Hierarchical Dense Neural Point Cloud-based SLAM

Liuxin Qing* Longteng Duan* Guo Han* Shi Chen*
ETH Zürich
Rämistrasse 101
8092 Zurich, Switzerland
{liuqing, loduan, guohan, shichen}@student.ethz.ch

Abstract

The development of dense visual simultaneous localization and mapping (SLAM) has been significantly facilitated by the recent emergence of scene representations based on neural radiance field (NeRF). Point-SLAM, a novel approach that anchors encoded scene features in a point cloud, achieves state-of-the-art tracking performance on synthetic dataset Replica. However, when applied to real-world datasets like ScanNet, it tends to suffer from tracking instability. We attribute this degradation to the inherent sensitivity of Point-SLAM to imaging effects specific to real domains, such as motion blur and specularities. Furthermore, we suspect that the finer details rendered by Point-SLAM actually increase the chance of being trapped at local minima, consequently hindering camera pose optimization. To address this issue, we build upon Point-SLAM a coarse-to-fine optimization strategy that leverages multiple sets of point cloud with varying resolutions. Indeed, we found through quantitative evaluation that our technique improves the ATE RMSE of tracking by 21.27% on average over Point-SLAM on the ScanNet dataset.

1. Introduction

Dense visual Simultaneous Localization and Mapping (SLAM) has gained significant attention within the field of 3D computer vision due to its wide range of industrial applications, including autonomous vehicle navigation and mixed/augmented reality (MR/AR). While many dense SLAM studies are based on traditional methods [5, 10, 12], recent years have witnessed a notable shift towards learning-based approaches [1, 19, 15]. Particularly, the incorporation of Neural Radiance Field (NeRF) [8] has revolutionized scene representation by offering impressive photometric accuracy. The pioneering work iMap [14] introduced real-time dense SLAM that utilizes neural implicit representations to encode scenes. NICE-SLAM [20], built on the advancement of iMap, further enabled accurate camera tracking and mapping in larger indoor scenes by optimizing multi-level spatially partitioned feature grids. Instead of storing features in regular grids, Point-SLAM [11] proposed an adaptive point cloud-based scheme that populates neural points depending on information density in the scene in order to achieve a better memory vs. accuracy tradeoff. As a result, Point-SLAM is capable of rendering finer details than NICE-SLAM, and also exhibits better tracking performance.

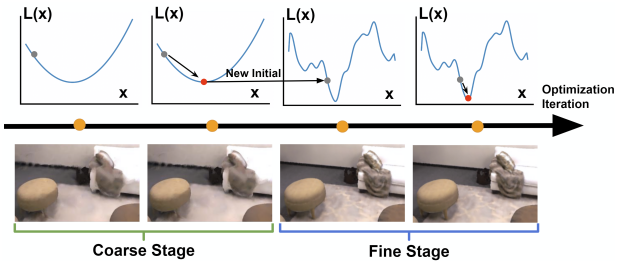

Figure 1: **Illustration of multi-stage optimization scheme.** The coarse stage optimization gives a better initialization for the fine one to prevent getting stuck in local minima.

Despite its superior performance in synthetic settings, Point-SLAM [11] encounters challenges in maintaining stable camera pose tracking in certain real-world conditions. On the ScanNet dataset [2], which is a scanned real-world dataset, Point-SLAM is only on par with NICE-SLAM [20] in terms of tracking quality. We attribute this degradation of tracking accuracy in real-world scenes to the sensitivity of Point-SLAM to imaging effects specific to real domains (e.g. motion blur, specularities). We hypothesize that the higher sensitivity arises due to Point-SLAM's capacity to render finer geometric and appearance details, which, in turn, increases the likelihood of the optimization process being trapped in local minima.

*These authors contributed equally to this work

To mitigate this issue, we propose a coarse-to-fine strategy tailored to the Point-SLAM framework. During mapping, we generate multiple sets of point clouds with different coarseness, which is controlled by specific preset sampling radii. During tracking, we adopt a multi-stage scheme where the camera pose is first optimized based on coarser point clouds and subsequently refined using finer ones. Figure 1 gives an intuition of the multi-stage scheme.

To validate the effectiveness of our coarse-to-fine strategy, we conduct quantitative evaluations of the tracking performance on ScanNet [2] with NICE-SLAM [20] and Point-SLAM [11] as baselines. The results demonstrate that our proposed method outperforms both NICE-SLAM and Point-SLAM on ScanNet.

2. Related Work

In this section, we first present multiple previous works on dense visual SLAM and NeRF-based SLAM. Then, we proceed to introduce works applying the coarse-to-fine method that we incorporate.

2.1. Dense Visual SLAM

Recent studies in the field of visual SLAM have primarily employed the decoupled parallel structure of tracking and mapping, which was initially introduced in the seminal work PTAM [5]. Extending the sparse methodology of PTAM, DTAM [10] was devised to achieve dense visual SLAM in a similar decomposed approach.

In contrast, several studies have proposed an alternative strategy that directly optimizes depth maps [1, 19, 15]. In this way, optimization is conducted on a latent scene representation, which can be subsequently decoded into dense geometry. This approach alleviates the previously expensive computational cost and has subsequently enhanced the practicality of dense visual SLAM, particularly concerning real-time capability. Additionally, it can be regarded as a prototype for recent endeavors involving NeRF-based SLAM.

2.2. NeRF-based SLAM

In recent years, the Neural Radiance Field (NeRF) [8] has emerged as the most widely utilized method for implicit scene representations. NeRF utilizes a large Multi-Layer Perceptron (MLP) that takes a spatial location and a viewing direction as input and outputs corresponding volume density and radiance to implicitly represent the scene.

The view-dependent nature of NeRF has motivated preceding efforts to regress camera poses using an existing NeRF representation [18], which achieves a crucial prerequisite for dense visual SLAM. Additionally, a method for simultaneously fitting camera poses and a NeRF model was proposed in [6], bringing NeRF even closer to practical application in SLAM. However, the optimization process involving a single large MLP remained prohibitively slow for online SLAM.

IMap [14] was the first to achieve real-time dense visual SLAM using RGB-D input. Nonetheless, iMap still relies on a single MLP to represent the entire scene, thereby limiting its performance in large-scale environments. NICE-SLAM [20], an advancement over iMap, addresses this limitation by encoding the scene using a hierarchical feature grid and employing multi-level decoders to predict occupancy and colors. Instead of the grid-based representation, Point-SLAM [11] anchors neural implicit features of a scene in a point cloud similarly to Point-NeRF [17]. In this way, Point-SLAM dynamically distributes more points to regions with finer geometry or appearance.

2.3. Hierarchical Neural Implicit Representations

Despite the decent quality of novel view synthesis achieved by NeRF [8], its lengthy training time presents a major challenge limiting its practicality. Consequently, researchers have proposed several variants of NeRF to enhance its time- and memory-efficiency. A popular approach to achieving this involves the adoption of a hierarchical representation to facilitate optimization in a coarse-to-fine fashion. Notable methods that have embraced this strategy include NSVF [7], DirectVoxGO [16], and Plenoxels [3], all of which speeds up the original NeRF by orders of magnitude for the task of object-level novel view synthesis. Recently, Instant-NGP [9] further proposed a multi-resolution hash encoding that can be efficiently looked up at a computational cost of $O(1)$, which technically enabled real-time application of NeRF.

In a setting of SLAM, a hierarchical neural implicit scene representation is typically used not only for efficiency but also for its scalability and better convergence. This represents a significant improvement introduced by NICE-SLAM [20] over iMap [14]. In this project, we draw inspiration from NICE-SLAM and adopt a hierarchical point cloud representation to improve the tracking robustness of Point-SLAM [11] against local minima on real-world scenes.

3. Method

An overview of our method is given in Figure 2. Section 3.1 provides a comprehensive discussion of the pipeline structure and design specifics. Drawing inspiration from Point-SLAM[11], we employ neural point cloud as our scene representation. As new areas are explored, points are iteratively added to the scene based on the estimated camera pose. Two distinct levels of neural point clouds are generated to independently extract geometry and color features, as detailed in Section 3.2. To obtain color and depth values, volumetric rendering is utilized and briefly introduced

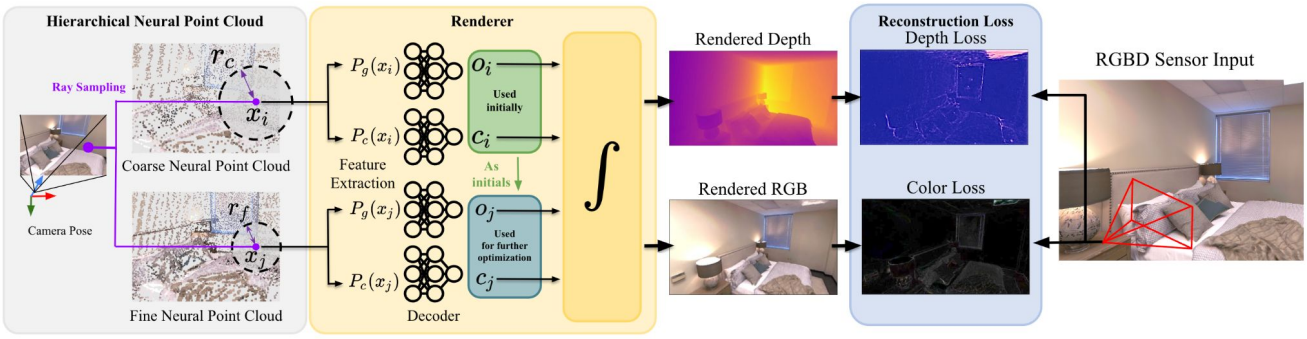

Figure 2: **Hierarchical Point-Slam Architecture.** Neural point clouds at both coarse and fine levels are derived from the predetermined estimated camera pose. Initially, a sparse array of neural points is introduced into the point cloud. Subsequently, we sample two distinct sets of points (x_i and x_j), each associated with a different search radius r. For each point, geometric and color features pertaining to two tiers of neural points are extracted ($P_g(x_i)$, $P_c(x_i)$, $P_g(x_j)$ and $P_c(x_j)$ resp.) and passed to the MLP decoder. This process facilitates the acquisition of their respective occupancies and colors (o_i, c_i, o_j and c_j resp.), which are then utilized to impose the depth and color re-rendering loss upon the input image, fostering optimization. This optimization is pursued in a coarse-to-fine sequence in both mapping and tracking phases. Following several iterations of optimization using coarse-level features, subsequent optimization is conducted using fine-level features, with the coarse level serving as initial conditions.

in Section 3.3. Parameters are refined following a coarse-to-fine strategy during the mapping and tracking phases, which is further elaborated in Section 3.4.

3.1. High-level Pipeline Structure Design

As mentioned before, coarse-to-fine method is a standard and common-use technique to prevent the tracker from being stuck at local minimum. To integrate this strategy to the original Point-SLAM[11] pipeline, we propose adding an additional processing level to the existing "fine-level" stage. This involves creating an extra coarse-level point cloud. Naturally, we should address the issues regarding ways to optimize neural features represented in the coarse-level point cloud and camera poses given both coarse and fine scene feature representations.

Before addressing these considerations, let's clarify the relationship between the two-level features. Inherited from NICE-SLAM[20], it is natural to view the fine-level features as a complementary residual to the coarse-level features. The core idea behind this design is to concatenate the two-level features and decode them using a fine-level decoder. However, in theory, the residual design primarily enhances scene detail reconstruction, which contradicts the assumption that fine details negatively impact tracking. Therefore, in our approach, we treat the two levels as independent scene representations, where neural features in the coarse and fine point clouds are optimized independently.

Given this independent setting, during the tracking stage, we first optimize the camera pose based on the coarse-level features for a certain number of iterations. Subsequently, we use the optimized coarse-level camera pose as an initialization for further optimization based on the fine-level features. Thus, we can effectively leverage the benefits of both coarse and fine-level features.

3.2. Hierarchical Point Cloud

We have now two levels of neural point clouds, which are defined as:

$$P^h = \{(p_i, f_i^g, f_i^c) | i = 1, ..., N\}, \tag{1}$$

where $h \in \{0, 1\}$ represents the coarse and fine levels of the neural point cloud. Each point is positioned at $p_i \in \mathbb{R}^3$ and has a geometric feature $f_i^g \in \mathbb{R}^{32}$ and a color feature $f_i^c \in \mathbb{R}^{32}$.

Points are added following a specific strategy. We sample points from pixels with the highest color gradient magnitude, considering an estimated camera pose. These points are then unprojected into 3D space, allowing us to search for neighbors within a radius r. If a sampled ray lacks at least 2 neighbors within the spherical radius r, we add 3 points along that ray. The value of r is determined using a dynamic resolution method based on the color gradient:

$$r^h(u,v) = \begin{cases} r_l^h & \text{if } \nabla I(u,v) \geq g_u \\ \beta_1 \nabla I(u,v) + \beta_2 & \text{if } g_l \leq \nabla I(u,v) \leq g_u \\ r_u^h & \text{if } \nabla I(u,v) \leq g_l \end{cases}, \tag{2}$$

$\nabla I(u, v)$ represents the gradient magnitude at the location (u, v). By increasing the search radius, we can explore a larger neighborhood and potentially find more neighboring

points. To generate two levels of the neural point cloud, we introduce different lower and upper bounds, denoted as $r^h(u, v)$. These bounds define the range within which we search for neighbors and determine the scale of the respective level of the point cloud.

3.3. Rendering

Volume rendering is a technique used to render color and depth information given sampled pixels. The approach applied here shares similarities with Point-NeRF [17] and is identical to the one used in Point-SLAM [11]. Below, we provide a brief explanation of how color and depth values are calculated.

Assuming we have the camera pose with origin O, the rendered color and depth values in the direction of a ray, denoted as $d \in \mathbb{R}^3$, are obtained by taking a weighted average of the values along the ray. Totally N points, $\{x_i\}_{i=1}^N$, are sampled given a specified ray direction. Eq. (3) presents formulas for computing these two variables.

$$\hat{D} = \sum_{i=1}^{N} \alpha_i z_i, \quad \hat{I} = \sum_{i=1}^{N} \alpha_i \mathbf{c}_i \tag{3}$$

Here, z_i represents the depth value of point x_i, and $\mathbf{c}_i$ is the color value obtained from the decoder using color features around point x_i. The termination probability of the ray at point x_i is denoted as α_i and is calculated using Eq.(4). In Eq. (4), the occupancy value $\mathrm{o}_{\mathbf{p}}$ is obtained from the decoder given geometry related features. The details of the MLP decoder are explained in [20].

$$\alpha_i = \mathrm{o}_{\mathbf{p}_i} \prod_{j=1}^{i-1} \left(1 - \mathrm{o}_{\mathbf{p}_j}\right) \tag{4}$$

The geometric depth variance along the ray which is used for loss normalization is defined as Eq. (5)

$$\hat{S}_D = \sum_{i=1}^{N} \alpha_i \left(\hat{D} - z_i\right)^2 \tag{5}$$

3.4. Mapping and Tracking

In this section, we will delve into the optimization of the scene geometry features, denoted as f^g, the color features, denoted as f^c, as well as the camera extrinsics represented by $\{R, t\}$. We will provide an overview of the mapping and tracking processes, with an emphasis on the parallel structure in mapping and a two-stage scheme applied in tracking. Besides that, we also briefly explain collaboration between mapping and tracking.

Mapping. During each mapping iteration, the mapper will firstly optimize the coarse-level scene representation, followed by the fine-level. Although implemented in a sequential way, they are actually independent processes. For a given level, the mapper starts by adding new points to the existing neural point cloud. Subsequently, optimization of the scene representation, including geometry and color features, takes place. The detailed strategy for adding points and expanding the scene can be found in Section 3.2.

To optimize the scene representation, a total of M pixels are uniformly rendered across the current frame and selected previous keyframes. This method enables us to effectively explore new areas, while maintain the integrity of captured information. The keyframe dataset is expanded at regular intervals, and the selected keyframe set at each step is determined based on their view overlaps with the current frame. Only those frames in the dataset having view overlapping with the current input will be randomly selected to compose the keyframe set, and only those points inside current frame's view frustum will be included in optimization.

The loss function for the mapping is a combination of an L_1 geometric depth loss and an L_1 color loss over the M sample pixels as shown in Eq. (6). The rendered results are denoted as $\hat{D}_m$ and $\hat{I}_m$, while the ground truth values are represented as D_m and I_m. λ_m is a weighting parameter. Initially, the optimization process focuses solely on the geometric depth loss and later integrates the color loss.

$$\mathcal{L}_{\text{map}} = \sum_{m=1}^{M} \left| D_m - \hat{D}_m \right|_1 + \lambda_m \left| I_m - \hat{I}_m \right|_1 \tag{6}$$

Tracking. During the tracking process, the optimization of the camera pose for each frame is carried out. In our approach, for every frame, the optimization process is divided into two stages.

In the first stage, during the initial 50% of the iterations, the camera pose is optimized based on the coarse-level scene features. This involves rendering depth and color values at sampled pixels using the coarse-level neural point cloud.

In the second stage, for the remaining iterations, the camera pose optimization is performed using the fine-level scene features. Here, the feature values at the sampled pixels are rendered based on the fine-level neural point cloud.

By employing this two-stage optimization approach, we are able to leverage the coarse-level features to provide a robust initialization for the subsequent optimization based on the fine-level features. This initialization helps improve the convergence and accuracy of the camera pose estimation process.

At the beginning of the whole tracking process, to eliminate redundant Degrees of Freedom (DoFs), it is assumed that the ground truth camera poses are available for the first and second frames. Consequently, the camera pose optimization process begins from the third frame onwards. To provide a good starting point for optimization at each frame, the camera pose for frame i is not optimized starting from

Method \ Scene	0025_02	0059_00	0062_00	0103_00	0106_00	0126_00	0181_00	0207_00	Avg.
NICE-SLAM [20]	10.11	14.00	4.59	**4.94**	**7.90**	21.80	13.40	6.20	10.37
Point-SLAM [11]	8.35	8.29	**3.48**	6.97	11.86	10.03	17.30	8.21	9.31
Ours	**6.13**	**7.06**	3.54	6.73	12.99	**6.57**	**11.08**	**4.53**	**7.33**

Table 1: **ScanNet Tracking** We report the ATE RMSE (↓ [cm]) from our method using a fixed seed. The results of NICE-SLAM [20] and Point-SLAM [11] are from the Point-SLAM [11] paper. The best results are highlighted as **first**, second, and third.

the identity matrix, but based on the camera poses from the previous frames with indices $i-1$ and $i-2$. A constant speed assumption is utilized and the initialization is calculated as follows:

$$\mathbf{T}_{\text{c2w\_ini}}[i] = \delta * \mathbf{T}_{\text{c2w\_predicted}}[i-1] \tag{7}$$

where δ is the relative transformation between the pose at frame $i-2$ and $i-1$.

During optimization, the camera pose is represented as a rotational quaternion with a translation vector. Totally M_t pixels are sampled across the input frame for calculation of color and depth values. Importance sampling is applied according to image gradient from input RGB image. The tracking loss shown as Eq. (8), is a combination of a normalized geometric loss and a color loss with weighting factor λ_t. The $\hat{S}_D$ calculate as Eq. (5) is the geometric depth variance.

$$\mathcal{L}_{\text{track}} = \sum_{m=1}^{M_t} \frac{\left|D_m - \hat{D}_m\right|_1}{\sqrt{\hat{S}_D}} + \lambda_t \left|I_m - \hat{I}_m\right|_1 \tag{8}$$

Collaboration between Mapping and Tracking. The mapping and tracking processes are not independent; they interact and exchange optimized scene geometry and camera poses with each other. Here, a clarification is provided on how they run interactively.

In this system, the mapper performs optimization every f frames, while the tracker performs optimization for every frame. When the tracker needs to optimize the camera pose for frame k, it checks whether mapping optimization should be performed on frame $k-1$. If mapping optimization is not required for frame $k-1$, the tracker proceeds with the camera pose optimization for frame k directly.

However, if mapping optimization is required for frame $k-1$, the tracker waits for the mapping process on frame $k-1$ to finish. Once the mapping optimization on frame $k-1$ is complete, the tracker updates the scene-related information obtained from the mapper.

Additionally, if mapping optimization is needed for frame k, the mapper waits for the tracker to finish tracking at this frame before starting the mapping optimization.

4. Experiments

We will first explain our experimental setup and then compare our method against the state-of-the-art dense neural RGBD SLAM methods on the real world ScanNet[2] dataset.

Parameter Configurations. We use dynamic search radius $r^1 = [2, 8](cm)$ for the fine level neural point cloud and $r^0 = [12, 48](cm)$ for the coarse level neural point cloud. In mapping, we use 300 mapping iterations for each level. In tracking, we use 50 tracking iterations for each level. Mapping is done for every 5 frames.

Evaluation Metrics. We evaluate the tracking accuracy using absolute trajectory error (ATE) RMSE [13]. It computes the translation differences between the estimated trajectory and the ground truth. Before calculating ATE RMSE, we align two trajectories using [4].

Dataset. Our motivation is to improve the tracking performance of Point-SLAM under real-world conditions, so we evaluate our method on the real-world ScanNet [2] dataset.

4.1. Tracking Results

Table 1 presents our tracking results on selected ScanNet scenes[1]. On average, our method exhibits the best tracking performance among existing methods. We believe that our coarse-to-fine optimization strategy reduces Point-SLAM's sensitivity to specularities and motion blur in real-world scenes. Additionally, optimizing the coarse-level point cloud provides a better initialization for the fine-level point cloud, resulting in a more robust tracking capability.

Figure 3 offers an intuitive evaluation of the reconstructed point clouds obtained from Point-SLAM and our proposed method in a selected ScanNet scene. Notably, in this particular scenario, Point-SLAM suffers from tracking failure, resulting in multiple erroneous reconstructions. In stark contrast, our method effectively addresses and mitigates such tracking drift, leading to highly accurate and robust reconstruction performance.

[1]As we have some problems in reproducing the scene0000_00 and scene0169_00 results for the Point-SLAM[11] pipeline, we removed these two scenes from results table. Once we've solved these issues, we will update results in our github repo.

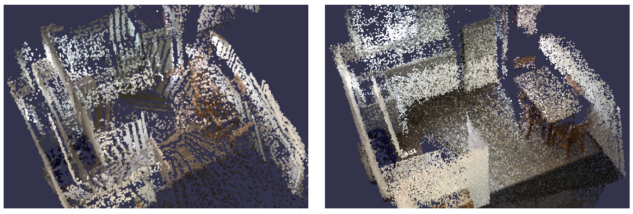

Figure 3: A comparison of the resulting neural point cloud taken from ScanNet scene 0181 at the frame 2438. **Left**: Point-SLAM, **Right**: our method.

4.2. Coarse Level Rendering Iterations Study

To reduce run time, we experimented with fewer mapping iterations in the coarse stage. We use the coarse-level point cloud to improve initialization for the fine level, reducing iterations in the coarse level may further decrease sensitivity to specularities. Table 2 shows the tracking performance using fewer mapping iterations on selected ScanNet [2] scenes. The iterations needed for the mapper to converge are scene dependent. In scene 0025 and scene 0059, the fluctuation is not large when the mapping iterations change. We believe it is because the mapper already converges on these scenes after 100 iterations. In scene 0207, the performance becomes better as we increase the number of iterations. We think it is because the mapper did not converge before 300 iterations, and the coarse level provided a worse initialization for the fine level before convergence. This makes the tracking results even worse than only using the fine level point cloud.

Iterations \ Scene	0025_02	0059_00	0207_00
Coarse 100 \ Fine 300	6.50	6.49	10.99
Coarse 200 \ Fine 300	7.33	**5.69**	8.76
Coarse 300 \ Fine 300	**6.13**	7.06	**4.53**

Table 2: **Tracking results with different mapping iterations.** The ATE RMSE (↓ [cm]) on selected ScanNet [2] scenes. The best results are highlighted as **first**, second, and third.

5. Discussion

Our coarse-to-fine optimization method effectively addresses a known limitation of Point-SLAM, specifically its sensitivity to specularities and motion blurs commonly encountered in real-world scenes. As a result, the tracking stability of our method is significantly improved. Furthermore, our approach preserves the high-quality rendering capabilities that are inherent to Point-SLAM. In scene 0106, our performance relative to Point-SLAM is subpar. We attribute this to either the mapper's failure to converge within 300 iterations or fluctuation at local minima.

Furthermore, our method demonstrates a negligible increase in memory usage. As an illustrative example, when applied to ScanNet scene 0060, our hierarchical Point-SLAM method requires a total of 5084MB for processing 83580 neural points, whereas Point-SLAM uses 4893MB for 63594 points. Consequently, the additional memory consumption amounts to only 191MB, which is deemed to be relatively low.

6. Conclusion

We developed a dense SLAM system with a hierarchical structure, building upon Point-SLAM [11]. Our approach incorporates a coarse-level point cloud for multi-stage optimization, effectively addressing local minima issues. Extensive experiments on the ScanNet dataset [2] demonstrate our method's competitive tracking performance compared to recent neural implicit SLAM systems.

7. Work Distribution and Other Clarifications

Our work distribution is shown in table 3. All group members have contributed equally to the project.

Task	**Members**
Literature study	All group members
Solution design	All group members
Hierarchical point cloud Implementation and multi-staged tracker	All group members
Residual pipeline implementation and experiments	Longteng Duan, Guo Han
Parallel pipeline implementation and experiments	Liuxin Qing, Shi Chen
Further experiments on parallel pipeline	Longteng Duan, Liuxin Qing
Reproduction of original Point-SLAM [11]	Guo Han, Shi Chen
Results analysis	All group members

Table 3: Work split of team members.

Regarding the code implementation, we have utilized the original Point-SLAM [11] pipeline as a foundation. Based on that, we have made several enhancements and implemented the following parts by ourselves. Firstly, we developed a hierarchical point cloud structure. Furthermore, we incorporated the optimization of the coarse-level point cloud within the mapper component, including the parallel structure and residual structure. In the tracker module, we implemented a two-stage camera pose optimization technique. To facilitate result monitoring and analysis of the hierarchical structure, we have expanded the visualization capabilities.

References

[1] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2560–2568, 2018.

[2] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[3] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[4] B. K. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, 1988.

[5] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 83–86. IEEE, 2009.

[6] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5741–5751, 2021.

[7] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.

[8] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[9] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[10] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[11] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald. Point-slam: Dense neural point cloud-based slam. *arXiv preprint arXiv:2304.04278*, 2023.

[12] T. Schops, T. Sattler, and M. Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 134–144, 2019.

[13] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. 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.

[14] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021.

[15] E. Sucar, K. Wada, and A. Davison. Nodeslam: Neural object descriptors for multi-view shape reconstruction. In *2020 International Conference on 3D Vision (3DV)*, pages 949–958. IEEE, 2020.

[16] C. Sun, M. Sun, and H.-T. Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022.

[17] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022.

[18] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin. inerf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330. IEEE, 2021.

[19] S. Zhi, M. Bloesch, S. Leutenegger, and A. J. Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11776–11785, 2019.

[20] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022.