Depth fusion visual SLAM

Hongqing Liu
hl85

Feiran Wang
feiranw2

Stanley Gu
Shgu2

Abstract

NeRF has demonstrated outstanding performance in high-fidelity and fast reconstruction. The implementation of neural implicit representation holds promise for Simultaneous Localization and Mapping (SLAM) in scenes with limited boundaries, such as indoor environments like rooms. However, outdoor scenes pose greater challenges due to their unbounded nature and sparser semantic information. In this work, we present a depth fusion visual SLAM that enables stable SLAM in outdoor scenes using color and depth inputs. Our approach consists of a depth prediction architecture to enhance depth information and a SLAM architecture for tracking and map creation. We design a depth fusion technique to combine estimated depth with captured depth, refining the depth information. To improve and stabilize tracking, we introduce a tracker model filter that uses two quaternion parameters instead of nine rotation matrix parameters. We also develop a weighted sample module to filter irrelevant information for tracking. Our method is implemented on our own dataset, collected using a ZED 2 camera. Project page: Github.

1. Introduction

Visual Simultaneous Localization and Mapping (SLAM) is a critical technology in various applications, such as robotics [4], navigation [1], and augmented reality [10]. It enables devices to understand their surroundings, build accurate maps, and navigate through complex environments. Over the past few years, RGB-D based SLAM systems have gained popularity due to their ability to provide detailed 3D reconstruction models and precise camera poses for tracking and mapping [16, 17]. This technology is particularly beneficial for indoor environments, where depth information is critical for creating accurate maps [22]. However, most existing methods are tailored for indoor scenarios and struggle to cope with the challenges posed by outdoor settings [6]. In this work, we aim to address this limitation and develop an effective real-time visual SLAM approach capable of generating detailed 3D maps for outdoor scenes.

1.1. Background and Problem Definition

The primary goal of our research is to develop an RGB-based visual SLAM system for outdoor environments. The existing state-of-the-art work, Nice-SLAM [27], is limited to indoor settings due to its reliance on RGB-D image streams, which provide accurate depth information only within a small range. Outdoor environments pose significant challenges for RGB-D based methods, as sensors struggle to obtain precise depth information for distant objects, ultimately affecting the reconstruction and mapping capabilities of previous methods [1].

Outdoor environments present a different set of challenges compared to indoor settings. They typically have a larger scale and dynamic elements such as moving objects [1] and changing weather conditions [7]. Additionally, outdoor scenes often contain repeating textures and patterns, which can cause ambiguity in feature matching and pose estimation [11]. Moreover, the availability of accurate depth information is essential for creating precise maps, but it is challenging to obtain in outdoor settings due to limitations in sensor range and accuracy.

1.2. Approach Outline

To overcome the aforementioned limitations, we propose modifications to the Nice-SLAM algorithm and introduce a tri-module model capable of achieving real-time visual SLAM and generating 3D maps for outdoor scenes.

Firstly, we incorporate the DeepPruner [3], a deep learning stereo algorithm, to estimate depth information from RGB images captured by binocular cameras. This enables our model to generate accurate depth maps even in challenging outdoor scenes with varying textures and scales more effectively.

Secondly, we implement sensor fusion [8] of depth images from a stereo camera and the depth information estimated by the DeepPruner. This fusion process combines the strengths of both methods, providing more accurate and reliable depth information, which is crucial for generating detailed 3D maps in outdoor environments.

Thirdly, we design weighted sample module to filter repeated texture and prioritize distinctive shapes by generat-

ing weighted map and sampling based on weighted, which dramatically improves the tracking.

Lastly, we enhanced the SLAM architecture by decreasing training parameters, which boost efficiency and performance significantly. In outdoor settings, tracking objects becomes considerably more difficult due to the increased velocity of the camera mounted on a moving vehicle.

We demonstrated the performance of our method on our own dataset collected through ZED 2 camera. To further showcase the effectiveness of the introduced modules, we made comprehensive ablation experiments and detailed analyses.

2. Related Literature

2.1. Implicit Neural Representations

Neural Radiance Fields (NeRF) [13], a recent development in 3D scene representation and rendering, has brought implicit networks into prominence. NeRF employs implicit networks to define a continuous 3D function, mapping 3D world coordinates to radiance values and volume densities [14]. The implicit function is derived from a sparse collection of 2D images taken from various viewpoints. Demonstrating remarkable results in generating novel perspectives of scenes with intricate geometry and appearance [21], NeRF has become an appealing choice for applications such as view synthesis and virtual reality [12].

Nonetheless, NeRF exhibits certain shortcomings that render it less appropriate for our proposed problem. Its primary limitation is the slow rendering speed, which is not conducive to real-time applications like SLAM. Moreover, NeRF necessitates a vast number of images and extensive training to achieve high-quality results [18], a condition that is impractical in dynamic outdoor environments where the scene is in constant flux.

2.2. RGB-based Simultaneous Localization and Mapping

RGB-based Simultaneous Localization and Mapping (SLAM) has its origins in early research on monocular SLAM, such as MonoSLAM [2], which utilized a single camera to estimate camera poses and construct sparse maps of the environment. The advent of more powerful computing devices and efficient algorithms has facilitated the development of advanced RGB-based SLAM systems, including ORB-SLAM [15] and LSD-SLAM [5]. These methods employ feature matching [19] and direct methods [24], respectively, for camera pose estimation and 3D map creation.

Nice-SLAM [27] is a cutting-edge method that utilizes RGB-D data for visual SLAM in indoor environments. Although it exhibits high accuracy and robustness, its dependence on depth information from RGB-D sensors renders it less suitable for outdoor settings where obtaining accurate depth information is challenging. Additionally, the feature matching techniques employed by numerous RGB-based SLAM systems often encounter difficulties in outdoor environments, which typically include large-scale scenes, dynamic elements, and variable lighting conditions [1].

2.3. Model-based Stereo Matching

Model-based stereo matching has an extensive history in the field of computer vision [20]. In recent years, the emergence of deep learning has led to the development of deep learning-based stereo matching methods [**?**], providing improved accuracy and robustness in comparison to conventional methods [26].

DeepPruner [3] is one such method that implements a neural network architecture for pruning the search space of pixel correspondences and efficiently estimating disparities. It has exhibited superior performance in comparison to traditional stereo matching methods, particularly in challenging scenes with occlusions, variable lighting conditions, and non-Lambertian surfaces [25].

However, deep learning-based stereo matching methods have their own set of drawbacks. They require large training data and are computationally demanding [9], which hinder their use in real-time applications like SLAM. They also struggle with large-scale outdoor scenes due to greater disparities between stereo images compared to indoor settings [23]. This affects depth estimation accuracy and 3D map generation.

3. Method

Our objective is to develop a visual SLAM for outdoor environments. In this section, we first present the model structure, which includes the depth prediction architecture and the SLAM architecture. Next, we describe the fusion of estimated depth and captured depth. We then introduce the weighted sample to enhance tracking. Finally, we explain how the tracker model filter improves adaptability to outdoor scenes and maintains robust performance.

3.1. Model Structure

We propose a novel visual SLAM model for outdoor environments that consists of the depth prediction architecture and the SLAM architecture, as shown in Fig 1. The depth prediction architecture is based on DeepPruner, which takes two RGB images as input and predicts depth information. We then fuse the estimated depth with the captured depth from the depth camera. The SLAM architecture builds upon the foundation of Nice-SLAM, which takes the generated RGBD information as input to track the camera pose and create a map.

The depth prediction architecture adopts the algorithm of DeepPrunner to predict the stereo Depth from two RGB images. The model uses differentiable PatchMatch to predict

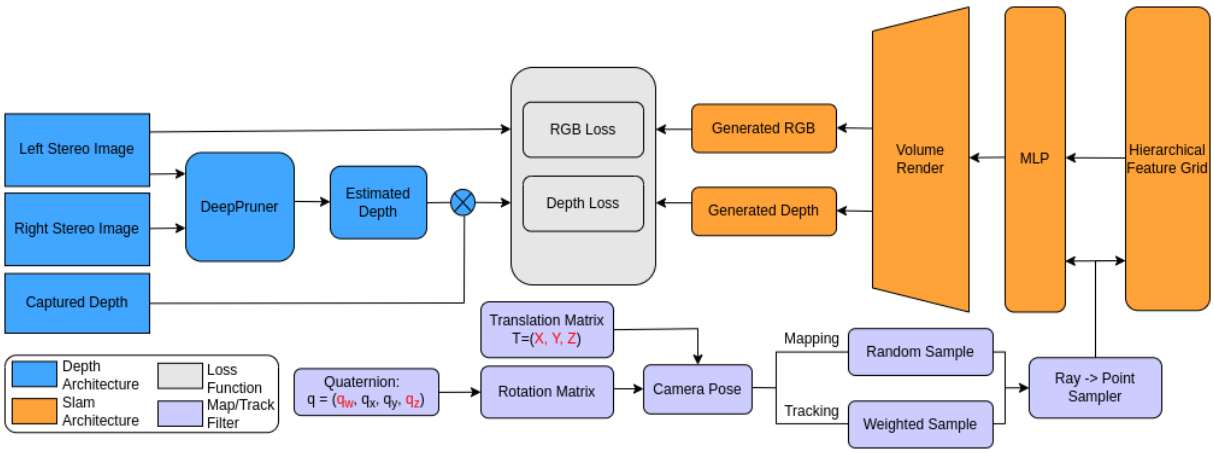


Figure 1. **Overview:** The model takes stereo images and the captured depth as input to track and create the map for outdoor scenes. Firstly, the Depth Architecture predicts depth based on stereo RGB images. Secondly, the depth fusion module fuses the captured and estimated depth with weights. Then, the fused depth and left color image are sent to Slam Architecture to track and generate the map. The Slam Architecture predicts depth, color and density for sampled points in neural implicit space and renders back to RGB and depth images. In Map/Track filter, we implement weighted sample and tracker model filter to enhance the tracking. **Tracking:** The model updates the q_m, q_z in Quaternion and X, Y, Z in translation matrix through back propogation of losses. Besides, weighted sample module filters useless information to improve the tracking. **Mapping:** The model only updates feature grids with backpropagation of losses.

depth. There are three layers in PatchMatch. The particle sampling layer would randomly generate K values for each pixel. The propagation layer will propagate the disparity information to adjacent pixels. The evaluation layer chooses the best value as the depth information for each pixel. Finally, the model aggregates the cost and refines the depth output. If the camera facility can only capture the stereo RGB information, depth prediction architecture could generate confident depth for the SLAM. If the camera could capture the depth information, the estimated depth would be the extra information to improve the depth. For instance, the ZED2 camera can only get depth information within 20 meters. The captured depth information is not enough for the RGBD SLAM. The depth prediction architecture could predict depth beyond 20 meters, which will bridge the gap. We finetune the DeepPruner on our dataset with 1000 data for 500 epochs, which helps the model learn the camera's intrinsic and extrinsic parameters to fit ZED 2 camera. The performance would be much better after sensor fusion on estimated depth and captured depth than using predicted depth or captured depth alone.

The SLAM architecture adopts the architecture of the NICE-SLAM algorithm [27]. The architecture takes color and depth as the input to track the camera pose and create a map simultaneously. The architecture creates a hierarchical Feature Grid to store the map in coarse, middle, and fine levels. Each level encodes the geometric information while the fine layer also incorporates color information. To render images from implicit space, the architecture firstly samples pixels from the image under an assumed camera pose, then casts rays for each pixel. Secondly, the architecture samples 3D points along the ray and calculates the trilinear interpolation features for each point based on the neighboring grid point features. Thirdly, features are sent to a pre-trained decoder network to predict color, depth, and density for each sampled point. Finally, the architecture renders the RGB image and depth image for each pixel through the differentiable color rendering process. The architecture minimizes the reconstruction loss between the rendered and ground truth keyframes to update the grid features and camera pose. More specifically, the system operates on three separate threads. One thread is dedicated to tracking, where the system fixes the features of the hierarchical feature grid and trains the camera pose. Two threads are assigned to mapping, where the system fixes the camera pose and trains the hierarchical feature grids for the coarse, middle, and fine levels. By simultaneously training the grid features and camera pose on different threads with different frequencies, the system is capable of running the V-SLAM algorithm in real time.

3.2. Depth fusion module

We introduce a depth fusion module to fuse the estimated depth images and the captured depth images.

The primary limitation of the captured depth image is the limited detecting range from 0.4 to 20 meters, which is restricted by the depth sensor's technical specifications. Nonetheless, the ground truth depth image provides highly precise distance measurements when objects are detected within the 20-meter range. Besides, even within 20 meters,

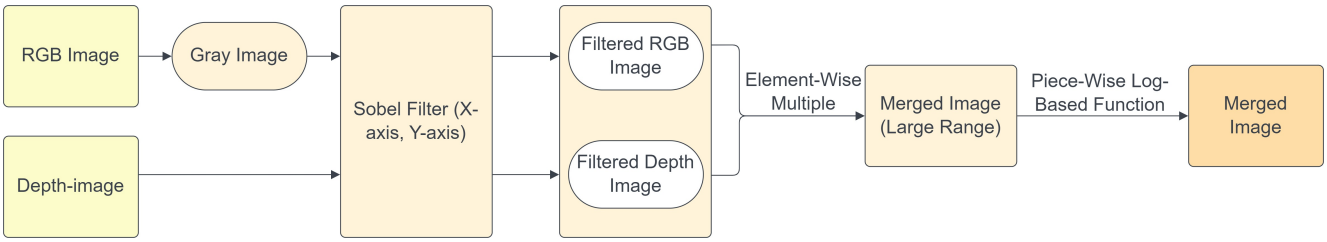


Figure 2. Pipeline of the weighted sample method

some area shows an Infinite distance due to the reflection.

The pros of estimated depth image are that it can provide reasonably accurate depth information for uniform areas and objects beyond 20 meters, resulting in more reasonable geometry relationships and object distances. However, its accuracy for objects within 20 meters is not as accurate as captured depth images.

In order to obtain a more accurate depth image and also keep the image's geometric continuity, we propose to integrate the estimated depth image and captured depth image with weighted. We utilize the captured depth to calculate the weights for each pixel.

The equation 1 and visualization of the fusion weight Fig 3 are presented below.

$$\text{weight}(x, a) = \frac{1}{1 + e^{-a(x-20)}} \tag{1}$$

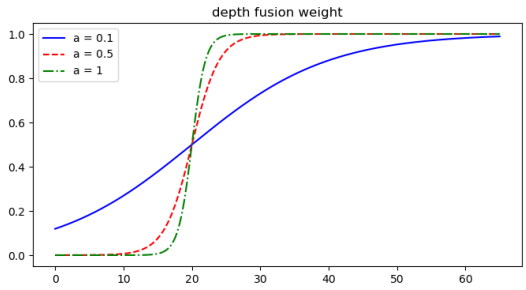


Figure 3. Depth fusion weight. This image shows curves that depict three different weights used for depth fusion.

The formulation of the merged depth is shown as equation 2.

$$\text{M} = \text{weight} * \text{P} + (1 - \text{weight}) * \text{C} \tag{2}$$

where the M means merged depth fusion, P means predicted depth image, C means captured depth image. We assign a higher weight to the ground truth depth image when the distance value is less than 20 meters, as it is known to be more accurate in this range, while we assign a higher weight to the estimated depth image for distances beyond 20 meters, where it is expected to provide a more reasonable representation of the scene geometry.

3.3. Weighted sample for tracking

We introduce the weighted sample method to prioritize sampling pixels on objects with more distinctive shapes and features. The pipeline of the weighted sample method is presented in Figure 2, which consists of two parts: generating the weighted map and sampling based on the weighted map.

In the first part, the RGB image is converted to grayscale and both the RGB grayscale image and depth image are filtered with a Gaussian blur filter to reduce noise. Subsequently, a Sobel filter is applied to filter the RGB grayscale and depth images along the x-axis and y-axis, and then weighted combined on each image. The RGB grayscale image is effective in filtering out most of the lane, while the depth image can effectively filter out most of the sky pixels. In the next step, the two filtered images are multiplied to obtain the merged weighted map. The intuition behind this step is the "or" operation, where the sharper parts have larger values and the smoother parts have lower values. During testing, it was discovered that the value for the larger parts is too high, which can lead to poor weighted results in the next step. In order to tackle this problem, a piecewise function , as shown in Equation 3, was employed to attenuate the discrepancies present in the weighted sample image.

$$f(x) = \begin{cases} x & \text{if } x < 1 \\ 1 + \ln(x) & \text{otherwise} \end{cases} \tag{3}$$

In the second part, a list is created to accumulate weighted values in weighted map, which are subsequently normalized to a range between 0 and 1. Next, n random values between 0 and 1 are generated. For each weighted sample, the index of the first accumulated value that larger than the corresponding random value is selected. The schematic diagram illustrates this approach is presented below. The rationale for this method is that pixels with larger weighted values are more likely to be chosen, thus allowing for the sampling of more points in areas with sharper features than in smoother regions.

The majority of the pixels in a camera scene are captured by a vehicle corresponding to the lane. When the car

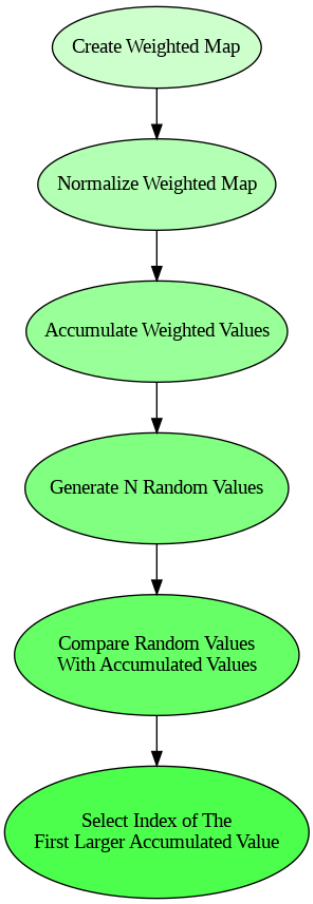


Figure 4. weighted sampling schematic diagram

is moving forward, the depth and RGB images of the lane show relatively minor variations, thereby posing a challenge for the system to accurately track the camera pose. The tracking pipeline, as described in section 3.1, involves generating sample rays by sampling pixels in an image using an assumed camera pose, and creating sample points along those rays to render the image. By comparing and training based on the render rgb-d and ground truth rgb-d, the system can find the optimal camera pose. If the sampling of pixels is random, the majority of them will correspond to pixels of the lane. Since the ground truth depth value and RGB information of the lane are not changed significantly when the car is moving forward, the system cannot effectively track the camera pose.

3.4. Tracker model filter

We introduce the tracker model filter to decrease the training parameters by transforming the projection matrix from 2 rotation quaternion parameters and 3 translation parameters.

Due to the increased velocity of the camera mounted on a moving vehicle, tracking objects in an outdoor environment becomes significantly more challenging. Utilizing the original tracker model needs more iterations to attain optimal performance, which, in turn, impairs the system's real-time capabilities as the tracker must be executed in each frame. Given that the motion models of both the camera and lidar sensors conform to the Dubin car motion model, it is possible to leverage certain a priori assumptions for the tracker.

There are two optimal strategies that can be employed to improve tracking performance in the Dubin car motion model. Firstly, due to the fact that translation typically changes faster than rotation, the system separates the two parameters and allocates a larger learning rate to translation. Secondly, the camera's rotation is limited to its z-axis (from down to up) according to the car model. To reduce the number of parameters during tracking, the rotation matrix can be converted to a quaternion and only the z and w values need to be trained.

Given a quaternion $q = (q_w, q_x, q_y, q_z)$, the corresponding rotation matrix R can be calculated as follows:

$$R = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_xq_y - 2q_wq_z & 2q_xq_z + 2q_wq_y \\ 2q_xq_y + 2q_wq_z & 1 - 2q_x^2 - 2q_z^2 & 2q_yq_z - 2q_wq_x \\ 2q_xq_z - 2q_wq_y & 2q_yq_z + 2q_wq_x & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix}$$

In light of these considerations, a new step has been added to the tracker training pipeline. Firstly, the input variables are two rotation quaternion parameters (z and w) and three translation parameters. Subsequently, these variables are transformed into the projection matrix, which provides the camera pose required by the system to generate sample rays.

3.5. Loss

Inherited from the idea of Nice-SLAM [27], Our model consists of three threads to speed up the optimization:

One thread for depth and color optimization, the color loss $\mathcal{L}_p$ is L_1 loss between the observation I and the generated color $\hat{I}$. $\mathcal{L}_d^c$ and $\mathcal{L}_d^f$ are L_1 loss between the observation D and the generated depth $\hat{D}$ in coarse and fine level.

One thread for mapping, the reconstruction render loss $\mathcal{L}_r$ in Eq. (4) is the combination of geometric loss in coarse level $\mathcal{L}_d^c$, fine level $\mathcal{L}_d^f$ and color loss $\mathcal{L}_p$ with weight λ_m to M pixels.

$$\mathcal{L}_r = \mathcal{L}_d^c + \mathcal{L}_d^f + \lambda_m \mathcal{L}_p \tag{4}$$

One thread for tracking, the track loss $\mathcal{L}_t$ in Eq. (5) is the combination of modification of depth loss and color loss $\mathcal{L}_p$ with weight λ_t to T pixels.

$$\mathcal{L}_t = \frac{1}{T} \sum_{t=1}^{T} \frac{\left| D_m - \hat{D}_m^c \right|}{\sqrt{\hat{D}_{\text{var}}^c}} + \frac{\left| D_m - \hat{D}_m^f \right|}{\sqrt{\hat{D}_{\text{var}}^f}} + \lambda_t \mathcal{L}_p \tag{5}$$

4. Experiments

In this section, we first describe the setup environment. Then we explain the collection and preprocess of our own dataset. Thirdly, we visualize the result of our modules and rendered maps. Finally, we made a comprehensive ablation experiment and analysis.

4.1. Implement details

We run our system on a desktop PC with a 2100MHz Intel i7-12700F CPU and an NVIDIA RTX 3090 Ti GPU. In all our experiments, we use the number of sampling points on a ray N-sample = 64 and N-surface = 32. We select K = 5 keyframes and sample M = 2000 pixels in each image. As we sample more points on the image and render more sample points, the mapping and tracking performance increases. Those parameters are the trade-off result between mapping performance and real-time capabilities.

4.2. Dataset

Highbay dataset: We create our own dataset for the model. There are two reasons to use the own dataset: Firstly, the dataset could be used to train the model. Then we could implement our model in the autonomous vehicle to test the performance in the real scene. Secondly, to render the predicted image and depth in the SLAM architecture, objects have to be static. Once the objects move, the map will contain residual shadow. Therefore, we need an outdoor dataset with RGB and depth images for static scenes.

The dataset contains several scenes when the car is moving. The scene is located at the Research Park of UIUC. We sample 1000 frames to finetune the DeepPrunner and 500 frames to run RGB-SLAM. The RGB and depth information was collected by ZED 2 camera which is set on an autonomous vehicle with ROS system and two 3090 GPUs. The depth information ranges from 0.4 to 20 meters, containing "NAN" which is caused by the difference between two images, and "INFINITY" which means too far to calculate. Therefore, we transform "NAN" to 0 meters and "INFINITY" to 65 meters which is much larger than the maximum range of the ZED 2. To fit the scale of the SLAM architecture, we transform depth from meters to millimeters and save with uint16 type.

4.3. Result

4.3.1 Qualitative and Quantitative Analysis

To evaluate on High-bay parking lot, we use a depth fusion SLAM algorithm with a depth fusion module, weighted sample, and tracker model filter. Our method is able to reconstruct the geometry in the rendered cube precisely within limited iterations. As shown in Fig. 5, there are two frames of results with depth and color information.

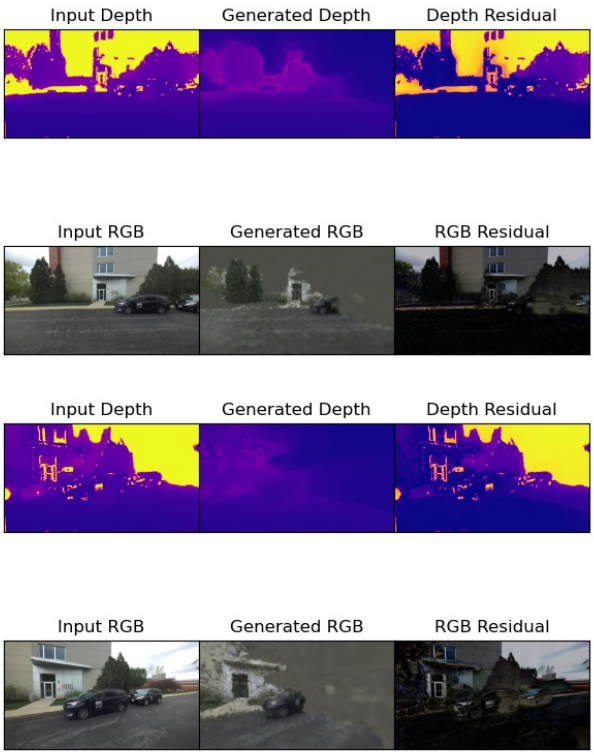


Figure 5. Reconstruction Results in frames 50 and 150 on the parking lot.

iteration	Best Result	No Depth fusion
0	6.7866	6.1715
1	0.2422	0.3644
50	0.2823	0.6633
100	0.3643	0.7124
150	0.4498	1.0352
200	0.4604	1.3453
250	0.4763	1.4083

Table 1. Reconstruction loss on parking lot

From left to right, we visualize the captured images, generated images, and residual images for color and depth.

The generated results successfully map the outdoor scene including cars and trees and filter things outside of the cube, such as the sky which represents gray. Besides, the correct render demonstrates the success of tracking.

As shown Best Result in Tab. 1, our method has a low reconstruction render loss and in Tab. 2 out tracking loss is also very low compared with the ground truth pose.

iteration	Best Result	No Module 2	No Module 3
0	0.0145	0.0241	0.0179
1	0.0212	0.1026	0.1147
50	0.0134	0.1153	0.1324
100	0.0131	0.2435	0.2298
150	0.0351	0.4532	0.3907
200	0.0236	0.4732	0.3670
250	0.0148	0.8254	0.3961

Table 2. Tracking loss on parking lot; Module 2: Tracker model filter; Module 3: Weighted sample filter

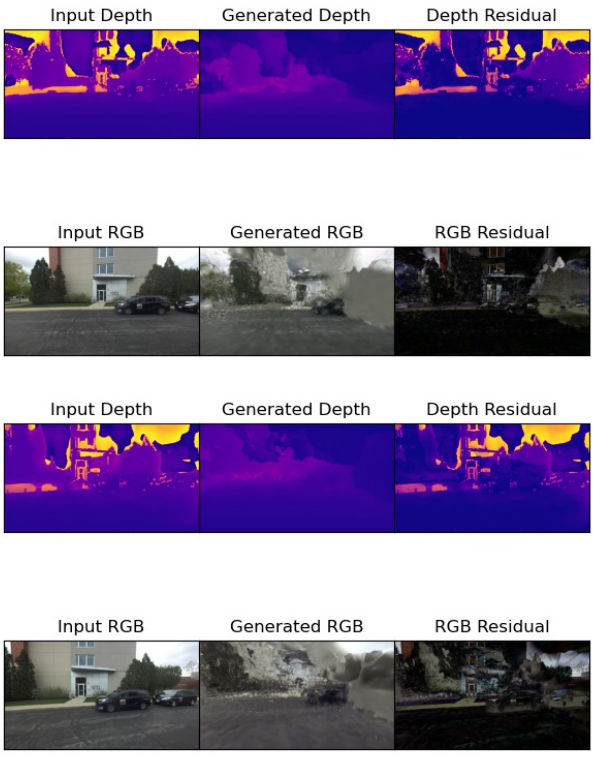


Figure 6. Mapping results of only using captured depth information for frames 50 and 150

4.4. Ablation Experiments

4.4.1 Ablation on depth fusion

We implement the depth fusion module to enhance mapping. we present visualizations using only captured depth in Fig. 6. Compared to the merged depth in Fig. 5, the map is blurred when only using the captured depth, and objects beyond the cube like sky aren't filtered as gray. The reconstruction render loss of "No Depth fusion" in Tab. 1 is much higher than the Best result which uses the Depth fusion Module.

In Fig. 7, we visualize the captured, estimated and merged depth images: the left top is the captured depth image, the right top is the estimated depth image, the left bottom is captured RGB image and the right bottom is merged depth image. The captured depth image has a bad geometry representation for objects over 20 meters and smooth objects in the range of 10 to 20, the merged depth image presents a more reasonable representation of the scene geometry.

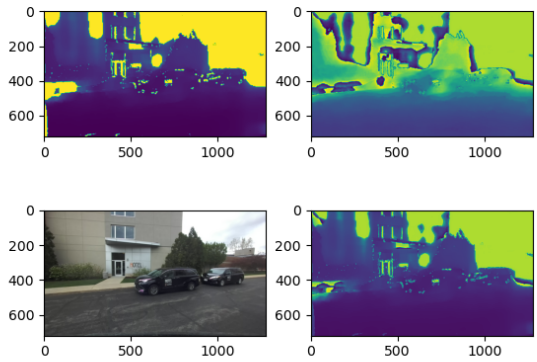


Figure 7. The Comparison of captured, estimated, and merged depth information

4.4.2 Ablation on tracker model filter

The primary purpose of the tracker model filter method is to enhance tracking performance. As a result, we present visualizations of its mapping output in Fig. 8 as a comparison with Best result in 5.

As shown in the result, without a tracker model, the system can not track the camera effectively and precisely in the outdoor environment, its estimated camera pose rotated along x-axis when we do not fix it to rotate around z-axis. Bad tracking result also leads to bad mapping result. It is important to note that the gray area is the part that is out of the bounding cube, so the system does not render that part. Besides, its corresponding tracking loss "No Module 2" in Tab. 2 is much higher and unstable than Best Result which uses tracker model filter.

4.4.3 Ablation on weighted sample filter

The primary purpose of the weighted sample filter method is to enhance tracking performance. The corresponding tracking loss "No Module 3" is shown in Tab. 2, which is increasing with the iteration. The sample weighted map of one frame is shown in Fig. 9, the left top image is the ground truth color image, the right top is captured depth

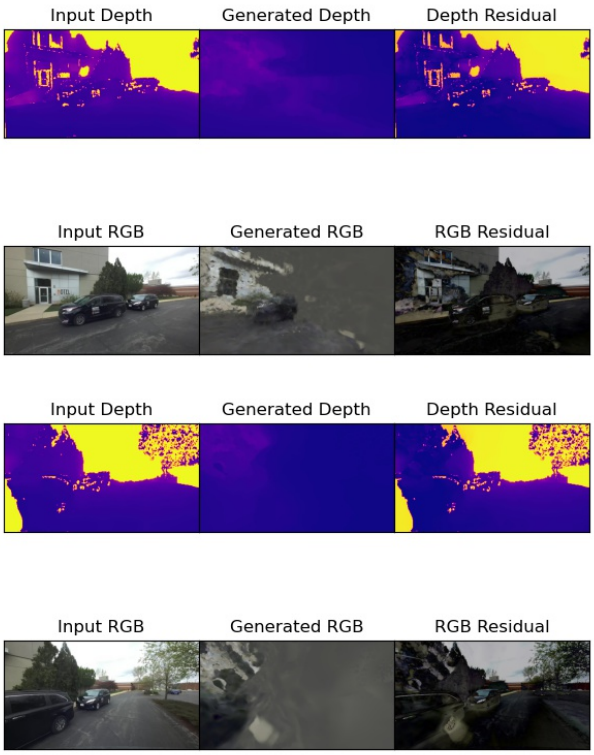
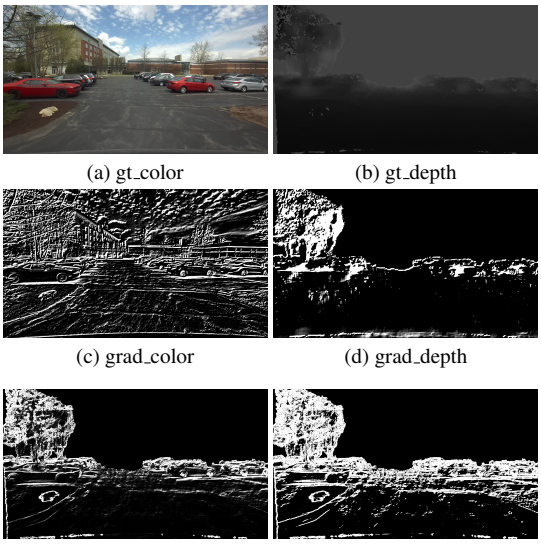


Figure 8. Mapping results of not using tracker model filter for frames 50 and 150

image, the left middle is the color image after sobel filter and the right middle is depth image after sobel filter, left bottom is weighted map with a large range and the right bottom is weighted map with small range. It is clear that after multiplying the gradient color and gradient depth images, the resulting weighted map can filter out the sky and lane. However, the edges have high values, which can cause the sample points to focus too heavily on those areas. After applying a piece-wise logarithmic operation, the map attenuates the discrepancies present in the weighted sample image. We utilize the result of the weighted sample points for tracking. Without this module, the tracking will collapse.

5. Conclusion

We present a novel method for outdoor scenes that incorporates a depth prediction architecture to refine the depth information and a Slam architecture to track and create the map. To overcome the challenge outdoor, we design a depth fusion module, weighted sample and tracker model filter, which dramatically enhance the model's performance. The main limitation of our method is the limited slam space due to the huge consumption of memory, which lead to grey space in generated results. To further improve the model, we could utilize a hash grid to replace the fixed feature grid



(a) gt_color (b) gt_depth

(c) grad_color (d) grad_depth

(e) weighted map with large range (f) weighted map with small range

Figure 9. weighted sample results

with an adaptive grid more effectively, which will be our next research direction.

References

[1] César Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[2] Andrew J. Davison. Real-time monocular slam: Why filter? In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 1, pages 265–271. IEEE, 2003.

[3] Shubham Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 883–899, 2018.

[4] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping (slam): Part i the essential algorithms. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[5] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[6] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Direct sparse odometry. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 40, pages 611–625. IEEE, 2018.

[7] Sourav Garg, Vijay Dhiman, Hanmei Wang, Michael Devy, and Aleš Leonardis. Adverse weather benchmark for slam. In *2020 IEEE/RSJ International Conference on Intelligent*

Robots and Systems (IROS), pages 10777–10784. IEEE, 2020.

[8] Sakshi Gupta and Itu Snigdh. Chapter 19 - multi-sensor fusion in autonomous heavy vehicles. In Rajalakshmi Krishnamurthi, Adarsh Kumar, and Sukhpal Singh Gill, editors, *Autonomous and Connected Heavy Vehicle Technology*, Intelligent Data-Centric Systems, pages 375–389. Academic Press, 2022.

[9] Alex Kendall, Yarin Gal, and Roberto Cipolla. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[10] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234. IEEE, 2007.

[11] David G Lowe. Distinctive image features from scale-invariant keypoints. In *International journal of computer vision*, volume 60, pages 91–110, 2004.

[12] Ricardo Martin-Brualla, Noha Radwan, and Noah Snavely. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2020.

[13] Ben Mildenhall, Vincent Sitzmann, Ruben Noll, Justus Thies, Michael Niemeyer, Arno Zamir, Alexei A. Efros, Gordon Wetzstein, and Michael Zollh"ofer. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–421, 2020.

[14] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[15] Raul Mur-Artal, Jose Maria Martínez Montiel, and Juan D. Tardos. Orb-slam: a versatile and accurate monocular slam system. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2015.

[16] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[17] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011.

[18] Jeong Joon Park, Peter Florence, Julian Straub, Richard A Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[19] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011.

[20] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[21] Katja Schwarz, Stephan P Lichtenberg, Ira Kemelmacher-Shlizerman, Noah Snavely, and Steve Seitz. Graf: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, 2020.

[22] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012.

[23] Alessio Tonioni, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Real-time self-adaptive deep stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12210–12219, 2019.

[24] Tommi Tykkälä, Cedric Audras, and Andrew I Comport. Direct iterative closest point for real-time visual odometry. In *ICCV Workshops*, 2011.

[25] Gengshan Yang and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5510–5518, 2019.

[26] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *CVPR*, 2019.

[27] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022.