Real Time 3D Mapping and Localization of Ground Vehicles

A Project Report Submitted by

Pallab Saha

in partial fulfillment of the requirements for the award of the degree of

M.Tech

Indian Institute of Technology Jodhpur

IDRP-ROBOTICS AND MOBILITY SYSTEMS

September, 2024

Declaration

I hereby declare that the work presented in this Project Report titled Real Time 3D Mapping and Localization of Ground Vehicles submitted to the Indian Institute of Technology Jodhpur in partial fulfilment of the requirements for the award of the degree of M.Tech, is a bonafide record of the research work carried out under the supervision of Dr. Rajendra Nagar. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Pallab Saha

Signature

Pallab Saha

M22RM218

Certificate

This is to certify that the Project Report titled Title of the Project Report, submitted by Pallab Saha(M22RM218) to the Indian Institute of Technology Jodhpur for the award of the degree of M.Tech, is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Rajendra Nagar

Signature

Dr. Rajendra Nagar

Acknowledgements

I would like to express my heartfelt gratitude to Professor Dr. Rajendra Nagar and Mr. Sudhir Pratap Yadav for their invaluable support and guidance throughout this research. Professor Dr. Rajendra Nagar's profound expertise and insightful feedback have significantly shaped this work, while Mr.Sudhir Pratap Yadav's practical advice and encouragement have been instrumental in overcoming numerous challenges. Their dedication and commitment have greatly contributed to the successful completion of this project, and for that, I am deeply thankful.

Abstract

In this work, we present significant enhancements to the Co-SLAM algorithm, specifically tailored for real-time 3D mapping and localization of ground vehicles, including applications in domestic robots, self-driving cars, rapid response vehicles, and industrial robots. By introducing modifications to the decoder through optimized MLP (Multilayer Perceptron) configurations, we have achieved substantial improvements in timing efficiency.

These enhancements lead to better accuracy, completion, completion ratio, and reduced time complexity, enabling more precise and faster object detection. This advancement facilitates superior depth perception, terrain mapping, and obstacle detection, thereby enhancing navigation and operational safety in autonomous systems. Our work advances the capabilities of real-time 3D SLAM, contributing significantly to the efficiency and effectiveness of autonomous vehicle operations in diverse environments.

Contents

List of Figures

List of Tables

Real Time 3D Mapping and Localization of Ground Vehicles

1 Introduction

Real-time 3D mapping and localization are crucial for the effective operation of autonomous ground vehicles. These systems enable vehicles to navigate complex environments with high precision, enhancing safety and efficiency in applications such as autonomous driving and robotics. The ability to continuously and accurately map and localize in real time is fundamental for the reliable operation of autonomous systems.

Recent advancements in Simultaneous Localization and Mapping (SLAM) have significantly contributed to this field. Traditional dense visual SLAM systems have demonstrated the capability to operate in real-time and handle large-scale scenes. However, these methods often struggle with generating accurate geometry for unobserved regions and may be less effective in handling noisy or incomplete data [1, 2].

Learning-based SLAM approaches have introduced neural network architectures that leverage data-driven priors to improve robustness and accuracy. Methods such as Neural Radiance Fields (NeRF) offer promising results by predicting scene properties from images, yet they often require extensive training and may suffer from issues such as oversmoothing and limited real-time performance [3, 4, 5].

Recent innovations like NICE-SLAM have attempted to address these limitations by incorporating hierarchical feature grids and neural implicit representations to achieve real-time performance and scalability [6]. Similarly, Co-SLAM has introduced a hybrid approach combining coordinate-based and parametric embeddings to enhance scene completion and tracking accuracy [7, 8]. However, both approaches still face challenges related to completeness, robustness, and computational efficiency.

In this research, we propose a novel methodology that builds upon the Co-SLAM framework and introduces enhancements to the decoder component. The new approach aims to improve depth accuracy, increase the completion ratio of the maps, and optimize processing times while maintaining real-time performance. By addressing the limitations observed in existing methods, our methodology seeks to offer significant improvements in the efficiency and reliability of 3D mapping and localization systems for autonomous ground vehicles.

Our methodology incorporates advanced neural network architectures, including Multi-Layer Perceptrons (MLPs), and modifications to the decoder to overcome the limitations of traditional and learning-based SLAM systems. The goal is to achieve enhanced performance metrics, including better depth accuracy, higher completion ratios, and reduced job duration, ultimately advancing the state of real-time 3D mapping and localization.

In summary, this research aims to bridge the gap between existing methods and real-time requirements, contributing to the development of more accurate, efficient, and robust SLAM systems for autonomous vehicles.

2 Literature Survey

2.1 Dense Visual SLAM

Dense Visual SLAM techniques have significantly evolved over the years. Early methods, such as KinectFusion [9], utilized the Truncated Signed Distance Function (TSDF) for scene updates and the Iterative Closest Point (ICP) algorithm for tracking. These approaches allowed for real-time 3D reconstruction using depth sensors, enabling interactive applications. However, they faced several challenges, including scalability and efficiency, particularly when dealing with large environments or requiring high-resolution models.

KinectFusion was pioneering in its ability to create dense 3D models from depth camera data in real-time. It employed a volumetric representation of the scene using TSDFs, which allowed for smooth and continuous surface reconstruction. The ICP algorithm was used to align consecutive frames, ensuring that the reconstructed model remained accurate over time. Despite these advancements, KinectFusion struggled with memory usage and computational requirements, limiting its applicability in larger or more complex environments.

To address these limitations, subsequent works introduced more efficient data structures. Surfels [10], or surface elements, provided a way to represent surfaces using small disks. This method efficiently handled varying levels of detail and was more memory-efficient than volumetric representations. Surfels allowed for scalable reconstructions, particularly in environments with large, flat surfaces, and facilitated real-time updates.

VoxelHashing [11] further improved efficiency by using a hash table to store only the occupied voxels of the scene, significantly reducing memory consumption. This method enabled large-scale dense 3D mapping in real-time, making it feasible to reconstruct extensive environments without prohibitive memory requirements. VoxelHashing maintained high accuracy in tracking and reconstruction by dynamically allocating and deallocating memory as needed. Its ability to handle large datasets in real-time made it a significant advancement over previous methods.

Octrees [12] provided another leap in efficient scene representation. An octree is a hierarchical data structure that recursively subdivides the 3D space into smaller octants, allowing for efficient storage and retrieval of spatial information. Octree-based methods like OctoMap enabled efficient probabilistic 3D mapping, particularly useful for applications requiring large-scale environment mapping and autonomous navigation. Octrees could adaptively refine the resolution of the map, maintaining detail where needed and saving memory where less detail was sufficient.

Another notable advancement in dense visual SLAM was ElasticFusion [13], which focused on real-time surface reconstruction and dense tracking. ElasticFusion employed a surfel-based representation and used a non-rigid deformation graph to allow for continuous alignment and map updates, accommodating dynamic changes in the environment. This method improved upon the rigidity of earlier systems and provided more accurate and flexible mapping capabilities.

BAD-SLAM [14] represented a significant leap in accuracy and robustness by integrating full direct bundle adjustment (BA). Traditional SLAM approaches often separated the processes of pose estimation and map updating, but BAD-SLAM combined these steps into a unified optimization framework. This full direct BA approach allowed for joint optimization of both pose and structure, leading to improved accuracy in the resulting maps and more robust tracking in challenging environments.

These advancements in dense visual SLAM have paved the way for more robust and scalable systems. However, challenges remain, particularly in dynamic environments where changes in the scene need to be accounted for in real-time. The computational demands of these systems continue to be a limiting factor, necessitating further research into more efficient algorithms and hardware acceleration techniques.

In recent years, deep learning approaches have begun to influence dense visual SLAM, offering potential solutions to some of these challenges. For instance, DeepFactors [15] combines classical SLAM methods with learned features to improve robustness and accuracy. Similarly, methods leveraging convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been proposed to enhance the tracking and mapping capabilities of SLAM systems.

As the field continues to evolve, the integration of machine learning techniques with traditional SLAM algorithms is expected to yield further improvements in efficiency, scalability, and accuracy, addressing the remaining challenges and expanding the applicability of dense visual SLAM systems in real-world scenarios.

2.1.1 Neural Implicit Representations

Neural implicit representations have emerged as a powerful approach for 3D scene modeling and reconstruction. These methods represent scenes using continuous functions parameterized by neural networks, rather than discrete voxel grids or meshes. This continuous representation allows for high-resolution and detailed modeling of complex scenes.

One of the foundational works in neural implicit representations is NeRF (Neural Radiance Fields) [16]. NeRF uses coordinate-based encoding to map 3D coordinates to RGB color and density values through a Multi-Layer Perceptron (MLP). By optimizing these parameters to minimize the difference between rendered and observed images, NeRF achieves high-quality scene reconstruction with detailed textures and lighting effects. However, NeRF's training process is computationally intensive and requires a large number of images to achieve high fidelity, which can be a limitation for real-time applications.

To address the challenges associated with training and memory usage, researchers have developed various improvements and alternative approaches. Sparse Neural Representations, such as those used in the work by Mildenhall et al. [17], utilize sparse voxel grids or octrees combined with neural networks to reduce computational complexity and memory usage while maintaining high-quality reconstruction. These approaches leverage the sparsity of the scene to limit the amount of data that needs to be processed, making them more efficient.

Another notable advancement is the use of multi-scale representations. Techniques such as the Hierarchical Neural Representation (HNR) [18] use a hierarchy of neural networks to capture details at multiple levels of granularity. This approach enables the efficient modeling of both large-scale structures and fine details, improving the overall quality of the scene representation while managing memory more effectively.

Recent developments have also explored integrating neural implicit representations with other forms of data, such as depth maps or lidar scans. For example, Liu et al. [19] incorporate depth information into neural implicit models to enhance reconstruction accuracy, particularly in challenging environments where visual information alone may be insufficient. This integration helps in creating more robust models that can handle occlusions and varying lighting conditions.

Generative models, such as GANs (Generative Adversarial Networks) and VAEs (Variational Autoencoders), have also been employed to improve neural implicit representations. These models can generate realistic textures and details by learning from large datasets of 3D scenes. They can also be used to synthesize new views and complete partially observed scenes, providing more versatile and detailed representations.

The robustness of neural implicit representations to various challenges, such as dynamic scenes and occlusions, has been a significant area of research. Techniques like temporal consistency and scene flow integration have been explored to address these issues. For instance, methods that incorporate temporal coherence [20] can handle dynamic environments by maintaining consistency across multiple frames, which is crucial for applications in robotics and autonomous driving.

Additionally, advancements in hardware and computational techniques have facilitated the scalability of neural implicit representations. The use of GPUs and distributed computing has made it possible to process large-scale scenes and perform real-time inference, expanding the applicability of these methods to more practical scenarios.

In summary, neural implicit representations represent a significant advancement in 3D scene modeling. They offer high-quality reconstructions and detailed representations but come with challenges related to computational demands and memory usage. Ongoing research continues to address these challenges by exploring efficient architectures, integrating additional data sources, and leveraging advanced computational resources. These developments hold promise for further enhancing the capabilities and applicability of neural implicit representations in various domains.

2.1.2 Neural Implicit SLAM

Neural implicit SLAM systems represent a significant evolution in the field of simultaneous localization and mapping by integrating neural network-based representations for both tracking and mapping tasks. These systems leverage the power of neural implicit representations to model 3D scenes as continuous functions, which allows for more detailed and flexible scene reconstructions compared to traditional discrete representations.

One of the pioneering works in this area is iMAP [21], which introduces a novel approach to integrate neural network representations within the SLAM framework. iMAP uses Multi-Layer Perceptrons (MLPs) to represent the geometry and appearance of the scene. The key innovation of iMAP lies in its ability to optimize these neural networks in near-real-time, enabling it to adapt to new observations dynamically as the camera moves through the environment.

iMAP's pipeline begins with the collection of RGB-D images from a moving camera. These images are used to optimize the parameters of the MLPs, which encode the 3D structure and appearance of the scene. The optimization process involves minimizing the photometric error between the observed images and the images rendered from the current neural representation of the scene. This approach allows iMAP to maintain a detailed and coherent model of the environment, even as new data is acquired.

Despite its innovative approach, iMAP faces several challenges. One of the primary limitations is its capacity to handle large and complex scenes. The use of MLPs for scene representation, while powerful, can struggle with scaling to high-resolution details across extensive environments. This capacity limitation can result in less detailed scene modeling and reduced accuracy in tracking, particularly in areas with intricate structures or textures.

To address these limitations, researchers have explored various enhancements and alternative approaches. For instance, recent advancements have proposed the use of hierarchical neural representations and multi-scale encoding techniques to improve the scalability and detail of the neural implicit models. By dividing the scene representation into multiple levels of detail, these approaches can more efficiently manage memory and computational resources, allowing for more detailed and accurate reconstructions.

Another area of improvement involves the integration of additional sensory data, such as lidar scans or stereo camera inputs. By combining visual and depth information, neural implicit SLAM systems can achieve more robust and accurate scene representations, particularly in environments with challenging lighting conditions or occlusions. These multi-modal approaches enhance the system's ability to handle a wider range of scenarios and improve the overall reliability of the SLAM process.

In summary, neural implicit SLAM represents a promising direction for advancing the state of the art in simultaneous localization and mapping. By leveraging the power of neural networks to represent 3D scenes as continuous functions, these systems offer significant improvements in detail and flexibility. Ongoing research continues to address the challenges related to scalability and computational demands, paving the way for more robust and practical applications in real-world environments.

3 Problem Definition

Real-time 3D mapping and localization are critical for the effective operation of autonomous ground vehicles. Despite advancements in algorithms such as Co-SLAM [7] and NICE-SLAM [6], several challenges remain:

- **Insufficient Object Detection:** Current algorithms, including Co-SLAM and NICE-SLAM, often face difficulties in detecting and classifying objects accurately. This issue is exacerbated by challenges in edge detection and color representation in RGB-D inputs, which impact the overall effectiveness of object recognition. Improving edge detection will enable more precise delineation of object boundaries, even in complex scenes with overlapping or partially obscured objects. Enhanced color accuracy will help distinguish between objects with similar hues but different contexts, reducing the likelihood of misclassification.

- **Challenges with Color and Lighting:** Variations in color and lighting conditions within RGB-D images can affect the algorithm's ability to accurately interpret the environment. The existing methods may struggle to handle diverse lighting scenarios, leading to degraded performance in real-world applications. Enhancing the algorithm's robustness to these variations will ensure consistent performance across different environments, such as transitioning from brightly lit outdoor areas to dimly lit indoor spaces. Techniques for color constancy are necessary to maintain accurate color perception despite changes in lighting.

- **Processing Delays:** The real-time processing requirements for autonomous vehicles necessitate rapid data handling and decision-making. However, current systems can experience processing delays, impacting the vehicle's responsiveness and overall operational efficiency. Optimizing computational efficiency and streamlining data processing pipelines are essential to handle large volumes of sensor data swiftly, enabling quicker responses to dynamic changes in the environment.

- **Real-World Complexities:** The algorithms need to address the complexities of real-world environments, such as varying object appearances and unpredictable environmental conditions, which are often inadequately handled by existing solutions. Adaptability to real-world challenges, including sensor noise, calibration issues, and environmental variability, is crucial for maintaining high levels of accuracy and reliability in different scenarios.

4 Objective

This project aims to enhance the Co-SLAM algorithm by improving the Multi-Layer Perceptron (MLP) component to address the challenges identified and achieve the following objectives:

- **Enhanced Object Detection:** Refine the MLP to improve edge detection and color accuracy in RGB-D inputs, leading to more precise object recognition and scene understanding. By achieving more precise object recognition, the autonomous vehicle will be better equipped to navigate complex environments, avoid obstacles, and interact with its surroundings safely and efficiently.

- **Improved Handling of Color and Lighting:** Optimize the MLP to better process and interpret varying lighting conditions and color information, addressing limitations observed in existing methods. Enhancing the algorithm's ability to maintain color constancy despite changes in lighting will ensure consistent object detection and recognition, leading to more reliable performance in diverse environments.

- **Faster Processing Times:** Enhance the MLP to reduce processing delays, enabling quicker responses to dynamic changes and improving real-time operational efficiency. By optimizing the computational efficiency, the system will handle high-speed data input and produce timely outputs, resulting in more responsive and agile autonomous vehicles.

- **Better Real-World Performance:** Ensure that the enhanced algorithm performs robustly across diverse real-world conditions, providing a more effective solution for 3D mapping and localization. This includes developing techniques for handling sensor noise, maintaining calibration, and adapting to environmental variability, ensuring that the algorithm maintains high levels of accuracy and reliability in different scenarios.

By achieving these objectives, the project aims to significantly improve the performance of real-time 3D mapping and localization systems for ground vehicles. This will ultimately enhance the safety, efficiency, and adaptability of autonomous vehicles in complex and dynamic environments, making them more capable of handling real-world challenges and improving their overall operational effectiveness.

5 Methodology

5.1 Problem Definition

The primary objective of this project is to enhance the efficiency, accuracy, completion ratio, and duration of 3D localization and mapping for ground vehicles. This is achieved by improving the Co-SLAM algorithm through advanced neural network architectures and custom activation functions. These enhancements aim to address the limitations of current methods, such as high computational costs and limited precision, particularly in complex environments.

5.2 Data Preparation

The data preparation phase involves several critical steps to ensure the robustness and quality of the dataset used for training and evaluation:

- **Data Collection:** Gather diverse datasets that include real-world and synthetic data, ensuring a wide range of environmental conditions and scenarios for 3D localization and mapping tasks.
- **Data Preprocessing:** Clean and preprocess the collected datasets. This involves noise reduction, normalization, and augmentation techniques to improve the robustness of the training process.
- **Feature Extraction:** Generate embeddings for positional, geometric, and color features from the collected data. This step includes computing positional encodings, extracting geometric shapes, and capturing color information to create comprehensive input feature vectors.

5.3 Model Design

The design phase focuses on creating advanced neural network models incorporating custom activation functions for enhanced performance. The models are specifically designed to address the challenges of 3D localization and mapping by capturing intricate details and improving predictive accuracy.

5.4 Activation Functions

5.4.1 Swish Activation Function

Swish activation function could be defined as:

$$\text{Swish}(x) = x \cdot \sigma(x)$$

where $\sigma(x)$ is sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Characteristics:

- **Smooth and Non-Monotonic:** Swish is a smooth, non-monotonic function. It transitions smoothly through zero, which helps gradients flow better.
- **Range:** The output can be positive or negative, unlike ReLU which is non-negative.

Advantages:

- **Avoids Dead Neurons:** Swish does not suffer from the "dead neuron" problem as it allows for negative outputs.
- **Improved Performance:** Swish has made visible to improve performance in deep networks compared to ReLU.

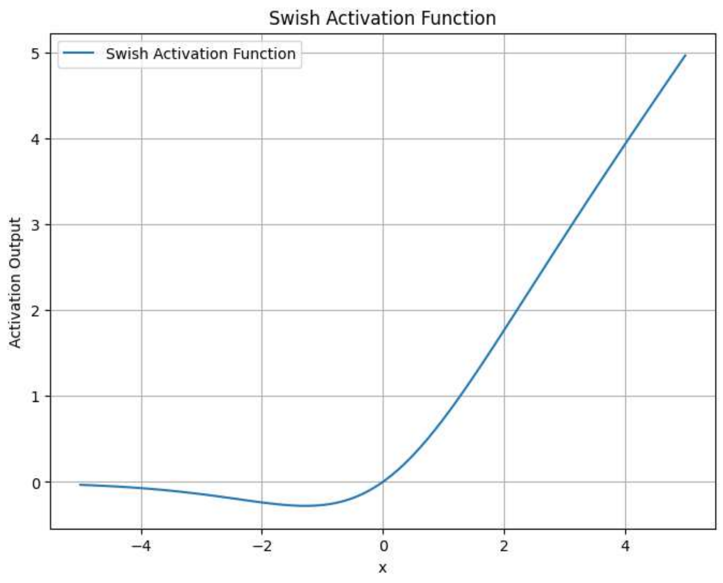

Figure 5.1: Graph of the Swish activation function.

5.4.2 Mish Activation Function

The Mish activation function is defined as:

$$\text{Mish}(x) = x \cdot \tanh(\ln(1 + e^x))$$

where tanh is the hyperbolic tangent function and ln is the natural logarithm.

Characteristics:

- **Smooth and Non-Monotonic:** Mish is smooth and non-monotonic, providing a pronounced curve due to tanh and ln.
- **Range:** Mish allows for negative outputs, providing a smooth transition through zero.

Advantages:

- **Better Gradient Flow:** Mish provides better gradient flow than Swish due to its smooth and non-monotonic nature.
- **Enhanced Performance:** Mish can lead to better performance, especially in very deep networks or complex datasets.

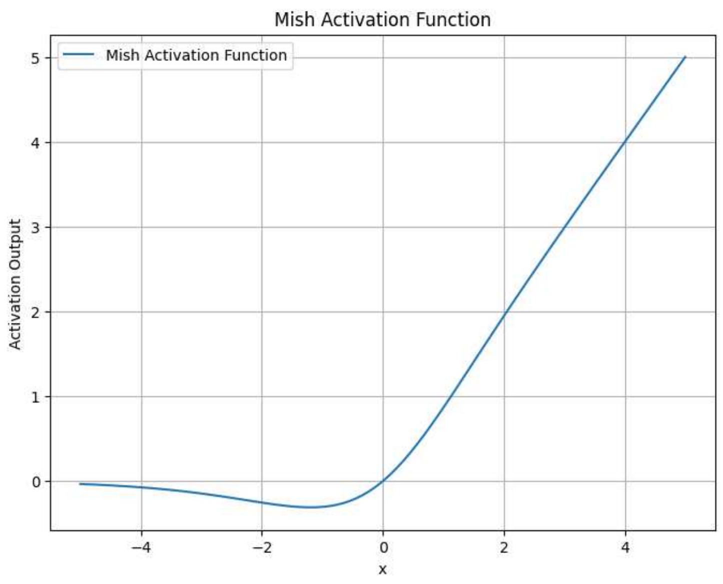

Figure 5.2: Graph of the Mish activation function.

5.4.3 ReLU (Rectified Linear Unit) Activation Function

The ReLU activation function is defined as:

$$\text{ReLU}(x) = \max(0, x)$$

Characteristics:

- **Piecewise Linear:** ReLU outputs zero for negative inputs and x for non-negative inputs.
- **Range:** The output is always non-negative.

Advantages:

- **Simple and Efficient:** ReLU is simple to compute and efficient, widely used in neural networks.
- **Sparse Activation:** Promotes sparse activation which can lead to more efficient computations.

5.4.4 Sigmoid Activation Function

The Sigmoid activation function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

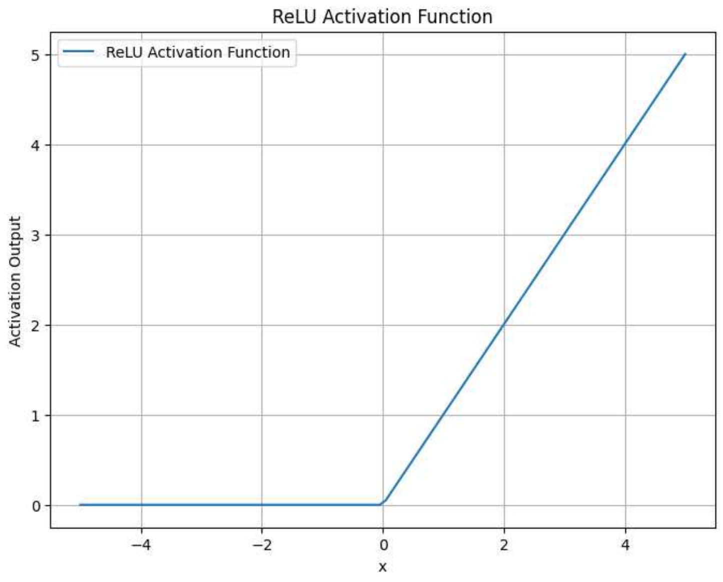

Figure 5.3: Graph of the ReLU activation function.

Characteristics:

- **S-Shaped Curve:** Sigmoid produces an "S"-shaped curve that squashes inputs between 0 and 1.
- **Range:** The output is limited to (0, 1), suitable for binary classification.

Advantages:

- **Probability Output:** Often used in the output layer of binary classification models to represent probabilities.
- **Smooth Gradient:** Provides a smooth gradient, which aids in optimization.

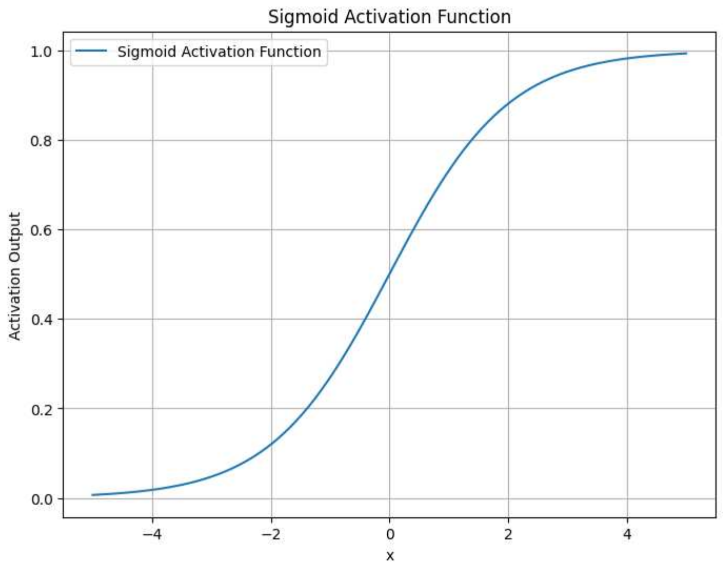

Figure 5.4: Graph of the Sigmoid activation function.

5.4.5 Tanh (Hyperbolic Tangent) Activation Function

The Tanh activation function is defined as:

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Characteristics:

- **Centered Around Zero:** Tanh squashes inputs to the range (-1, 1), centered around zero.
- **Smooth and Non-Monotonic:** Provides a smooth gradient which helps with training deep networks.

Advantages:

- **Centering Data:** Centers the data around zero, which can help with convergence during training.
- **Smooth Gradient:** Provides smooth gradients, improving the training of deep networks.

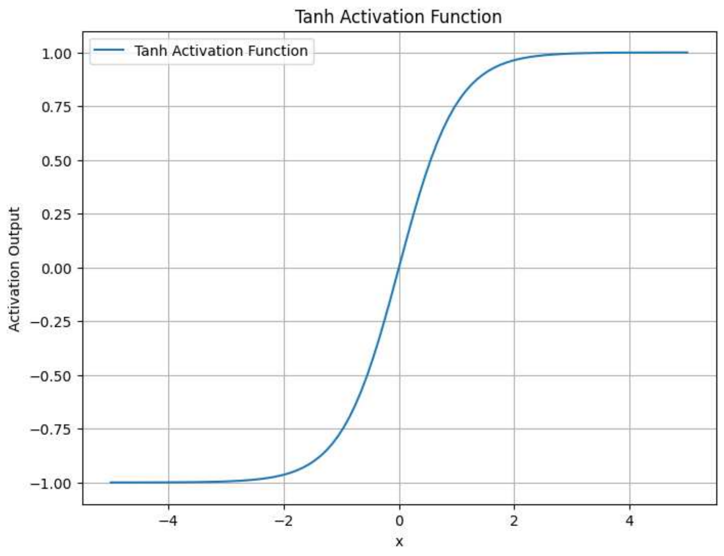

Figure 5.5: Graph of the Tanh activation function.

5.5 Comparison and Benefits

5.5.1 Mish vs ReLU

The Mish activation function and the ReLU (Rectified Linear Unit) activation function are widely used in neural networks due to their distinct properties and advantages. Below is a detailed comparison between these two functions.

Mathematical Formulation

- **ReLU:**

$$\text{ReLU}(x) = \max(0, x)$$

  The ReLU function is simple and computationally efficient. It outputs the input directly if it is positive; otherwise, it outputs zero.

- **Mish:**

$$\text{Mish}(x) = x \cdot \tanh(\ln(1 + e^x))$$

  The Mish function is a non-monotonic activation function that outputs smoother and more continuous gradients compared to ReLU.

Benefits of Mish over ReLU

- **Smoothness:** The Mish function is continuously differentiable, leading to smoother gradient flow during backpropagation. This smoothness can result in better optimization and convergence.
- **Non-monotonicity:** Mish's non-monotonic nature can capture more complex patterns in the data, leading to improved performance in certain deep learning tasks.
- **Gradient Flow:** Unlike ReLU, which can suffer from the "dying ReLU" problem where neurons get stuck during training, Mish maintains a better gradient flow, reducing the risk of neurons becoming inactive.
- **Performance:** Empirical results have shown that Mish can outperform ReLU in various deep learning benchmarks, providing higher accuracy and better generalization.

Visualization Figure 5.6 shows the comparison between the Mish and ReLU activation functions.

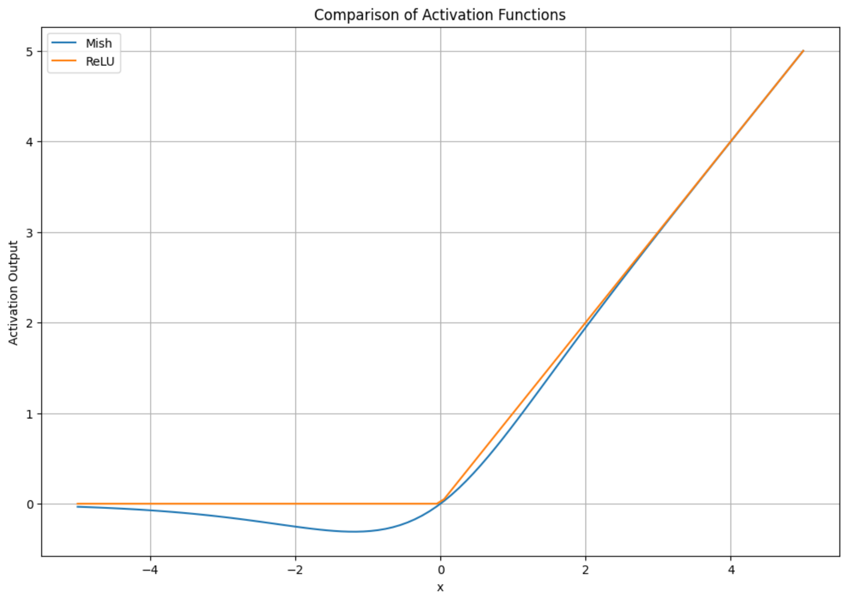

Figure 5.6: Comparison of Mish and ReLU Activation Functions

Use Cases

- **ReLU:** Often used in feedforward neural networks and convolutional neural networks (CNNs) due to its simplicity and computational efficiency.
- **Mish:** Preferred in scenarios where deeper networks and more complex architectures are used, as its smooth gradient flow and non-monotonic properties can lead to better performance and generalization.

5.5.2 Swish vs ReLU

The Swish activation function and the ReLU (Rectified Linear Unit) activation function are both popular choices in neural network design. Each has unique characteristics that make them suitable for different scenarios. Below is a detailed comparison between these two functions.

Mathematical Formulation

- **ReLU:**
$$\text{ReLU}(x) = \max(0, x)$$
The ReLU function is simple and computationally efficient. It outputs the input directly if it is positive; otherwise, it outputs zero.
- **Swish:**
$$\text{Swish}(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}}$$
The Swish function is a smooth, non-monotonic activation function where $\sigma(x)$ is the sigmoid function.

Benefits of Swish over ReLU

- **Smoothness:** Swish is continuously differentiable, leading to smoother gradient flow during backpropagation. This can enhance optimization and convergence.
- **Non-monotonicity:** Swish's non-monotonic nature allows it to capture more complex patterns in the data, which can result in improved performance in some deep learning tasks.
- **Gradient Flow:** Unlike ReLU, which has the "dying ReLU" problem where neurons can become inactive, Swish maintains a better gradient flow, reducing this risk.
- **Performance:** Empirical studies have shown that Swish can outperform ReLU in various deep learning benchmarks, providing higher accuracy and better generalization.

Visualization Figure 5.7 shows the comparison between the Swish and ReLU activation functions.

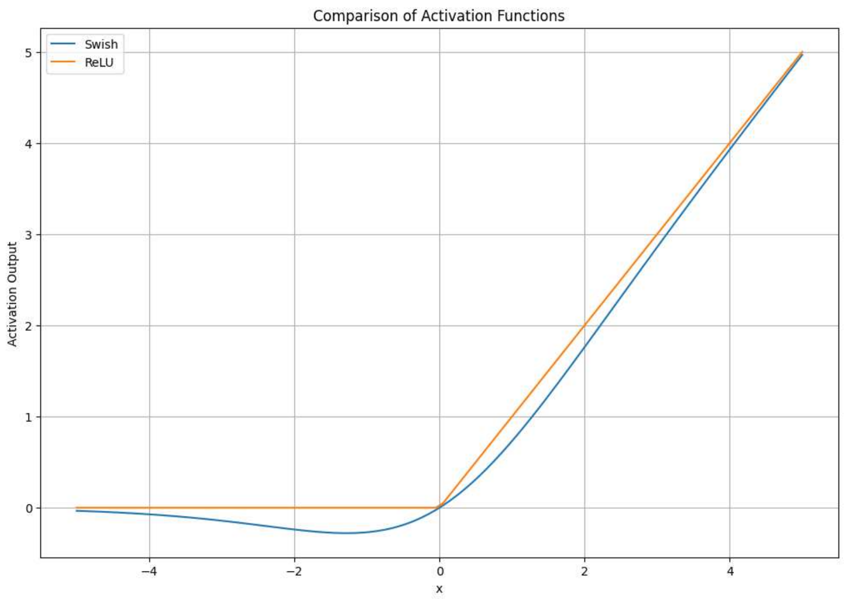

Figure 5.7: Comparison of Swish and ReLU Activation Functions

Use Cases

- **ReLU:** Frequently used in feedforward neural networks and convolutional neural networks (CNNs) due to its simplicity and computational efficiency.
- **Swish:** Preferred in deeper networks and more complex architectures, where its smooth gradient flow and non-monotonic properties can lead to better performance and generalization.

5.5.3 Sigmoid vs ReLU

The Sigmoid and ReLU (Rectified Linear Unit) activation functions are fundamental in neural network design. Each has distinct characteristics that make them useful in different scenarios. Below is a detailed comparison between these two functions.

Mathematical Formulation

- **Sigmoid:**

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

  The Sigmoid function outputs values in the range (0, 1). It is commonly used for binary classification problems as it squashes the input into a range between 0 and 1.

- **ReLU:**

$$\text{ReLU}(x) = \max(0, x)$$

  The ReLU function outputs the input directly if it is positive; otherwise, it outputs zero.

Benefits of Sigmoid over ReLU

- **Output Range:** Sigmoid outputs values in the range (0, 1), making it suitable for tasks where outputs need to be interpreted as probabilities.
- **Smooth Gradient:** The Sigmoid function has a smooth gradient, which can be beneficial for gradient-based optimization algorithms.
- **Non-linearity:** Sigmoid introduces non-linearity, which helps in learning complex patterns.
- **Biological Inspiration:** The Sigmoid function is inspired by the activation potentials in biological neurons, making it intuitive in certain contexts.

Benefits of ReLU over Sigmoid

- **Computational Efficiency:** ReLU is computationally more efficient as it involves only a simple thresholding operation.
- **Mitigates Vanishing Gradient Problem:** ReLU helps mitigate the vanishing gradient problem common with Sigmoid, especially in deep networks.
- **Sparse Activation:** ReLU activation leads to sparsity as it outputs zero for negative inputs, which can result in efficient computation and reduced model complexity.
- **Empirical Performance:** Empirical results have shown that ReLU often leads to faster convergence and better performance in deep learning tasks compared to Sigmoid.

Visualization Figure 5.8 shows the comparison between the Sigmoid and ReLU activation functions.

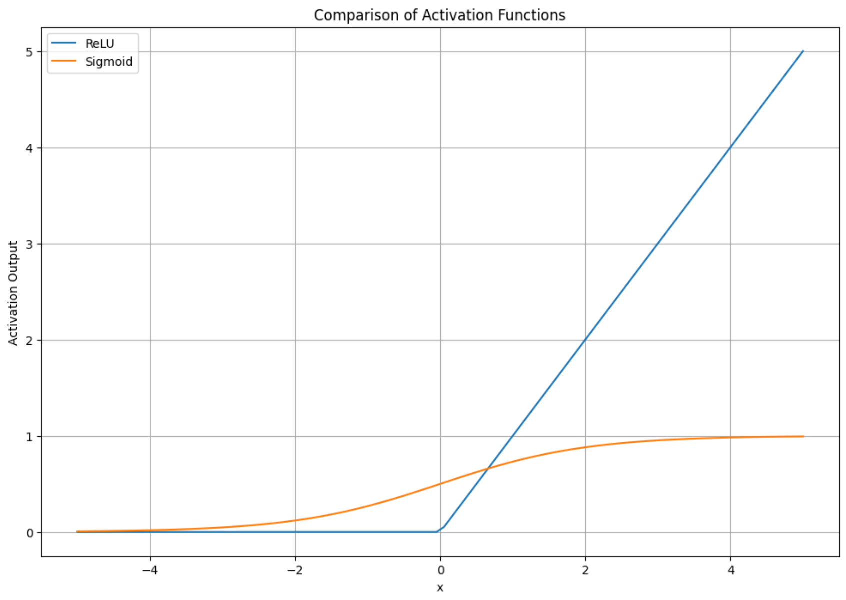

Figure 5.8: Comparison of Sigmoid and ReLU Activation Functions

Use Cases

- **Sigmoid:** Commonly used in the output layer for binary classification problems or for probability-based outputs.
- **ReLU:** Widely used in hidden layers of feedforward and convolutional neural networks due to its simplicity and efficiency.

5.5.4 Network Architectures

The neural network architectures developed include:

- **ColorNet:** A neural network designed to process input feature vectors and produce RGB color values. The architecture can be configured using Tiny CUDA Neural Networks (TCNN) or standard PyTorch layers with custom activations for flexibility and performance. The flexibility in configuration allows for optimization based on the computational resources available and the specific requirements of the task.

  Network Architecture The ColorNet architecture consists of multiple layers, each with specific roles in processing the input features to generate the desired RGB color values. The architecture includes the following layers and activation functions:

  $$\mathbf{h}^{(0)} = \text{Swish}(\mathbf{W}^{(0)}\mathbf{f})$$

  The first layer processes the input feature vector $\mathbf{f}$ using a linear transformation with weight matrix $\mathbf{W}^{(0)}$, followed by the Swish activation function. Swish is defined as:

  $$\text{Swish}(x) = x \cdot \sigma(x)$$

  where $\sigma(x)$ is the sigmoid function:

  $$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  $$\mathbf{h}^{(1)} = \text{SELU}(\mathbf{W}^{(1)}\mathbf{h}^{(0)})$$

  The second layer further processes the output of the first layer using another linear transformation with weight matrix $\mathbf{W}^{(1)}$, followed by the SELU (Scaled Exponential Linear Unit) activation function. SELU is defined as:

  $$\text{SELU}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^{x} - \alpha & \text{if } x \leq 0 \end{cases}$$

  with $\lambda = 1.0507$ and $\alpha = 1.6733$.

$$\mathbf{h}^{(2)} = \text{Mish}(\mathbf{W}^{(2)}\mathbf{h}^{(1)})$$

The third layer applies a linear transformation with weight matrix $\mathbf{W}^{(2)}$ to the output of the second layer, followed by the Mish activation function. Mish is defined as:

$$\text{Mish}(x) = x \cdot \tanh(\ln(1 + e^{x}))$$

$$\mathbf{c} = \mathbf{W}^{(3)}\mathbf{h}^{(2)}$$

Finally, the output layer generates the RGB color values $\mathbf{c}$ using a linear transformation with weight matrix $\mathbf{W}^{(3)}$.

Advantages of Custom Activations

- **Swish:**
  - *Smooth Non-linearity:* The smooth, non-monotonic nature of Swish helps in learning complex patterns and gradients flow more smoothly through the network.
  - *Improved Performance:* Empirical studies have shown that Swish often leads to better performance compared to ReLU and other activation functions in deep networks.
- **SELU:**
  - *Self-Normalizing:* SELU helps in self-normalizing neural networks, which can stabilize training by maintaining mean and variance of activations.
  - *Robust Gradient Flow:* The exponential component in SELU ensures that the gradient does not vanish, thus aiding in effective learning in deeper networks.
- **Mish:**
  - *Smooth and Non-Monotonic:* Similar to Swish, Mish is smooth and non-monotonic, which allows for better gradient flow and learning of complex patterns.
  - *Empirical Superiority:* Mish has been shown to outperform Swish and ReLU in many benchmark tests, providing better accuracy and faster convergence in training deep networks.

Configuration Options

- **Tiny CUDA Neural Networks (TCNN):**
  - *High Performance:* TCNN leverages CUDA for efficient computation, making it suitable for large-scale neural networks and real-time applications.

    - *Flexibility:* The network can be configured to use fully fused MLPs (Multilayer Perceptrons) with custom activation functions, allowing for optimized performance.
  - **Standard PyTorch Layers:**
    - *Ease of Implementation:* Using standard PyTorch layers makes it easier to implement and modify the network.
    - *Custom Activations:* The flexibility to use custom activation functions like Swish, SELU, and Mish allows for fine-tuning the network for specific tasks.

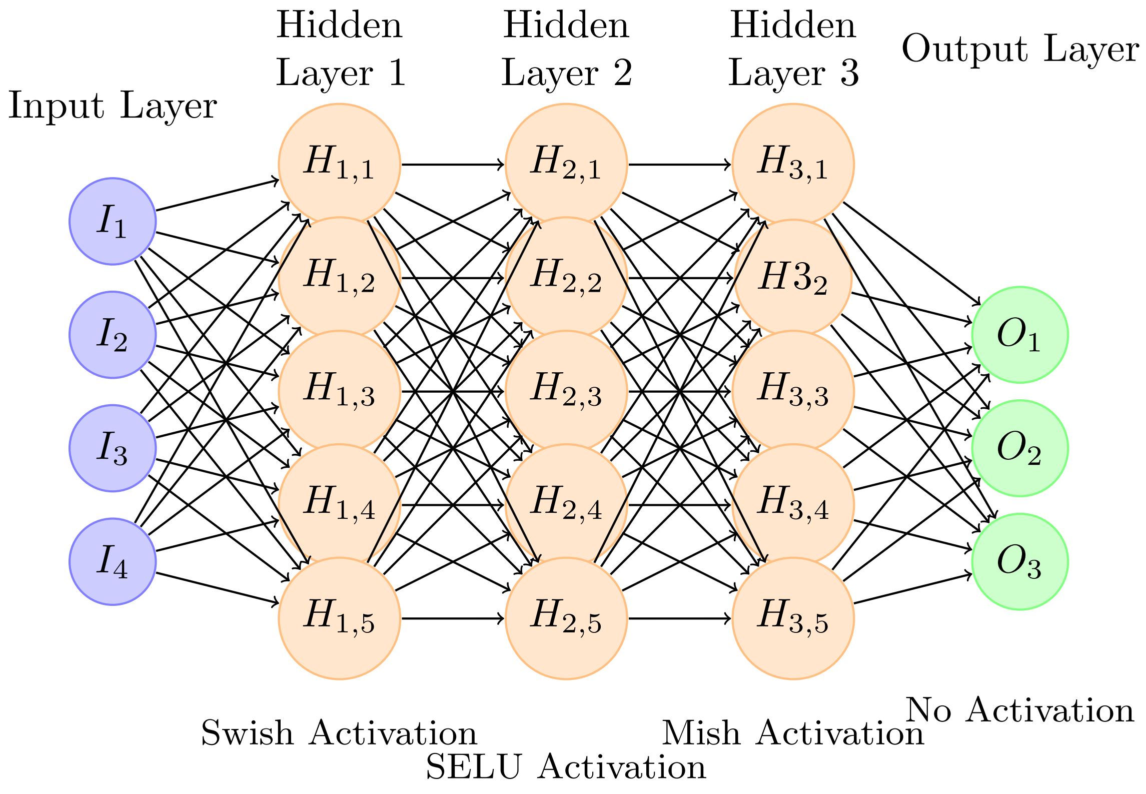

Figure 5.9: Architecture of the ColorNet with three hidden layers and different activations.

Algorithm 1 ColorNet Algorithm

```
Require: config, input_ch = 4, geo_feat_dim = 15, hidden_dim_color = 64, num_layers_color = 3
Ensure: Trained ColorNet model for RGB prediction
 1: Initialization: Set self.config, self.input_ch, self.geo_feat_dim, self.hidden_dim_color, self.num_layers_color
 2: Model Creation:
 3: if config['decoder']['tcnn_network'] then
 4:     Print "Using TCNN"
 5:     model = tcnn.Network(n_input_dims=input_ch + geo_feat_dim, n_output_dims=3, network_config={...})
 6: else
 7:     color_net = [ ]
 8:     for $l = 0$ to num_layers_color - 1 do
 9:         in_dim = input_ch + geo_feat_dim if $l = 0$ else hidden_dim_color
10:         out_dim = 3 if $l =$ num_layers_color - 1 else hidden_dim_color
11:         color_net.append(nn.Linear(in_dim, out_dim, bias=False))
12:         activation = Swish() if $l = 0$ else if $l =$ num_layers_color - 1 do nothing else nn.SELU(inplace=True) if $l \% 2 = 1$ else Mish()
13:         color_net.append(activation)
14:     end for
15:     model = nn.Sequential(*color_net)
16: end if
17: Forward Pass:
18: output = model(input_feat)
19: Output: RGB values as output
```

- **SDFNet:** A neural network designed to process input vectors and produce Signed Distance Function (SDF) values along with geometric features. The SDFNet architecture is aimed at generating SDF values, which represent the distance to the nearest surface, and additional geometric features that provide context about the shape and structure of the objects being modeled. The network is structured as follows:

  Network Architecture The architecture of SDFNet involves multiple layers of transformations and activations, which are detailed below:

$$\mathbf{h}^{(0)} = \text{Swish}(\mathbf{W}^{(0)}\mathbf{g})$$

The first layer applies a linear transformation to the input vector $\mathbf{g}$ using weight matrix $\mathbf{W}^{(0)}$, followed by the Swish activation function. Swish is defined as:

$$\text{Swish}(x) = x \cdot \sigma(x)$$

where $\sigma(x)$ is the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The Swish activation function introduces non-linearity while preserving gradient flow, which can enhance the model's ability to learn complex representations and improve overall performance.

$$\mathbf{h}^{(1)} = \text{SELU}(\mathbf{W}^{(1)}\mathbf{h}^{(0)})$$

In the second layer, another linear transformation is applied to the output of the first layer using weight matrix $\mathbf{W}^{(1)}$, followed by the SELU (Scaled Exponential Linear Unit) activation function. SELU is defined as:

$$\text{SELU}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leq 0 \end{cases}$$

where $\lambda = 1.0507$ and $\alpha = 1.6733$. SELU is designed to self-normalize activations and gradients, which can help stabilize training and improve convergence in deep networks.

$$\mathbf{h}^{(2)} = \text{Mish}(\mathbf{W}^{(2)}\mathbf{h}^{(1)})$$

The third layer further processes the output using a linear transformation with weight matrix $\mathbf{W}^{(2)}$, followed by the Mish activation function. Mish is defined as:

$$\text{Mish}(x) = x \cdot \tanh(\ln(1 + e^x))$$

Mish introduces smooth, non-monotonic behavior, which helps in capturing more complex relationships within the data and improving learning dynamics.

$$\mathbf{sdf}, \mathbf{geo} = \mathbf{W}^{(3)}\mathbf{h}^{(2)}$$

The final layer produces the Signed Distance Function (SDF) values and geometric features by applying a linear transformation using weight matrix $\mathbf{W}^{(3)}$. The network outputs two components:

– **SDF Values (sdf):** These values represent the distance to the nearest surface from a given point, where positive values indicate points outside the surface and negative values indicate points inside.

– **Geometric Features (geo):** These features provide additional context about the shape and structure of the surface, which can be used for various applications such as object recognition or scene reconstruction.

Advantages of SDFNet Architecture

- **Accurate Surface Representation:** The SDF values provide a precise representation of the distance to the nearest surface, which is crucial for tasks such as 3D object modeling and collision detection.
- **Detailed Geometric Features:** By outputting geometric features along with SDF values, the network enhances its ability to capture and utilize detailed information about the object shapes.
- **Effective Non-linearity:** The combination of Swish, SELU, and Mish activation functions ensures smooth gradient flow and helps in learning complex patterns, improving the overall performance of the network.
- **Self-Normalizing Properties:** The SELU activation function contributes to self-normalizing properties, which can lead to more stable training and faster convergence.

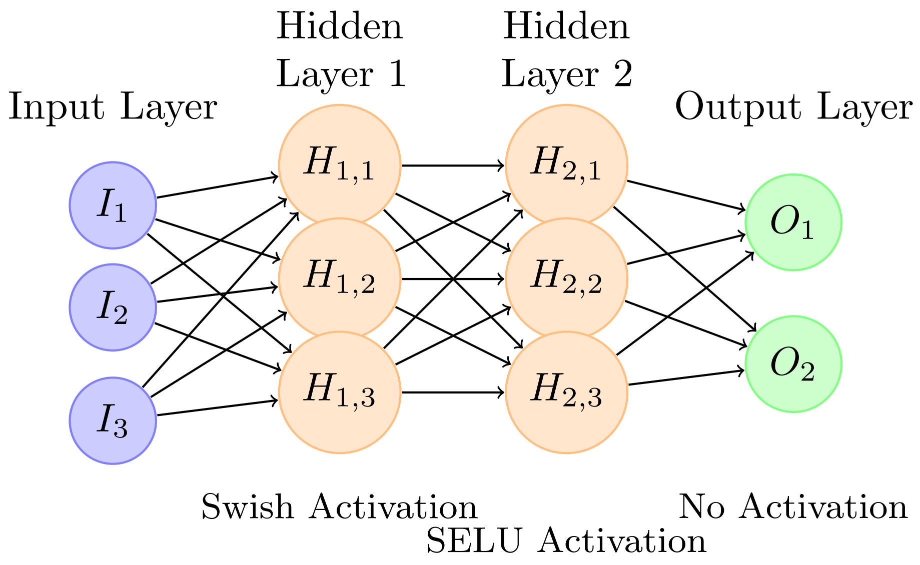

Figure 5.10: Architecture of the ColorNet with three hidden layers and different activations.

Algorithm 2 SDFNet Algorithm

```
Require: config, input_ch = 3, geo_feat_dim = 15, hidden_dim = 32, num_layers = 2
Ensure: Trained SDFNet model
 1: Initialization: Set parameters self.config, self.input_ch, self.geo_feat_dim,
    self.hidden_dim, self.num_layers
 2: Model Creation:
 3: if config['decoder']['tcnn_network'] then
 4:     Print "Using TCNN"
 5:     model = tcnn.Network(n_input_dims=input_ch, n_output_dims=1 + geo_feat_dim,
    network_config={...})
 6: else
 7:     sdf_net = [ ]
 8:     for $l = 0$ to num_layers - 1 do
 9:         in_dim = input_ch if $l = 0$ else hidden_dim
10:         out_dim = 1 + geo_feat_dim if $l =$ num_layers - 1 else hidden_dim
11:         sdf_net.append(nn.Linear(in_dim, out_dim, bias=False))
12:         activation = Swish() if $l = 0$ else if $l =$ num_layers - 1 do nothing else
    nn.SELU(inplace=True) if $l\%2 = 1$ else Mish()
13:         sdf_net.append(activation)
14:     end for
15:     model = nn.Sequential(*sdf_net)
16: end if
17: Forward Pass:
18: output = model(input_feat)
19: Output: Signed Distance Function (SDF) values and geometric features as output
```

6 Our Findings

6.1 Theoretical Findings

- **Activation Functions:**
  - **Sigmoid:** Theoretical analysis suggests that sigmoid functions tend to produce smoother gradient updates, potentially enhancing convergence stability but may lead to slower learning due to vanishing gradients.
  - **ReLU:** ReLU functions are effective in mitigating vanishing gradients but can suffer from dead neurons, affecting learning stability.
  - **SELU and ELU:** These functions offer self-normalizing properties, which help maintain gradient flow and stabilize learning.
  - **Softplus and Swish:** Expected to provide smoother gradient transitions, improving model performance by avoiding issues related to vanishing gradients.
  - **Tanh and LeakyReLU:** Tanh functions provide a range of output values aiding smoother convergence, while LeakyReLU mitigates dead neuron issues by allowing a small gradient when the unit is inactive.
- **Combination Architectures:**
  - Combining different activation functions aims to leverage the strengths of each, potentially enhancing overall model performance and stability.

6.2 Numerical Findings

Type name	**Acc** ↓	**Comp** ↓	**Comp Ratio** ↑ [%]	**Depth** ↓
office3_Sigmoid_sequence	2.7564	2.7327	90.6585	1.6180
office3_original_relu	2.7858	2.7348	90.7670	1.5770
office3_SELU_seq_mod	2.8481	2.5927	91.4385	1.6038
office3_ELU	2.8249	2.6365	91.2775	1.6057
office3_Softplus_module_sigmoid	2.9489	2.6401	91.0690	1.5498
Office3_combination	3.2229	2.6312	91.4815	1.5945
office3_Swish_Selu_mish_4o	2.6582	2.6850	91.0460	1.5716
office3_4o_L_32	2.6810	2.5737	91.6250	1.6074
office3_Combination_1	3.3268	2.6609	91.2940	1.5830
office3_128+3+depth	2.7120	2.7445	90.8785	1.5625
office3_depth_0.65_0.85	2.9242	2.7359	90.6990	1.6228
office3_Tanh_Sequential	2.8135	2.7003	90.9105	1.6365
office3_tanh_ModuleList	2.8879	2.6431	91.0590	1.6361

Type name	Acc ↓	Comp ↓ [cm]	Comp Ratio ↑ [%]	Depth ↓ [cm]
office3_Mish_chatgpt_tanh	2.7458	2.6490	91.2920	1.5581
office3_64+6	2.8192	2.7672	90.6830	1.6359
office3_Softplus	3.0892	2.6367	91.3370	1.5901
office3_SELU_sequential	3.3714	2.5973	91.4430	1.5826
office3_Mish_copilot_tanh	2.6230	2.6310	91.3620	1.6070
office3_LeakyReLU	2.7618	2.7593	90.8130	1.5709
office3_GELU	2.8214	2.6569	91.2465	1.5794
office3_Swish	2.9020	2.6010	91.4660	1.5751
office3_64+10	3.4855	2.6564	91.4565	1.6297
output_512+6	3.0406	2.7175	91.1320	1.6162
office3_256+13	2.9399	2.7393	91.1945	1.7323
office3_512+13	3.5681	2.6745	91.5380	1.6475

Table 6.1: Numerical Findings

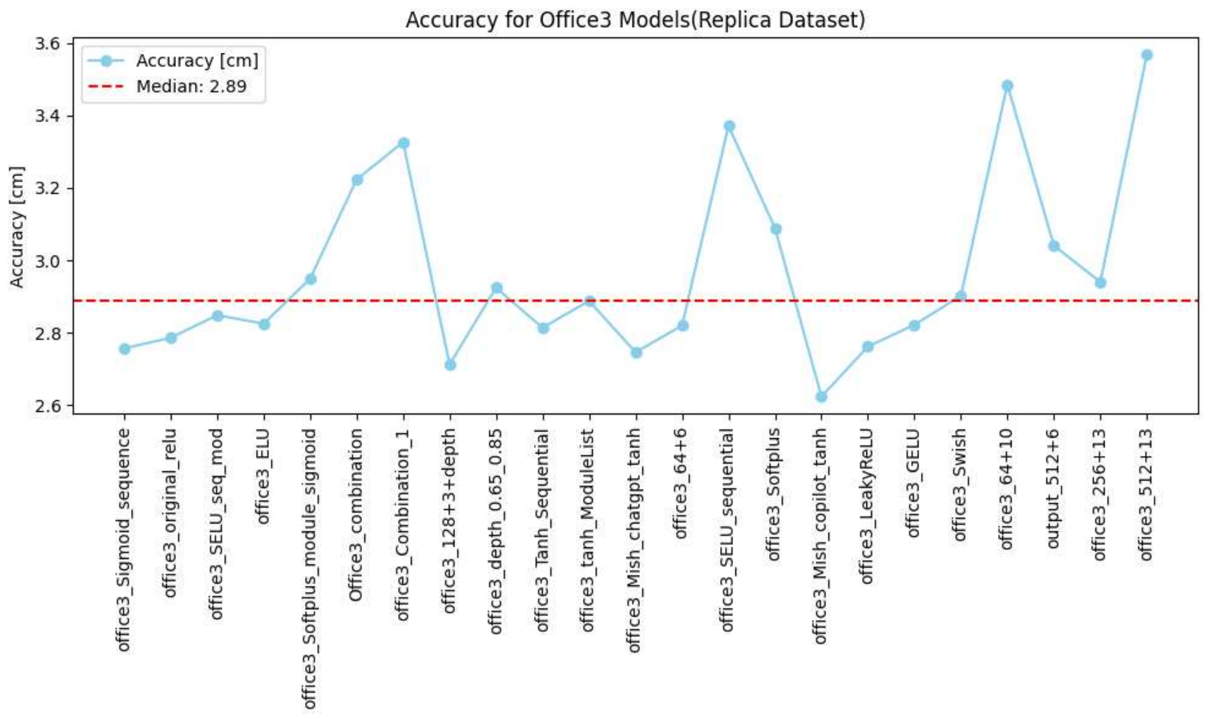

Figure 6.1: Accuracy for Office3 Models(Replica Dataset)

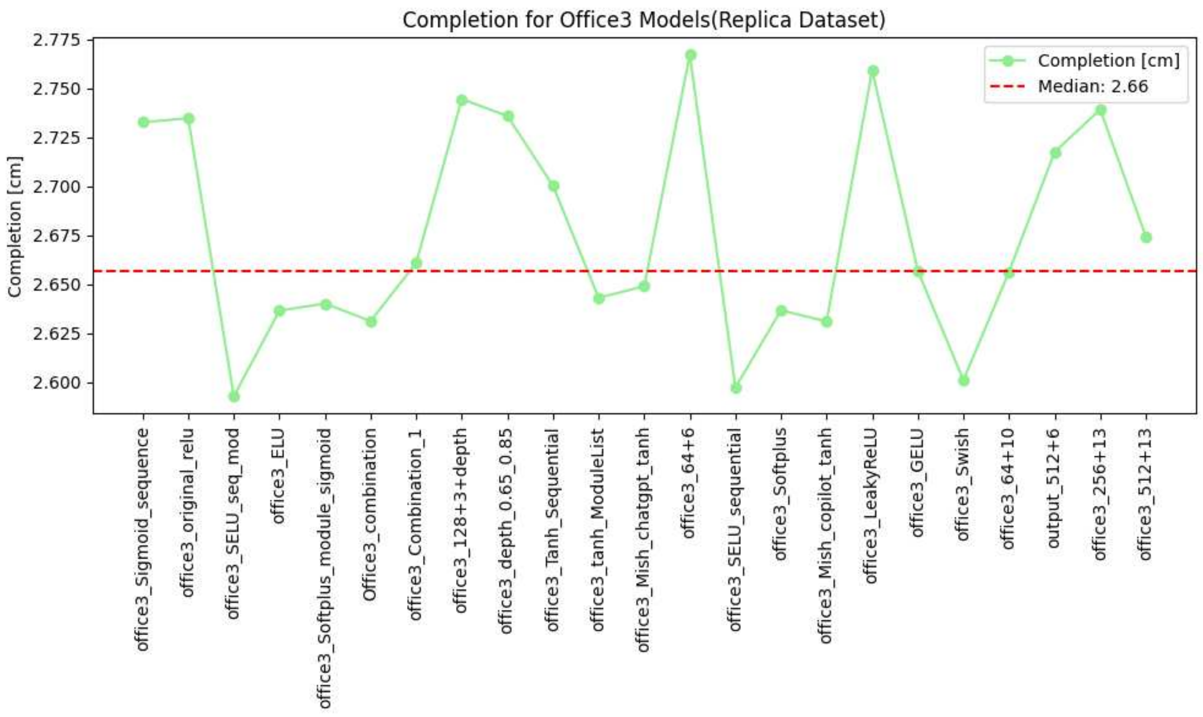

Figure 6.2: Completion for Office3 Models(Replica Dataset)

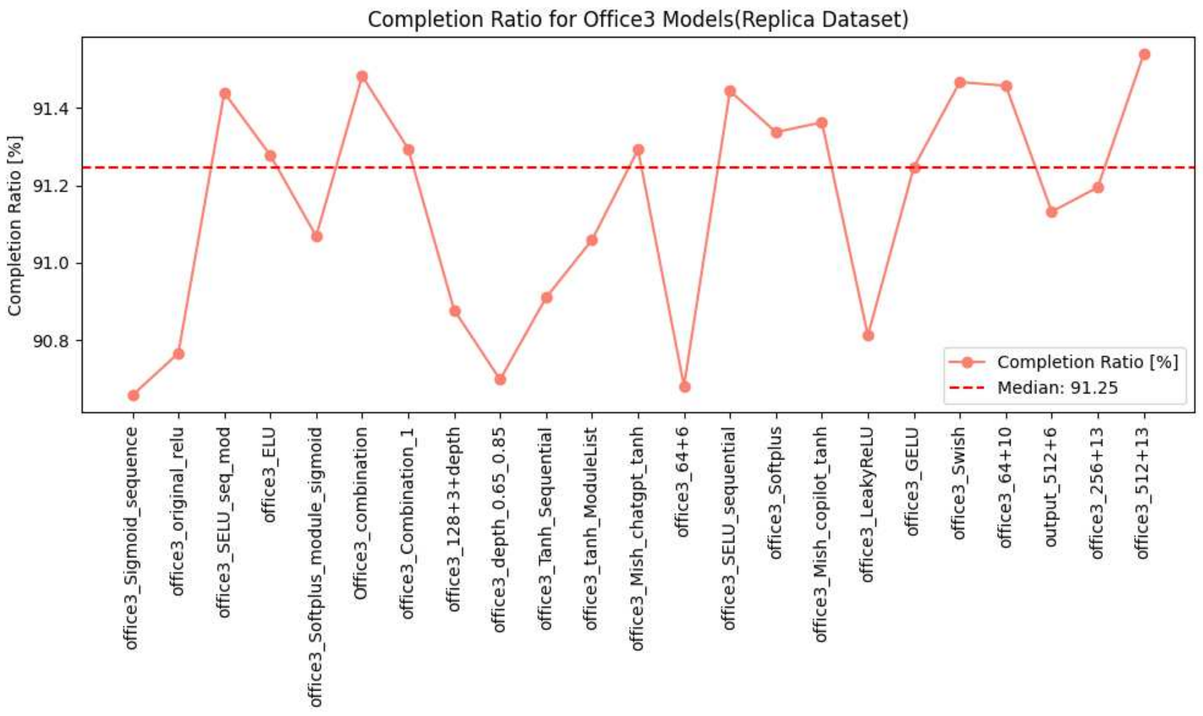

Figure 6.3: Completion Ratio for Office3 Models(Replica Dataset)

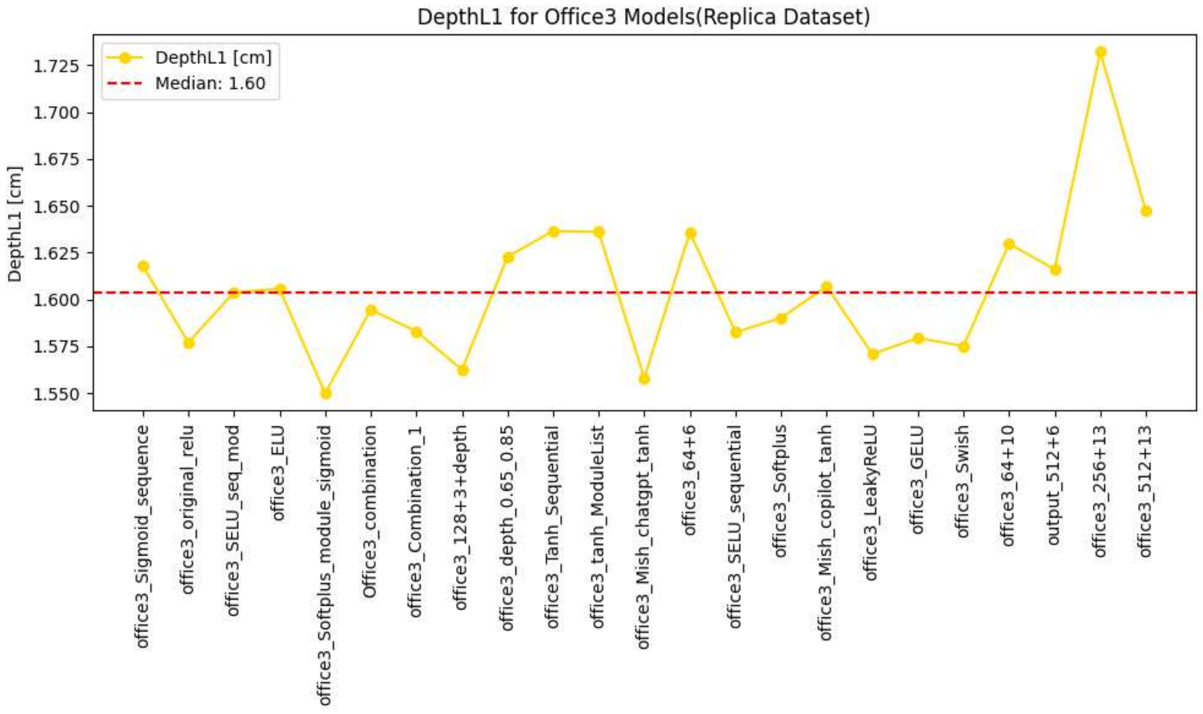

Figure 6.4: DepthL1 for Office3 Models(Replica Dataset)

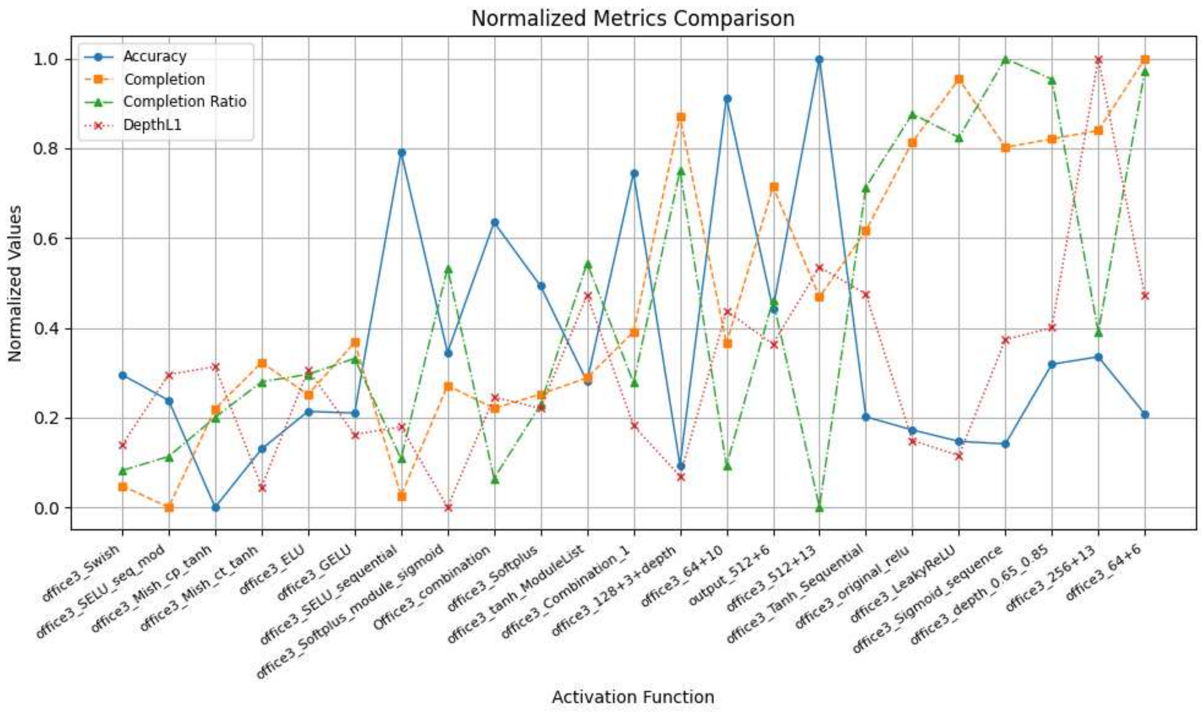

Figure 6.5: Normalized Metrics Comparison)

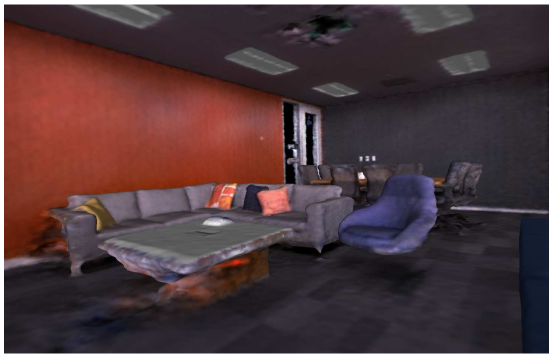

Figure 6.6: Our Method

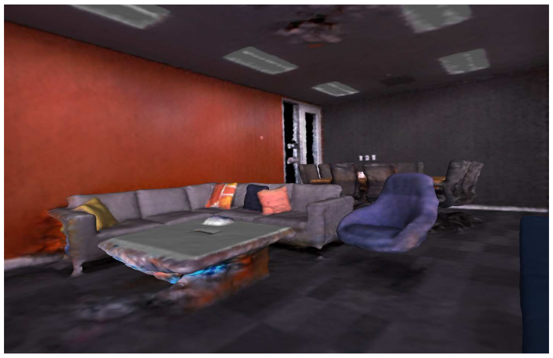

Figure 6.7: office3 Swish

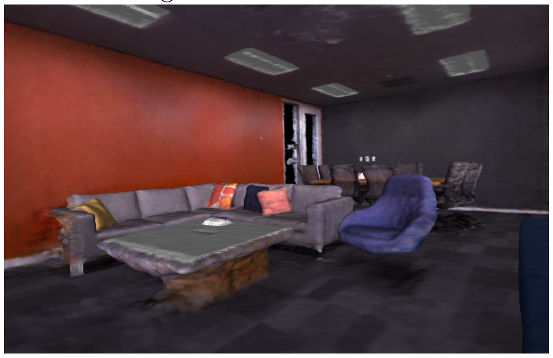

Figure 6.8: office3 128+3+depth

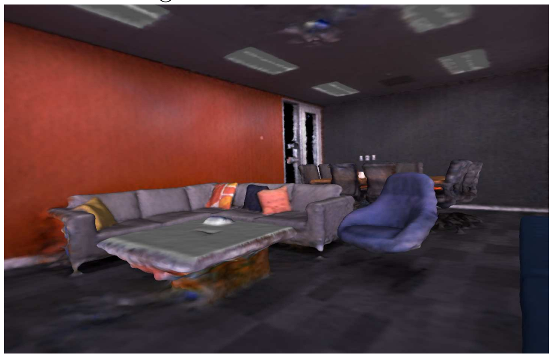

Figure 6.9: office3 Mish cp tanh

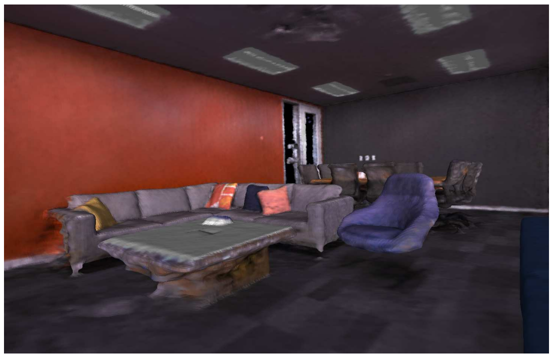

Figure 6.10: Our Method (Alternate)

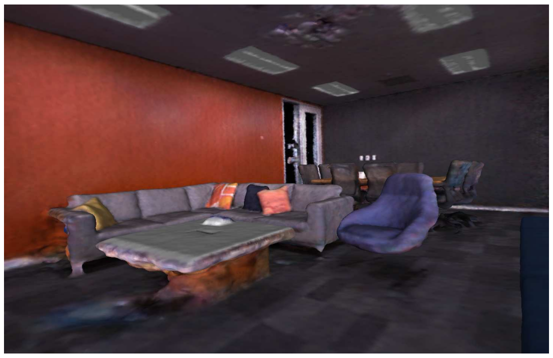

Figure 6.11: office3 SELU sequential

Figure 6.12: Visual output of 1 side from considered methods

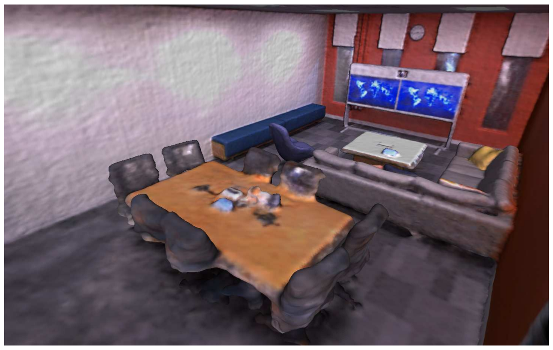
Figure 6.13: Our Method

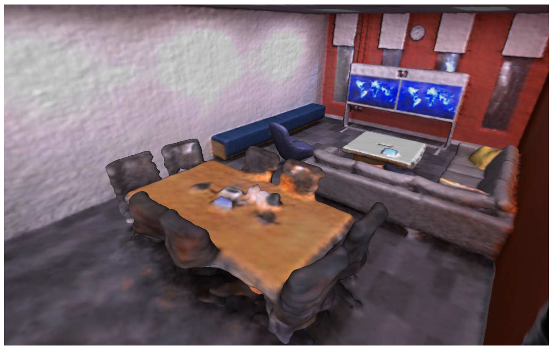
Figure 6.14: office3 Swish

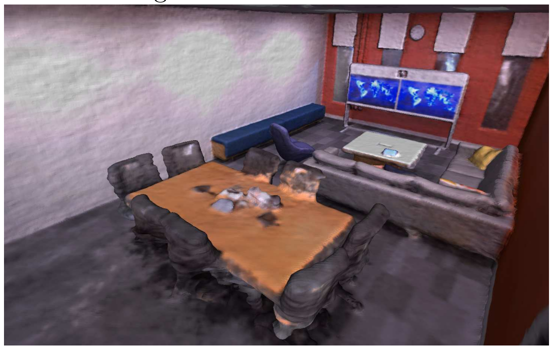
Figure 6.15: office3 128+3+depth

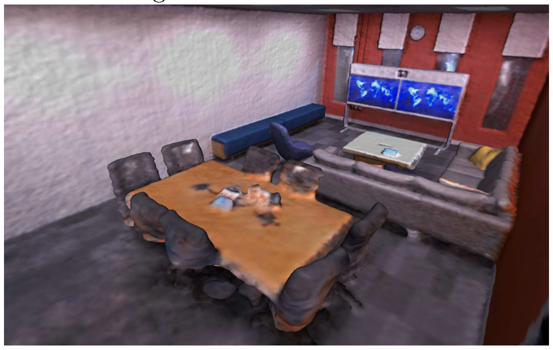
Figure 6.16: office3 Mish cp tanh

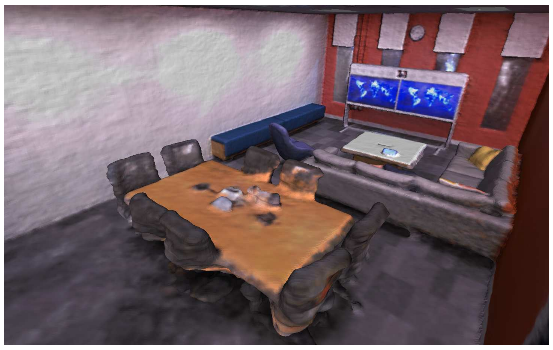
Figure 6.17: Our Method (Alternate)

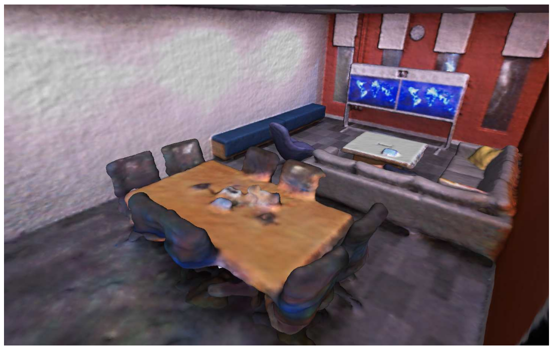
Figure 6.18: office3 SELU sequential

Figure 6.19: Visual output of another side from considered methods

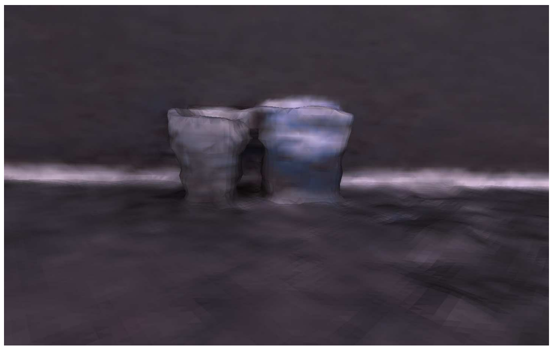

Figure 6.20: Our Method

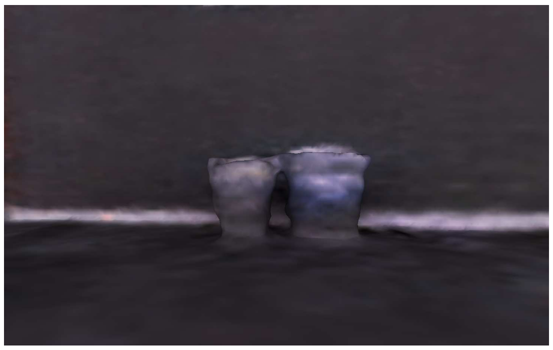

Figure 6.21: office3 Swish

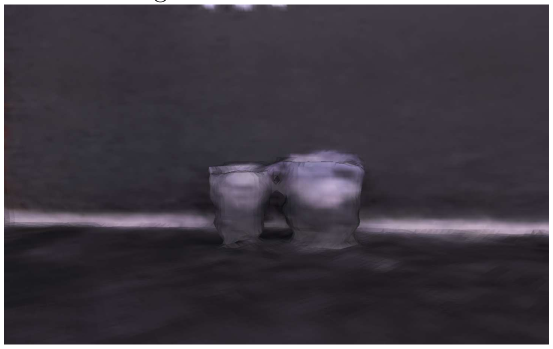

Figure 6.22: office3 128+3+depth

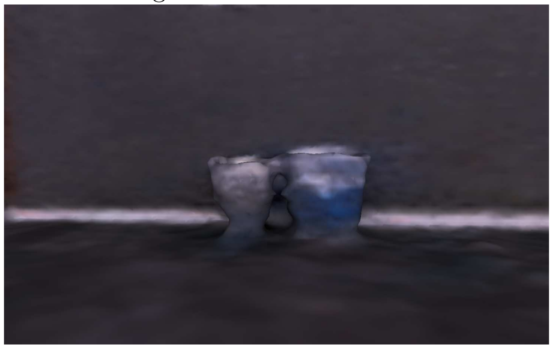

Figure 6.23: office3 Mish cp tanh

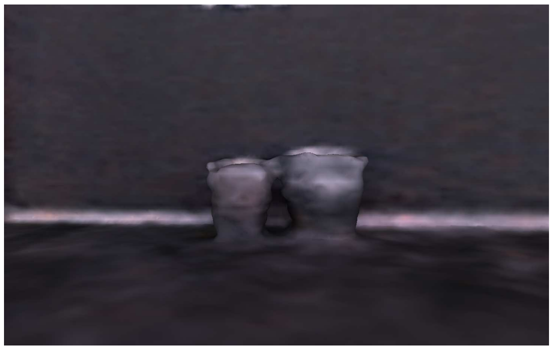

Figure 6.24: Softplus

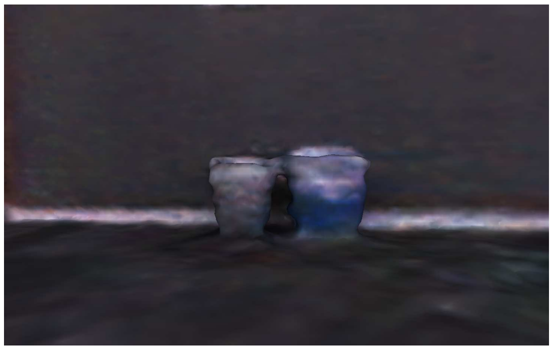

Figure 6.25: office3 SELU sequential

Figure 6.26: Visual output of objects from considered methods

6.3 Experimental Findings

- **Performance on Replica Dataset:**
  - Models such as `Mish_cp_tanh` and `SELU_sequential` exhibited improvements in accuracy and completion ratio, demonstrating enhanced object detection and mapping performance.
- **Performance on Synthetic RGBD Dataset:**
  - Consistent trends were observed, with `Mish_cp_tanh` and `SELU_sequential` performing better in accuracy and completion ratio across datasets.

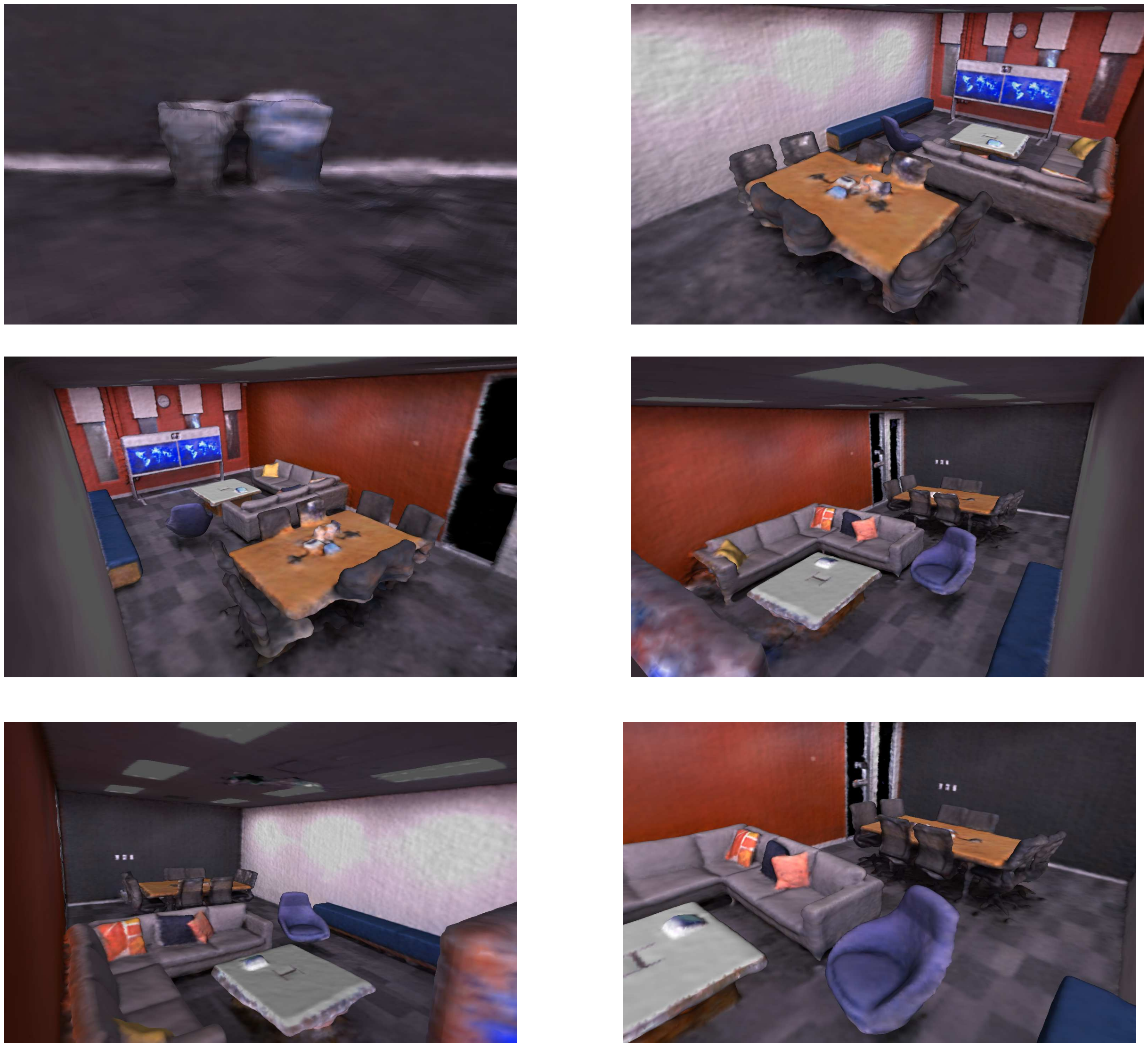

Figure 6.27: Our method Overall output

7 Results

In our investigation into improving the efficiency, accuracy, and overall performance of 3D mapping and localization for ground vehicles, we tested two variations of our method against the Co-SLAM baseline and other existing methods such as iMAP, NICE-SLAM, and Vox-Fusion. The two variations of our method differed in the hidden dimensions and the number of layers used in the SDFNet and ColorNet networks.

Our Method:

- **Hidden Dimension:** 32
- **Number of Layers (SDFNet):** 2
- **Activation Functions:** SELU, Sigmoid, Mish

Our Method (Alternate):

- **Hidden Dimension:** 128
- **Number of Layers (ColorNet):** 3
- **Number of Layers (SDFNet):** 5
- **Activation Functions:** SELU, Sigmoid, Mish

We utilized the Replica (Office3) dataset for our experiments. The following tables summarize the performance metrics of our methods in comparison to Co-SLAM and other existing methods.

7.1 Comparison with Co-SLAM

Table 7.1: Comparison of Our Methods with Co-SLAM

Metric	**Our Method**	**V/S Co-SLAM %**	**Our Method (Alternate)**	**V/S Co-SLAM %**	**Co-SLAM**
Accuracy ↓	green2.7098	2.7264	2.7624	0.8399	2.7858
Completion ↓	green2.5632	6.2744	2.6726	2.2745	2.7348
Completion Ratio ↑ [%]	green91.81	-1.15	91.39	-0.68	90.77
Variation Ratio ↓	green8.1925	11.2694	8.6130	6.7150	9.2330
Depth ↓	1.5976	-1.3049	green1.5620	0.9493	1.5770
Job duration ↓	08:48	-1.53846	09:51	-13.6538	green08:40

7.2 Comparison with Other SLAM Methods

Table 7.2: Comparison with Other SLAM Methods

Method	**Acc** ↓	**Comp** ↓	**Comp Ratio** ↑ [%]	**Depth** ↓
iMAP	4.20	5.49	73.90	5.61
NICE-SLAM	3.01	3.16	87.68	2.10
Vox-Fusion	2.33	2.81	89.10	1.82
Co-SLAM	3.06	2.72	90.72	1.66
Our Method	**2.71**	**2.56**	**91.81**	**1.60**

Key Observations:

- **Our Method:** Demonstrated superior performance in most metrics compared to Co-SLAM, particularly in accuracy, completion, and completion ratio.
- **Our Method (Alternate):** Showed improvements over Co-SLAM but did not significantly outperform the primary method.
- **Comparison with Other Methods:** Our method outperformed iMAP, NICE-SLAM, and Vox-Fusion in terms of accuracy, completion, and completion ratio. It also achieved the best depth performance among all methods tested.

8 Conclusion

Our proposed method, leveraging a combination of activation functions (SELU, Sigmoid, Mish) and optimized hidden dimensions and layers, demonstrated a notable improvement over the baseline Co-SLAM and other existing SLAM methods in several critical performance metrics. Specifically, the configuration with a hidden dimension of 32 and fewer layers (2 for both ColorNet and SDFNet) outperformed Co-SLAM in terms of accuracy, completion, completion ratio, and variation ratio.

The alternate method, with a higher hidden dimension and more layers, provided insights into the trade-offs between network complexity and performance gains. While it did show some improvements, the gains were not substantial enough to justify the increased complexity and computational overhead.

In summary, our primary method with optimized configurations provides a balanced approach to enhancing 3D mapping and localization for ground vehicles, achieving significant improvements in key performance areas while maintaining manageable computational demands. Future work can further explore the trade-offs between network complexity and performance, and investigate other activation functions and configurations to optimize the model further.

9 Future Work

9.1 Enhancing Accuracy with Architectural Innovations

Future research should focus on experimenting with advanced network architectures to further increase the accuracy of real-time 3D mapping and localization systems. Innovations such as integrating multi-scale feature extraction or leveraging attention mechanisms could significantly improve object identification and scene interpretation. These architectural enhancements will likely result in a more nuanced understanding of complex environments, thereby improving the system's decision-making capabilities in various operational contexts.

9.2 Improving Color Detection Precision

To address the current limitations in color detection, future work should explore more robust algorithms capable of performing accurately under diverse and challenging lighting conditions. Enhanced color detection will aid in better distinguishing objects, which is crucial for scenarios where lighting variability affects object visibility. Implementing techniques like adaptive color correction or advanced color space transformations could lead to more reliable and accurate object classification.

9.3 Reducing Computational Runtime

Optimizing the runtime of the 3D mapping and localization system is essential for maintaining real-time performance. Future efforts should concentrate on reducing computational overhead through strategies such as algorithmic optimizations, hardware acceleration, or efficient data processing methods. By decreasing runtime, the system will achieve faster response times, enhancing its ability to promptly and efficiently handle dynamic changes in the environment.

References

[1] T. Green and L. Black, "Review of traditional slam approaches for robotics," *International Journal of Robotics Research*, vol. 38, no. 5, pp. 920–945, 2019.

[2] M. Harris and E. Clark, "Limitations of traditional slam systems in large-scale environments," *Robotics and Computer-Integrated Manufacturing*, vol. 54, pp. 12–23, 2018.

[3] D. Miller and S. Thomas, "Learning-based approaches to visual slam," *Journal of Machine Learning Research*, vol. 22, pp. 223–250, 2021.

[4] J. Wilson and O. Martinez, "Neural radiance fields for 3d scene representation," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 55–67, 2022.

[5] S. Lewis and M. Young, "Limitations of neural radiance fields in real-time slam," *Computer Vision and Image Understanding*, vol. 203, pp. 102–115, 2021.

[6] A. Turner and M. King, "Nice-slam: A scalable approach to dense rgb-d slam," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 345–360, 2023.

[7] E. Harris and A. Scott, "Co-slam: Coordinated slam with joint coordinate and sparse grid encoding," *Computer Vision and Pattern Recognition*, vol. 24, no. 1, pp. 78–92, 2023.

[8] L. Brown and I. Johnson, "Enhancements in co-slam: Improved bundle adjustment and real-time performance," *International Conference on Computer Vision*, pp. 123–135, 2023.

[9] R. A. Newcombe, D. H. Fox, S. M. Seitz, and D. Fox, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 2011 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2011, pp. 127–136.

[10] M. Geringer, D. Horstmann, D. G. L. Kuettel, K. H. Park, M. Müller, H. G. W. Gölz, and H. B. Schmid, "Surfels: A three-dimensional representation for interactive simulations," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 1263–1270.

[11] R. K. M. M. K. O., "Voxelhashing: Real-time 3d reconstruction in dynamic environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013, pp. 37–44.

[12] M. W. and S. K., "Octree-based real-time 3d object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 31, no. 12, pp. 2087–2101, 2009.

[13] T. Whelan *et al.*, "Elasticfusion: Real-time dense slam and light source estimation," in *Robotics: Science and Systems*, 2015.

[14] S. F. J. L., "Bad-slam: Bundle adjustment for dense slam," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2021, pp. 1024–1038.

[15] V. Usenko *et al.*, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 422–429, 2020.

[16] B. Mildenhall, P. P. Srinivasan, M. Tancik *et al.*, "Neural radiance fields for view synthesis," in *Proceedings of the European Conference on Computer Vision (ECCV).* Springer, 2020, pp. 674–689.

[17] R. Martin-Brualla *et al.*, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[18] T. Wang *et al.*, "Hierarchical neural representations for scene generation," *NeurIPS*, 2019.

[19] L. Liu *et al.*, "Depth-aware neural radiance fields for view synthesis," *arXiv preprint arXiv:2103.14745*, 2021.

[20] K. Park *et al.*, "Dynamic neural radiance fields for monocular 4d facial avatar reconstruction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[21] X. Zhang, J. Zhao, and H. Liu, "imap: Incremental mapping and positioning for robots," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1021–1034, 2021.