathcal{L}_{rgb} + \lambda_{depth} \mathcal{L}_{depth} + \lambda_{sdf} \mathcal{L}_{sdf} + \lambda_{fs} \mathcal{L}_{fs}$$

, where λ_s, λ_{rgb}, λ_{depth}, λ_{depth}, λ_{sdf} and λ_{fs} are respective weighting factors for semantic, color, depth, TSDF(near surface), and TSDF(free space) loss functions.

3.3. Experiments

3.3.1. Implementation Details

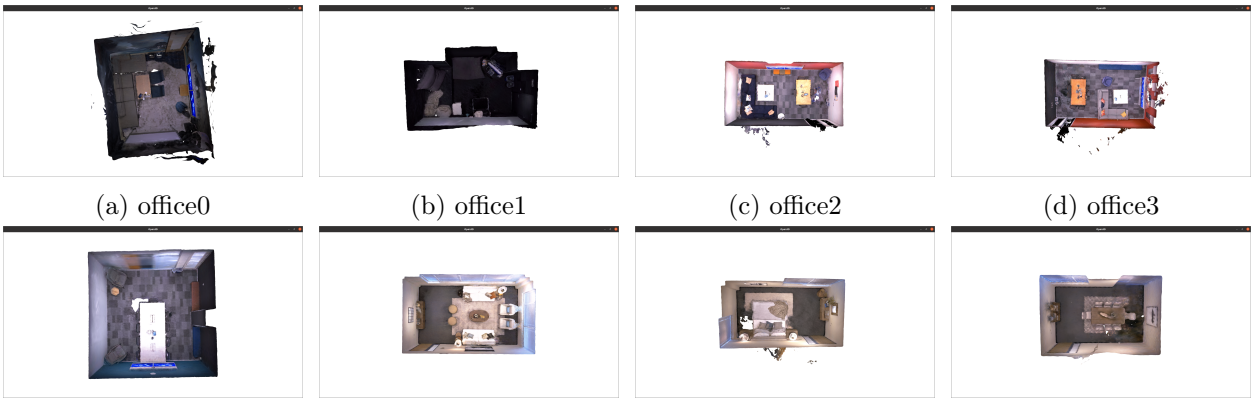
The proposed approach is evaluated on the Replica using a semantic-informed neural SLAM framework, and four metrics are used for assessment. Accuracy is defined as the average distance from a point on the ground truth mesh to its nearest point on the reconstructed mesh, with lower values indicating better accuracy. Completion measures the average distance between sampled points from the ground truth mesh and the nearest point on the reconstructed mesh, where lower values are

desirable. The completion ratio represents the percentage of points in the reconstructed mesh with a completion distance of less than 5 cm, with higher values indicating better completion. Lastly, Depth L1 is a 2D metric that calculates the average L1 difference between the rendered ground truth depth and the reconstructed depth, with lower values signifying better depth reconstruction. We select iMAP, NICE-SLAM, and Co-SLAM as our baseline methods.

I employ six levels of hash grids ($L = 6$) and use 16 bins per dimension for the OneBlob encoding. For the decoders, I utilize two separate two-layer MLPs with 32 hidden units each to decode color and SDF. Additionally, I use a single two-layer MLP with 128 hidden units for decoding semantic labels. The loss weights are set as follows: $\lambda_{rgb} = 5$ for color, $\lambda_{depth} = 0.1$ for depth, $\lambda_{sdf} = 1000$ for the signed distance function, $\lambda_{fs} = 10$ for feature strength, and $\lambda_s = 0.01$ for semantic loss. The truncation distance tr is set at 10 cm.

For camera tracking, I select 1024 pixels and conduct 10 iterations. In mapping and bundle adjustments, I choose 2048 pixels and undertake 200 iterations for the initial frame mapping and 10 iterations for bundle adjustments every 5 frames. The learning rates are set at 1×10^{-2} for the feature grid and decoders, and 1×10^{-3} for camera parameters during bundle adjustments.

3.3.2. Reconstruction and Localization Performance



(a) office0 (b) office1 (c) office2 (d) office3

(e) office4 (f) room0 (g) room1 (h) room2

Figure 3.2: Qualitative reconstruction results on Replica dataset.

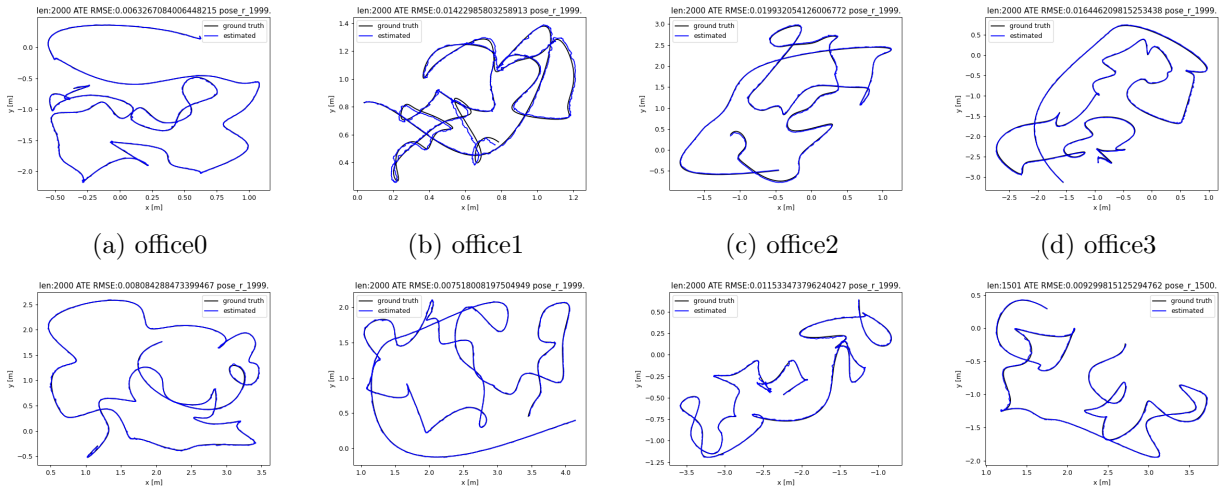
I present quantitative results on Replica in Table 3.1, where my method does not surpass Co-SLAM in terms of reconstruction and localization performance for most scenes. There are four reasons for this. First, using only one MLP is insufficient for fully leveraging the semantic information in

Method	Metrics	rm0	rm1	rm2	off0	off1	off2	off3	off4
iMAP	Accuracy↓	4.01	3.04	3.04	3.34	2.10	4.06	4.20	4.34
	Completion↓	3.20	4.40	5.84	3.63	3.62	4.73	5.49	6.65
	Completion Ratio↑	78.34	85.85	85.85	83.59	88.45	79.73	73.93	74.77
	Depth L1↓	3.44	3.44	5.08	3.79	3.76	3.97	5.61	5.71
NICE-SLAM	Accuracy↓	2.44	2.10	2.17	1.85	1.56	3.28	3.01	2.54
	Completion↓	2.60	2.19	2.73	1.84	1.82	3.11	3.16	3.61
	Completion Ratio↑	91.81	93.56	91.48	94.93	94.11	88.27	87.68	87.23
	Depth L1↓	1.79	1.33	2.20	1.43	1.58	2.70	2.10	2.06
Co-SLAM	Accuracy↓	2.11	1.68	1.99	1.57	1.31	2.84	**3.06**	2.23
	Completion↓	2.02	**1.81**	**1.96**	**1.56**	**1.59**	2.43	2.72	2.52
	Completion Ratio↑	**95.26**	**95.19**	**93.58**	**96.09**	**94.65**	91.63	90.72	90.44
	Depth L1↓	1.05	**0.85**	**2.37**	**1.24**	**1.48**	**1.86**	1.66	**1.54**
Ours	Accuracy↓	2.65	6.30	3.23	9.28	1.89	3.78	4.89	**2.21**
	Completion↓	**2.00**	1.89	2.46	1.75	1.91	**2.40**	**2.66**	**2.34**
	Completion Ratio↑	95.01	94.73	89.60	95.33	93.75	**91.90**	**90.90**	**91.19**
	Depth L1↓	**1.01**	1.40	2.58	1.32	1.76	2.06	**1.65**	2.38

Table 3.1: Reconstruction performance on Replica dataset.

Method	rm0	rm1	rm2	off0	off1	off2	off3	off4
iMAP	5.23	3.09	2.58	2.40	1.17	5.67	5.08	2.23
NICE-SLAM	1.69	2.13	1.87	1.26	0.84	1.71	3.98	2.82
Co-SLAM	**0.69**	**0.83**	1.02	0.58	**0.55**	2.22	**1.30**	0.87
Ours	0.75	1.15	**0.93**	0.63	1.42	**1.99**	1.64	**0.81**

Table 3.2: Tracking performance (ATE RMSE (cm)) on Replica dataset.



(a) office0 (b) office1 (c) office2 (d) office3

(e) office4 (f) room0 (g) room1 (h) room2

Figure 3.3: Qualitative tracking results on Replica dataset.

complex scenes with numerous semantic classes, and the size of the MLP is also unsuitable for this task. The complexity and capacity of an MLP to model intricate relationships in data are limited by the number of layers and neurons per layer. In scenarios involving complex scenes with many semantic classes, a single MLP may not have enough capacity to capture the high variability and intricate relationships among different semantic categories. Additionally, a single MLP model tasked with handling all classes might generalize too much and fail to specialize in the characteristics of specific semantic classes. This generalization can lead to poorer performance in recognizing and differentiating between classes that have subtle distinctions. Multiple MLPs or a more complex neural network architecture might be necessary to scale effectively with the increasing complexity and to manage the interactions between different semantic features more efficiently. Second, there is no interaction between the semantic information and RGB-D data. Since I use independent MLPs to process semantic labels and RGB-D data, Without interaction between these networks, the overall scene understanding could be less accurate, as the system may not correlate these insights effectively. Separate processing might lead to situations where both MLPs are independently attempting to solve similar problems, such as identifying and classifying objects, without sharing insights. This could result in inefficiencies and potentially conflicting interpretations of the scene data. To improve the performance, considering methods to facilitate some level of interaction or fusion between the MLPs

handling different data types could be crucial. Third, I employ a simple summation to compute the semantic logits for each pixel. Simply summing the logits assumes that each point along the ray contributes equally to the final semantic interpretation of the pixel. However, different points along the ray might have varying degrees of importance based on their position, occlusion status, and other factors. For example, points closer to the camera or those representing surfaces facing the camera might be more indicative of the true semantic nature of the pixel than points that are farther away or occluded. However, the significance of each sample point along the rays varies; the model should prioritize more important points along each ray. Additionally, different semantic classes may overlap along the depth of a ray due to occlusions or perspective effects. Summing the logits can blur these distinctions, leading to a mixed signal that does not accurately reflect the most visible or relevant semantic class from the viewer's perspective. To address these issues, more sophisticated methods for aggregating semantic information along a ray could be considered. Techniques such as weighted averaging where weights are assigned based on the depth, clarity, or confidence of semantic predictions at each point, or using a max-pooling approach to select the most probable semantic class along the ray, could provide more accurate and meaningful results. Finally, the weighting of each loss function must be carefully adjusted. Properly weighted loss functions can help the model generalize better to unseen data by ensuring that it does not overfit to particular aspects of the training data that are overemphasized by disproportionately weighted losses. To effectively learn the geometric, photometric, and semantic information, we need to wisely assign the weights for each loss. If not, the semantic loss could detrimentally affect the overall optimization process.

CHAPTER 4

CONCLUSION

In conclusion, this thesis presents a comprehensive solution to enhance the generalization capabilities of neural SLAM systems by incorporating Lipschitz continuity and semantic information. The novel semantic-informed RGB-D neural SLAM system proposed in this work leverages the strengths of both Lipschitz regularization and semantic integration to achieve significant improvements in the overall performance and robustness of neural SLAM systems.

The primary contributions of this research include the development of weight normalization and Lipschitz regularization techniques, which significantly improve the robustness and generalization of MLPs. These techniques enable the neural networks to better handle noisy environments and maintain stable performance. Additionally, the thesis introduces a method that leverages semantic information to guide the training process. My research on the proposed semantic-informed SLAM elucidates why it does not surpass Co-SLAM. Key issues include the use of a single MLP, which is insufficient for complex scenes with numerous semantic classes. The lack of interaction between semantic information and RGB-D data hampers accurate scene understanding. Aggregating semantic logits by simple summation overlooks variations in point importance along a ray, potentially missing critical semantic details. Advanced aggregation methods like weighted averaging or max-pooling could yield more precise results.

Overall, the research presented in this thesis provides valuable insights and advancements in the field of neural SLAM. It opens up new possibilities for future work, including further exploration of Lipschitz regularization and integration of additional semantic information.

BIBLIOGRAPHY

[1] Alexandre Araujo, Aaron Havens, Blaise Delattre, Alexandre Allauzen, and Bin Hu. A unified algebraic perspective on lipschitz neural networks, 2023.

[2] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction, 2022.

[3] Yimeng Chen, Tianyang Hu, Fengwei Zhou, Zhenguo Li, and Zhiming Ma. Explore and exploit the diverse knowledge in model zoo for domain generalization, 2023.

[4] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks, 2023.

[5] Mahyar Fazlyab, Taha Entesari, Aniket Roy, and Rama Chellappa. Certified robustness via dynamic margin maximization and improved lipschitz regularization, 2024.

[6] Wenzhi Guo, Bing Wang, and Lijun Chen. Neuv-slam: Fast neural multiresolution voxel optimization for rgbd dense slam, 2024.

[7] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[8] Yujia Huang, Huan Zhang, Yuanyuan Shi, J Zico Kolter, and Anima Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds, 2021.

[9] Seth Isaacson, Pou-Chun Kung, Mani Ramanagopal, Ram Vasudevan, and Katherine A. Skinner. Loner: Lidar only neural representations for real-time slam, 2024.

[10] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields, 2023.

[11] Lukas Koestler, Nan Yang, Niclas Zeller, and Daniel Cremers. Tandem: Tracking and dense mapping in real-time using deep multi-view stereo, 2021.

[12] Klas Leino, Zifan Wang, and Matt Fredrikson. Globally-robust neural networks, 2021.

[13] Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. Learning smooth neural functions via lipschitz regularization, 2022.

[14] Max Losch, David Stutz, Bernt Schiele, and Mario Fritz. Certified robust models with slack control and large lipschitz constants, 2023.

[15] Raul Mur-Artal and Juan D. Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, October 2017. ISSN 1941-0468. doi: 10.1109/tro.2017.2705103. URL http://dx.doi.org/10.1109/TRO.2017.2705103.

[16] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6), nov 2013. ISSN 0730-0301. doi: 10.1145/2508363.2508374. URL https://doi.org/10.1145/2508363.2508374.

[17] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception, 2022.

[18] Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgower. Training robust neural networks using lipschitz bounds. *IEEE Control Systems Letters*, 6:121–126, 2022. ISSN 2475-1456. doi: 10.1109/lcsys.2021.3050444. URL http://dx.doi.org/10.1109/LCSYS.2021.3050444.

[19] Chris Rohlfs. Generalization in neural networks: A broad survey, 2023.

[20] Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-slam: Dense neural point cloud-based slam, 2023.

[21] Zhouxing Shi, Yihan Wang, Huan Zhang, Zico Kolter, and Cho-Jui Hsieh. Efficiently computing local lipschitz constants of neural networks via bound propagation, 2022.

[22] Sahil Singla, Surbhi Singla, and Soheil Feizi. Improved deterministic l2 robustness on cifar-10 and cifar-100, 2022.

[23] Jamie Stirling and Noura Al Moubayed. Addressing performance inconsistency in domain generalization for image classification. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08, 2023. doi: 10.1109/IJCNN54540.2023.10191685.

[24] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The replica dataset: A digital replica of indoor spaces, 2019.

[25] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. imap: Implicit mapping and positioning in real-time, 2021.

[26] Emanuele Vespa, Nikolay Nikolov, Marius Grimm, Luigi Nardi, Paul H. J. Kelly, and Stefan Leutenegger. Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151, 2018. doi: 10.1109/LRA.

2018.2792537.

[27] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam, 2023.

[28] Ruigang Wang and Ian R. Manchester. Direct parameterization of lipschitz-bounded deep networks, 2023.

[29] Thomas Whelan, Michael Kaess, Maurice F. Fallon, Hordur Johannsson, John J. Leonard, and John B. McDonald. Kintinuous: Spatially extended kinectfusion. In *AAAI Conference on Artificial Intelligence*, 2012. URL https://api.semanticscholar.org/CorpusID:15010509.

[30] Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. Voxurf: Voxel-based efficient and accurate neural surface reconstruction, 2023.

[31] Bohang Zhang, Tianle Cai, Zhou Lu, Di He, and Liwei Wang. Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons, 2021.

[32] Bohang Zhang, Du Jiang, Di He, and Liwei Wang. Boosting the certified robustness of l-infinity distance nets. *ArXiv*, abs/2110.06850, 2021. URL https://api.semanticscholar.org/CorpusID:238744365.

[33] Bohang Zhang, Du Jiang, Di He, and Liwei Wang. Rethinking lipschitz neural networks and certified robustness: A boolean function perspective, 2022.

[34] Wei Zhang, Tiecheng Sun, Sen Wang, Qing Cheng, and Norbert Haala. Hi-slam: Monocular real-time dense mapping with hybrid implicit fields, 2023.

[35] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction, 2023.

[36] Siting Zhu, Guangming Wang, Hermann Blum, Jiuming Liu, Liang Song, Marc Pollefeys, and Hesheng Wang. Sni-slam: Semantic neural implicit slam, 2024.

[37] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam, 2022.

[38] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R. Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam, 2023.