AlphaTablets: A Generic Plane Representation for 3D Planar Reconstruction from Monocular Videos

Yuze He[1], **Wang Zhao**[1], **Shaohui Liu**[2], **Yubin Hu**[1],
Yushi Bai[1], **Yu-Hui Wen**[3], **Yong-Jin Liu**[1*]
[1]Tsinghua University [2] ETH Zurich [3] Beijing Jiaotong University

Abstract

We introduce AlphaTablets, a novel and generic representation of 3D planes that features continuous 3D surface and precise boundary delineation. By representing 3D planes as rectangles with alpha channels, AlphaTablets combine the advantages of current 2D and 3D plane representations, enabling accurate, consistent and flexible modeling of 3D planes. We derive differentiable rasterization on top of AlphaTablets to efficiently render 3D planes into images, and propose a novel bottom-up pipeline for 3D planar reconstruction from monocular videos. Starting with 2D superpixels and geometric cues from pre-trained models, we initialize 3D planes as AlphaTablets and optimize them via differentiable rendering. An effective merging scheme is introduced to facilitate the growth and refinement of AlphaTablets. Through iterative optimization and merging, we reconstruct complete and accurate 3D planes with solid surfaces and clear boundaries. Extensive experiments on the ScanNet dataset demonstrate state-of-the-art performance in 3D planar reconstruction, underscoring the great potential of AlphaTablets as a generic 3D plane representation for various applications. Project page is available at: `https://hyzcluster.github.io/alphatablets`.

1 Introduction

3D planar reconstruction from monocular videos is a crucial aspect of 3D computer vision, dedicated to the precise detection and reconstruction of underlying 3D planes from consecutive 2D imagery. The reconstructed 3D planes serve as a flexible representation of surfaces, facilitating various applications such as scene modeling, mixed reality and robotics.

Traditional methods for 3D planar reconstruction rely heavily on explicit geometric inputs [36, 41], hand-crafted features [6, 4], strong assumptions [10, 15] and solvers [37, 42, 16] to detect and reconstruct the planes, thereby imposing limitations on scalability and robustness. Learning-based methods [27, 26, 2, 28, 48] leverage the power of data-driven training to directly segment the plane instances and regress the plane parameters from single or sparse-view images. Notably, PlanarRecon [50], a pioneering monocular video-based learning method, operates within the 3D volume space to detect and track planes, and has demonstrated promising results in recovering planar structures. However, it often falls short in detecting complete planar reconstructions and struggles with generalization across diverse scenes. How to build an accurate, complete and generalizable 3d planar reconstruction system is still a challenging open problem.

We inspect this problem from the perspective of plane representation. Current methodologies employ various representations, such as view-based 2D masks [42, 16, 3, 9, 26, 2, 48], 3D points [37, 30], surfels [36], and voxels [50]. While 2D masks can precisely illustrate plane contours, this 2D plane representation faces inconsistencies across different views and necessitates complex matching and fusion processes to reconstruct 3D surfaces. In contrast, 3D representations directly depict 3D planar surfaces. However, they suffer from discontinous geometry and texture due to discretized sampling, and struggle to accurately model complex plane boundaries.

*Corresponding Author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

To address above limitations, we propose a novel plane representation termed AlphaTablets. AlphaTablets define 3D planes as rectangles with alpha channels, providing a natural delineation of irregular plane boundaries and enabling continuous solid 3D surface representation. AlphaTablets combine the best of 2D and 3D plane presentations: by defining and optimizing in 3D, AlphaTablets ensure efficiency and consistency across all views; meanwhile, by introducing alpha channels and texture maps in plane's canonical coordinates, AlphaTablets can accurately model solid surfaces and complex irregular plane boundaries. Furthermore, we introduce rasterization formulations for AlphaTablets, facilitating differentiable rendering into images.

Based on AlphaTablets, we present a bottom-up pipeline for 3D planar reconstruction from posed monocular videos. Initially, leveraging 2D superpixels [1], we initialize the AlphaTablets using pre-trained depth and surface normal models [19, 12]. This initialization yields a dense yet noisy assembly of relatively small and overlapping AlphaTablets. Next, these small planes are optimized via differentiable rendering with hybrid regularizers to adjust both geometry, texture and alpha channels. We further introduce an effective merging scheme that facilitates the fusion of neighboring tablets, thereby promoting the growth of larger, cohesive planes. Through iterative cycles of optimization and merging, our final reconstructions boast solid surfaces, clear boundaries, and interpolatable texture maps, delivering accurate 3D planar structures. Moreover, our approach enables flexible plane-based scene editing. Extensive experiments on ScanNet [11] dataset demonstrate state-of-the-art performance of 3D planar reconstruction, showing the great potential of being a generic 3D plane representation for subsequent applications.

In summary, our contributions are threefold:

- We propose AlphaTablets, a novel and generic 3D plane representation which features effective and flexible modeling of plane geometry, texture and boundaries. We derive the rasterization formulation for AlphaTablets, enabling differentiable rendering to images.
- We build an accurate and generalizable 3D planar reconstruction system from monocular videos upon AlphaTablets. The key components are effective initialization from pre-trained monocular cues, differentiable rendering based optimization, and the proposed merging mechanism for AlphaTablets.
- The proposed system achieves state-of-the-art performance for 3d planar reconstruction, while also enabling flexible plane-based scene editing for various applications.

2 Related Works

Classical 3D Plane Reconstruction. Traditional 3D plane reconstruction methods typically fit planes from geometric inputs like RGB-D images [36, 41, 35, 21] and 3D point cloud [7, 43], using robust estimators such as RANSAC [14] and its variants [37]. Another line of research approaches utilize multi-view images as input. Early works [42, 16, 15, 3] detect 3D planes from reconstructed sparse 3D points and lines, and then optimize plane masks as multi-label segmentation through image-based solvers such as Markov Random Fields (MRF) and Graph Cut [23]. Argiles et al. [3] propose a plane consistency check to determine plane boundaries using square cells rather than sparse point contours. Manhattan world assumption is introduced [10, 53] to better reconstruct the dominant planes. While these methods can segment 2D complex planes, the 2D mask representation of planes causes view inconsistency, visibility issues and requires additional efforts to reconstruct 3D planes. In contrast, AlphaTablets directly model and optimize planes in 3D with differentiable rendering, eliminating the inconsistency and occlusion problems. Given a monocular video as input, DPPTAM [9] reconstructs both the 3D planar and non-planar structure in a SLAM fashion, and adapt superpixel to group homogeneous pixels for textureless regions. Our method also uses superpixels as 2D units to initialize AlphaTablets, but provides greater flexibility in adjusting geometry, texture, and boundaries through optimization.

Learning-based 3D Plane Reconstruction. Data-driven methods leverage large-scale training data to learn geometric priors, enabling the reconstruction of 3D planes from single or sparse view images. PlaneNet [27], PlaneRecover [52] and PlaneRCNN [26] create training datasets for plane detection and reconstruction, and train deep neural networks to directly segment plane instances and regress plane parameters from single RGB image. Further enhancements along this line include the use of transformer architectures [47], multi-task collaboration [40], pairwise plane relations [34], and feature clustering [54], etc. Due to the highly ill-posed nature of single-image plane reconstruction, many

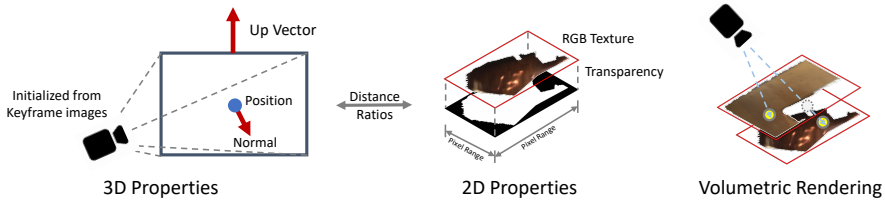


Figure 1: **Illustration of tablet properties and rendering.** Normal and up vector determines the rotation of a tablet in 3D space, while every tablet maintains a distance ratio between the coordinates of the 3D field and 2D-pixel space.

works [2, 22, 28, 48] adopt sparse view images as inputs, and explore joint plane detection, association and optimization to help the final reconstruction. However, these methods can only recover local 3D planar structures from sparse views, and it is challenging to extend them to image sequences. PlanarRecon [50] proposes a 3D volume-based planar reconstruction system with plane detection and tracking modules that sequentially process video frames and update 3D planar reconstructions. While effective in producing clean and consistent 3D planes, PlanarRecon often oversimplifies planar structures, resulting in incomplete reconstructions. Furthermore, being trained on indoor datasets with gravity-aligned camera poses, PlanarRecon struggles to generalize to unseen data. On the contrary, our method optimizes AlphaTablets for each scene, thus can generalize to any video sequence.

3D Planar Surface Representation. While most of current works employ 2D plane representation for detection and reconstruction, 3D representation is more efficient and consistent by aggregating 2D image information directly in 3D. 3D point cloud [37, 38, 33, 51, 32] is one of the most widely used 3D surface representation, yet they are discrete samples of a surface and cannot fully capture continuous geometry and textures. Extending points into planar primitives, surfels [36, 17] and 2D gaussians [18, 20] represent surface with 2D disks in 3D space, and demonstrate impressive results for both planar and non-planar surfaces. Unfortunately, both surfels and 2D gaussians only provide local planar structure within one unit, making it difficult to directly represent large geometric planes.

In terms of continuous surface representation, mesh is the most popular representation with mature graphic pipelines. Many works [50, 42, 49, 39] convert other 3D surface representations into mesh for visualization or further optimization. While meshes using 3D triangles or rectangles can represent 3D planes with regular quadrangular shapes, they struggle with irregular complex boundaries and are difficult to build and optimize from scratch. AlphaTablets inherit the advantages of continuous geometry and canonical texture modeling from mesh, and further introduce alpha channels to handle the irregular plane boundaries, acting as "rectangle soup with alpha channels". With the popularity of implicit neural representation, several works [25, 8] have explored encoding 3D plane primitives with MLPs. Compared to implicit representations, AlphaTablets offer the advantages of explicit representation, such as easy editing and fast rendering without network inference.

3 Method

Our proposed AlphaTablets is a novel and generic 3D plane representation, which enables accurate and generalizable 3D planar reconstruction from monocular video inputs. In Sec.3.1, we first introduce the data format of AlphaTablets, then discuss the differentiable rasterization of AlphaTablets in Sec.3.2. Finally, in Sec.3.3, we introduce a bottom-up pipeline based on AlphaTablets to conduct 3D planar reconstruction from monocular video input.

3.1 AlphaTablets: Representing 3D Planes with Semi-Transparent Rectangular Primitives

As illustrated in Figure 1, our proposed AlphaTablet is shaped as a rectangle with 3D geometry properties and 2D in-tablet properties. The 3D geometry includes the tablet center point p, normal vector $\mathbf{n}$, up vector $\mathbf{u}$ and right vector $\mathbf{r}$. The normal, up and right vectors are orthogonal. The 2D in-tablet information contains a texture map c, an alpha channel α, and a pixel range (ru, rv) that indicates the 2D resolution of the texture and transparency map. The alpha channel α ensures that arbitrary shapes can be modeled by our tablet formation. Since the ratio of unit distance in 3D space and 2D in-tablet canonical space varies across different tablets, distance ratios λ_u, λ_v of two directions on the texture map is also maintained for every tablet to acquire the tablet size in 3D space.

3.2 Differentiable Rasterization

As a generic 3D planar surface representation, efficient projection from 3D to 2D images is highly demanding for AlphaTablets. Therefore, we introduce the differentiable rasterization of AlphaTablets. The data format of AlphaTablets is the 3D rectangle soup with alpha channels, allowing us to easily adapt and utilize existing mesh-based efficient differentiable rendering frameworks, such as NVDiffrast [24] to composite and render an arbitrary number, shape, and position of tablets in a fully differentiable manner.

Pseudo Mesh Construction. We can leverage differentiable mesh rendering frameworks like NVDiffrast [24] by converting the tablets into pseudo meshes before each rendering pass. Note that this conversion is just used for the adaption of NVDiffrast. Given a single tablet t_i, we can convert it to two mesh triangle faces through the following process:

$$v_i = \begin{matrix}[p_i - ru_i/\lambda_{u,i} - rv_i/\lambda_{v,i},\ p_i - ru_i/\lambda_{u,i} + rv_i/\lambda_{v,i}, \\ p_i + ru_i/\lambda_{u,i} + rv_i/\lambda_{v,i},\ p_i + ru_i/\lambda_{u,i} - rv_i/\lambda_{v,i}]\end{matrix} \tag{1}$$

$$f_i = [[0, 1, 2], [0, 2, 3]] \tag{2}$$

Where v_i represents the 3D vertex coordinates, and f_i denotes the face indices, consistent with the general mesh definition. The texture and alpha maps of all tablets are tiled onto a global texture map according to their respective resolutions (ru_i, rv_i). The specific tile location serves as the texture coordinates for the four vertices of each tablet.

Multi-layer Rasterization. As transparency information is used in AlphaTablets, the rasterization process in our approach requires rasterizing multiple layers through depth peeling to extract multiple closest surfaces for each pixel. Given the model-view-projection (MVP) matrix M_k of a specific view k, the rasterization result of the l-th closest surface for the image pixel (i, j) can be formed as:

$$\mathcal{R}_l(M_k, i, j) = (u_{i,j}, v_{i,j}, tri_{i,j}) \tag{3}$$

where $u_{i,j}$ and $v_{i,j}$ are the barycentric coordinates within a triangle, and $tri_{i,j}$ is the triangle index. The color and alpha values $c(i, j)$ and $\alpha(i, j)$ are then acquired from the texture map using the two coordinates and the triangle index.

Anti-aliasing for AlphaTablets. Previous anti-aliasing techniques in rasterization predominantly work on non-transparent primitives without considering learnable alpha values of each face, yet alpha channel is a crucial component for AlphaTablets. An intuitive approach to incorporate alpha channels would be to conduct anti-aliasing for both texture and alpha within each rasterization layer. However, this straight-forward method does not work well on tablets with alpha channels. A simple counterexample is two overlapping planes with constant alpha values of 0 and 1, respectively. The rasterization results in two rasterized layers for the intersection pixel of the two planes. And the anti-aliasing would result in both layers having an alpha value below 1, causing incorrect transparency and colors at the intersected boundaries in the final alpha blending process.

To address this issue, we propose an anti-aliasing method for the semi-transparent primitives. Given a pixel through which the boundary lines of two planes pass, with the original colors c_1 and c_2, and alpha values a_1 and a_2, respectively, and anti-aliasing weights w and $1 - w$, the process to obtain the anti-aliased color c_{aa} is as follows:

$$c_{aa} = \frac{\alpha_1 c_1 w + \alpha_2 c_2 (1 - w)}{\alpha_1 w + \alpha_2 (1 - w)} \tag{4}$$

And the alpha value is not anti-aliased. The intuitive idea is that the blending weights of anti-aliasing should not only be determined by overlapping areas, but also take alpha values into consideration. For each rasterization layer, anti-aliasing can be further expressed using the following formula:

$$c_{aa} = \frac{AA(\alpha c)}{AA(\alpha)}, \quad \alpha_{aa} = \alpha \tag{5}$$

Where α and c are the alpha and color values before anti-aliasing, c_{aa} and α_{aa} are the alpha and color values after anti-aliasing, and AA is the anti-aliasing function. We show an example figure 9 in appendix to demonstrate the clear improvements after the anti-alias formula correction.

Alpha Composition. Once we have multiple rasterized layers, we can stack them in depth order and blend them using the alpha compositing process widely employed in volume rendering and neural

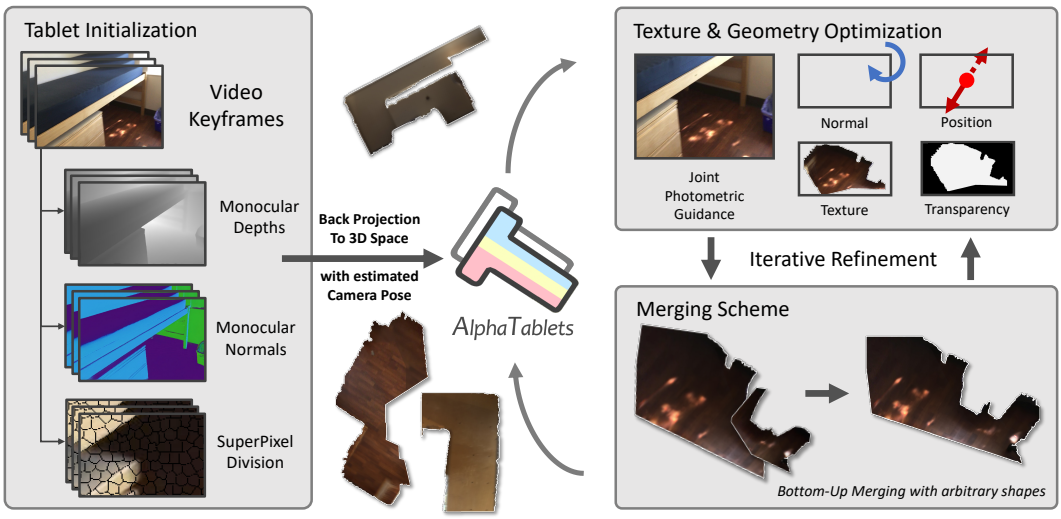


Figure 2: **Pipeline of our proposed 3D planar reconstruction.** Given a monocular video as input, we first initialize AlphaTablets using off-the-shelf superpixel, depth, and normal estimation models. The 3D AlphaTablets are then optimized through photometric guidance, followed by the merging scheme. This iterative process of optimization and merging refines the 3D AlphaTablets, resulting in accurate and complete 3D planar reconstruction.

radiance fields. The final color of pixel (i, j) can be calculated as:

$$c_{i,j} = \sum_{l=1}^{L} T_{i,j,l}\alpha_{i,j,l}c_{i,j,l}, \quad T_{i,j,l} = \prod_{m=1}^{l-1}(1-\alpha_{i,j,m}) \tag{6}$$

Where $c_{i,j,l}$ and $\alpha_{i,j,l}$ are the color and alpha values at pixel (i, j) of the l-th rasterization layer. $T_{i,j,l}$ represents the accumulated transmittance of the previous $l-1$ layers for the given pixel.

3.3 A Bottom-up Planar Reconstruction Pipeline with AlphaTablets

AlphaTablets provide flexible 3D plane modeling and efficient differentiable rendering. Building on this, we propose a bottom-up 3D planar reconstruction pipeline for monocular video input. As illustrated in Figure 2, AlphaTablets are first initialized via off-the-shelf geometric estimations, then the texture and geometry parameters of AlphaTablets are optimized using differentiable rendering. Tablets are examined and merged towards larger and more complete 3D planes. The optimization and merging benefit from each other through iterative refinement. By formulating the plane segmentation as a bottom-up clustering of 3D AlphaTablets and plane parameter estimation as rendering based optimization, our pipeline can accurately reconstruct detailed planar surfaces.

Initialization. We initialize the AlphaTablets using off-the-shelf geometric prediction models, including those for depth, surface normal and superpixel. Specifically, for each keyframe of the input video, we first apply SLIC [1] method to segment images into 2D superpixels based on local color homogeneity. Next, we utilize pretrained geometric models [19, 12] to estimate depth and surface normal for each image. To obtain initial depth and surface normal values for each superpixel, we perform 2D average pooling within each superpixel's region. These 2D superpixels are then back-projected into 3D space to form the initial tablet representation. Besides 3D geometry, the texture maps and alpha channels are initialized using 2D pixel colors and superpixel masks, respectively. The rectangle bounding box is determined by the minimum and maximum values on each 3D tablet's two orthogonal axes: up and right vector.

Optimization. Initialized from 2D view-based depth, surface normal and superpixel estimations, the initial 3D AlphaTablets may contain errors, overlaps, and inconsistency. We thus perform differentiable rendering based optimization to update the parameters of 3D AlphaTablets.

Learnable Parameters. While the 3D AlphaTablets offer significant flexibility, directly optimizing them for unrestricted movement in 3D space can cause instability. To mitigate this, we constrain each tablet's center to remain on its initial camera ray. In this way, we thus optimize the normal vector $\mathbf{n}$

and distance d, rather than 3D center position p, where d represents the distance between the tablet's center and the camera center of the view from which it was initialized. The up vectors of tablets are updated with normal to keep the rigid transformation characteristics of the tablet. Additionally, the texture and alpha channel values of each tablet are treated as learnable parameters, enabling appearance refinement to enhance the fidelity of the reconstructed planar surfaces.

Loss Design. The optimization process is driven by a set of carefully designed loss functions that collectively refine the tablet parameters to achieve accurate planar reconstruction. Given input monocular video, we adopt the photometric loss as the mean squared error between the rendered image c of the tablets and the observed input images c_{gt}: $\mathcal{L}_{pho} = ||c - c_{gt}||_2^2$. By minimizing the photometric loss, the 3D AlphaTablets can be optimized to better fit the input images. However, due to the ambiguity of photometric alignment and the complexity of optimization, updating AlphaTablets only with photometric loss results in fuzzy reconstructions. We thus introduce several important losses to help mitigate the ambiguity and regularize the reconstruction.

Specifically, we use alpha inverse loss to prevent the emergence of semi-transparent regions after alpha composition, which is defined as $\mathcal{L}_{ainv} = \prod_{l=1}^{L}(1 - \alpha_l)$. Moreover, we observed that multiple semi-transparency tablets, instead of one solid tablet, may occupy the same surface region to blend the rendering, which harms the geometric surface reconstruction. Inspired by mip-NeRF-360 [5], we utilize the distortion loss for AlphaTablets to penalize the multiple semi-transparency surfaces and promote the merging of tablets that are in close proximity. The distortion loss is defined as below:

$$\mathcal{L}_{dist} = \sum_{i=1}^{L-1} T_i T_{i+1} ||p_i - p_{i+1}||_2 \tag{7}$$

Where T_i is the blending weight of the i-th rasterization layer defined in Sec. 3.2, p_i is the 3D intersection point of the i-th rasterization layer. To further regularize the surface geometry and smoothness, we render the tablets to get the depths d_r and surface normal maps $\mathbf{n}_r$, and supervise them by the input monocular depth and surface normal estimations $d_m, \mathbf{n}_m$ with mean squared error:

$$d_r = \sum_{l=1}^{L} T_l \alpha_l d_l, \quad \mathbf{n}_r = \sum_{l=1}^{L} T_l \alpha_l \mathbf{n}_l \tag{8}$$

$$d = \frac{d_r}{\prod_{l=1}^{L}(1 - \alpha_l)}, \quad \mathbf{n} = \frac{\mathbf{n}_r}{||\mathbf{n}_r||_2} \tag{9}$$

$$\mathcal{L}_{depth} = ||d - d_m||_2^2, \quad \mathcal{L}_{normal} = ||\mathbf{n} - \mathbf{n}_m||_2^2 \tag{10}$$

The final objective is defined as:

$$\mathcal{L} = w_1 \mathcal{L}_{pho} + w_2 \mathcal{L}_{ainv} + w_3 \mathcal{L}_{dist} + w_4 \mathcal{L}_{depth} + w_5 \mathcal{L}_{normal} \tag{11}$$

where w_1, w_2, w_3, w_4, w_5 are hyperparameters to balance the losses.

Merging Scheme. The optimized 3D AlphaTablets are still describing local 3D planar surface, bounded by the 2D superpixels. Therefore, to represent the exact 3D planes, we need to coherently merge the individual tablets into larger tablets. We introduce a hierarchical merging strategy that considers hybrid information including color, distance, and normal, to prevent the wrong merging.

Specifically, we first construct a KD-tree to find the tablet neighborhoods, and initialize a union-find data structure for all tablets. Each union-find set dynamically maintains the average color, center, and normal of all its constituent unit tablets. For each tablet, we search the KD-tree to find the K nearest tablets and evaluate whether they satisfy the following constraints for merging: (1) The angle between the normals of the two tablets should be smaller than a threshold θ. (2) The angle between the average normals of the union-find sets to which the two tablets belong should be smaller than a threshold θ_s. (3) The projected distance between the average centers of the union-find sets along their average normal should be smaller than a threshold d. (4) The difference between the average colors of the union-find sets should be smaller than a threshold c.

If all these constraints are satisfied, the two tablets are merged into the same union-find set. We repeat this process for all tablets, continually updating the average color, center, and normal of each union-find set as merges occur. This iterative merging procedure continues until no further merges are possible, resulting in a set of coherent planar surfaces represented by the final merged tablets. More details about merging are included in the appendix.

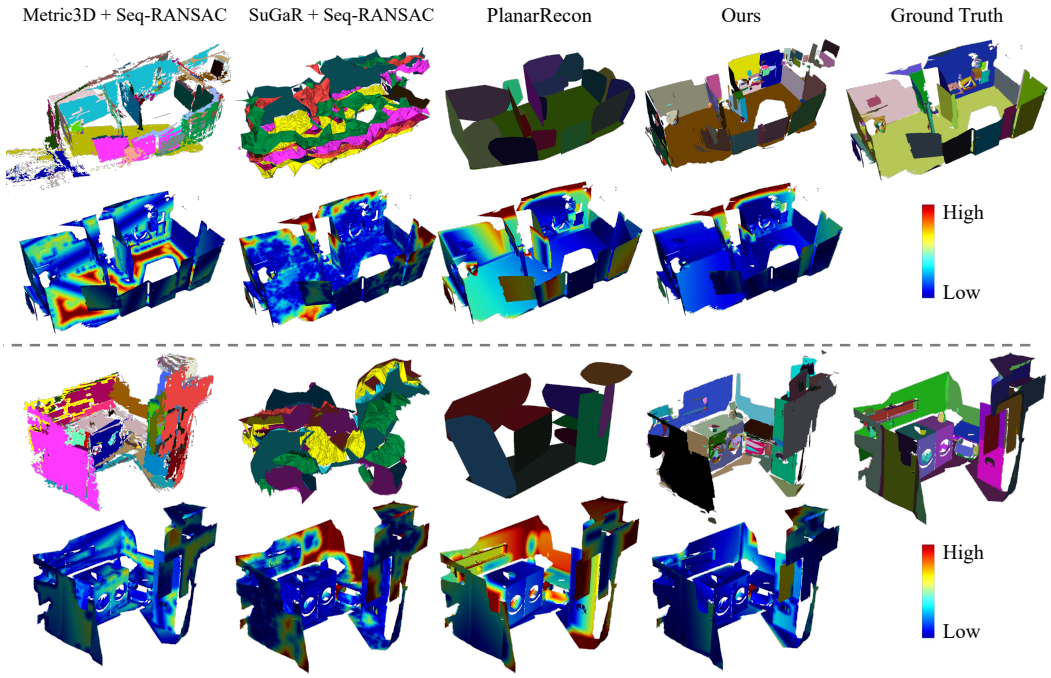


Figure 3: **Qualitative results on ScanNet.** Error maps are included. Better viewed when zoomed in.

Table 1: **3D geometry reconstruction results on ScanNet.**

Method	Comp ↓	Acc ↓	Recall ↑	Prec ↑	F-Score ↑
NeuralRecon [46] + Seq-RANSAC	0.144	0.128	0.296	0.306	0.296
Atlas [31] + Seq-RANSAC	0.102	0.190	0.316	0.348	0.331
ESTDepth [29] + PEAC [13]	0.174	0.135	0.289	0.335	0.304
PlanarRecon [50]	0.154	**0.105**	0.355	0.398	0.372
Metric3D [19] + Seq-RANSAC	**0.074**	0.379	0.426	0.161	0.231
SuGaR [18] + Seq-RANSAC	0.121	0.324	0.385	0.296	0.327
Ours	0.108	0.161	**0.481**	**0.447**	**0.456**

4 Experiments

4.1 Evaluation on 3D Plane Detection and Reconstruction

Implementation Details. We use Metric3Dv2 [19] for predicting monocular depths and Omnidata [12] for surface normals. We leverage the keyframe selection method in NeuralRecon [46] to segment the scene into multiple parts. Each part undergoes separate optimization, followed by joint optimizations. The keyframe number of each part is set to 9. The separate optimization for each part is performed for 32 epochs, while the joint optimization step is executed for 9 epochs. The weights for the loss functions are set as follows: $[w_1, w_2, w_3, w_4, w_5] = [1.0, 1.0, 20.0, 4.0, 4.0]$. We use Adam optimizer, and the learning rates for the tablet's texture, alpha, normal, and distance are set to 0.01, 0.03, 1e-4, and 5e-4, respectively. After the second merge step, the learning rate for the distance is reduced to 2e-4. The normal threshold is set to 0.93. The entire reconstruction process for a single scene takes around 2 hours on average when executed on a single A100 GPU.

Dataset and Evaluation Metrics. We use ScanNetv2 [11] dataset to evaluate the 3D plane detection and reconstruction performance of our proposed method. Following PlanarRecon [50], our method is tested on the validation set of Atlas [31] using generated 3D plane ground truth. For evaluation metrics, we follow previous works [50, 26, 54] to use Murez's 3D metrics [31] for geometry, and rand index (RI), variation of information (VOI), segmentation covering (SC) as plane segmentation metrics. To assess the segmentation performance, the reconstructed plane instances are transferred onto the ground truth planes using the nearest neighborhood approach, following common practices.

Table 2: **3D plane segmentation results on ScanNet.**

Method	VOI ↓	RI ↑	SC ↑
NeuralRecon [46] + Seq-RANSAC	8.087	0.828	0.066
Atlas [31] + Seq-RANSAC	8.485	0.838	0.057
ESTDepth [29] + PEAC [13]	4.470	0.877	0.163
PlanarRecon [50]	3.622	0.897	0.248
Metric3D [19] + Seq-RANSAC	4.648	0.862	0.209
SuGaR [18] + Seq-RANSAC	5.558	0.775	0.082
Ours	**3.468**	**0.928**	**0.273**

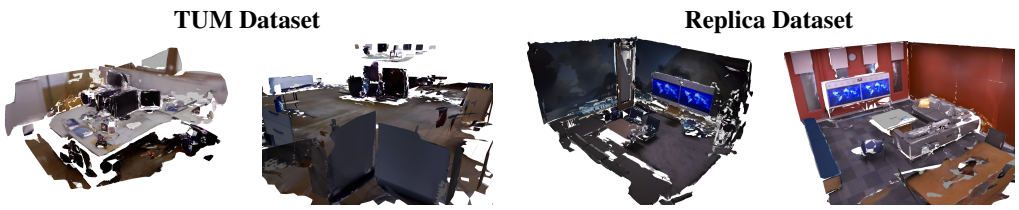


Figure 4: **Qualitative results on TUM-RGBD and Replica datasets.**

Baselines. We compare our method with different types of representative works. PlanarRecon [50] is the state-of-the-art method of learning-based 3D planar reconstruction from monocular video. Following it, we compare with strong baselines that first reconstruct the 3D scene and then fit 3D planes using RANSAC, including single or multi-view depth-based methods [29, 19], 3D volume-based methods [46, 31], and the recent 3D gaussian based method [18].

Quantitative Results. The evaluation results for 3D plane geometry and segmentation are presented in Table 1 and Table 2, respectively. Our proposed method achieves clear improvements compared to other state-of-the-art approaches across various evaluation metrics. 3D volume-based reconstruct-then-fit methods suffer from reconstruction errors and noise threshold-sensitive RANSAC, and exhibit relatively low performance in terms of planes' geometry and 3D coherence. Depth-based methods inherently encounter 3D inconsistency, resulting in fragmented and multi-layered predictions. PlanarRecon [50], which is specially trained on the ScanNet dataset, demonstrates the capability to predict major planes with high geometric accuracy. However, its performance is hindered by the limited coverage of the predicted planes and the failure to detect many small plane instances. Approaches based on 2D Gaussian Splatting [18] tend to be heavily influenced by the poor initialization, textureless regions and motion blurs in ScanNet, resulting in degraded reconstruction performance. Compared to other methods, our approach demonstrates much improved overall performance for both geometry and segmentation.

Qualitative Results. To provide a qualitative assessment of our method's performance, we follow PlanarRecon [50] and present the plane reconstruction results in Figure 3, along with the recall error maps. The SuGaR+Seq-RANSAC method suffers from erroneous geometric reconstructions, and the Metric3D+Seq-RANSAC is constrained by inconsistent fuzzy points and sub-optimal plane segmentations. PlanarRecon, while capable of reconstructing large planar surfaces with high geometric accuracy, struggles to capture and reconstruct smaller plane instances, resulting in incomplete representations of the scene. Our method benefits from the bottom-up planar reconstruction scheme, and can accurately predict the 3D plane instances while excelling in detecting and reconstructing details, particularly for smaller plane instances. This capability significantly outperforms other methods in handling fine-grained planar structures. To demonstrate the generalization ability of our method, we further test it on TUM-RGBD [45] and Replica [44] datasets. Qualitative results are shown in Figure 4. Our method can faithfully reconstruct 3D planar surfaces in various scenarios.

4.2 Ablation Studies

To validate the efficacy of our method's design, we conducted a series of ablation experiments exploring the impact of various components, including the loss functions, merge schemes, and tablet anti-aliasing. The results are presented in Table 3. Tablet distortion loss encourages the planar surfaces to converge and merge, leading to improved performance. Furthermore, the normal loss and depth loss significantly contribute to the geometric accuracy of the reconstructed planes, particularly

Table 3: **Ablation studies.** *AlphaInv* denotes the alpha inverse loss.

Method	F-score ↑	VOI ↓	RI ↑	SC ↑
Only Photometric and *AlphaInv* loss	0.240	4.096	0.936	0.191
+ Tablet Distortion loss	0.271	3.741	0.937	0.253
+ Normal loss	0.425	3.490	**0.944**	0.263
+ Depth loss	**0.456**	**3.466**	**0.944**	**0.284**
w/o tablet anti-aliasing	0.415	3.545	0.937	0.280
w/o tablet merge	0.188	6.991	0.939	0.098
Full	**0.456**	**3.466**	**0.944**	**0.284**

Figure 5: **3D scene editing examples of our method.**

in textureless regions where photometric loss constraints are insufficient. Merging scheme is crucial for producing appropriate 3D planes. Without merging, the 3D AlphaTablets remain small plane fragments, and thus can not reconstruct 3D planes, as shown in Table 3. Moreover, tablet antialiasing contributes to smoother results, leading to enhanced overall performance.

4.3 Example Application: 3D Plane-based Scene Editing

One of the significant advantages of AlphaTablets representation is its ability to perform consistent 3D scene editing by simply modifying the 2D texture maps associated with the reconstructed planes. As illustrated in Fig. 5, our method can achieve impressive results for editing 3D scenes. The accurate plane reconstruction allows for precise texture mapping, enabling the seamless application of textures, text, or other visual elements onto the planar regions within the scene. Furthermore, our method offers the flexibility to modify the color or perform style transfer on individual planes, providing a powerful tool for creative scene manipulation.

4.4 Limitations and Future Work

AlphaTablets effectively represent 3D planes, but it may struggle in highly non-planar regions where the planar assumption for a single superpixel does not hold. Additionally, the current AlphaTablets representation does not account for view-dependent effects. As a result, the optimization relies on color consistency across views, which can be compromised by non-Lambertian surfaces or changes in lighting. In the future, we aim to enhance AlphaTablets with view-dependent modeling, and explore hybrid scene representation such as AlphaTablets with Gaussians.

5 Conclusion

In this work, we introduce AlphaTablets, a novel and versatile 3D plane representation. AlphaTablets use rectangles with alpha channels to represent 3D planes, allowing for flexible and effective arbitrary 3D plane modeling. We derive a differentiable rasterization process for AlphaTablets to enable efficient 3D-to-2D rendering. Building on this, we propose a novel bottom-up 3D planar reconstruction pipeline from monocular video input. Leveraging the AlphaTablets representation, along with carefully designed optimization and merging schemes, our pipeline can reconstruct highly accurate and complete 3D planar surfaces in a generalizable manner. Experiments on the ScanNet dataset demonstrate significant improvements over baseline methods, highlighting the potential of AlphaTablets as a general representation for 3D planar surfaces.

Acknowledgement

This work was supported by Beijing Science and Technology plan project (Z231100005923029), the Natural Science Foundation of China (Project Number 62332019) and Beijing Natural Science Foundation (L222008).

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] Samir Agarwala, Linyi Jin, Chris Rockwell, and David F Fouhey. Planeformers: From sparse view planes to 3d reconstruction. In *European Conference on Computer Vision*, pages 192–209. Springer, 2022.

[3] Alberto Argiles, Javier Civera, and Luis Montesano. Dense multi-planar scene estimation from a sparse set of images. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4448–4454. IEEE, 2011.

[4] Caroline Baillard and Andrew Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 559–565. IEEE, 1999.

[5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[6] Adrien Bartoli. A random sampling strategy for piecewise planar scene segmentation. *Computer Vision and Image Understanding*, 105(1):42–59, 2007.

[7] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2):1–13, 2011.

[8] Zheng Chen, Qingan Yan, Huangying Zhan, Changjiang Cai, Xiangyu Xu, Yuzhong Huang, Weihan Wang, Ziyue Feng, Lantao Liu, and Yi Xu. Planarnerf: Online learning of planar primitives with neural radiance fields. *arXiv preprint arXiv:2401.00871*, 2023.

[9] Alejo Concha and Javier Civera. Dpptam: Dense piecewise planar tracking and mapping from a monocular sequence. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5686–5693. IEEE, 2015.

[10] Alejo Concha, Muhammad Wajahat Hussain, Luis Montano, and Javier Civera. Manhattan and piecewise-planar constraints for dense monocular mapping. In *Robotics: Science and systems*, 2014.

[11] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias 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.

[12] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796, 2021.

[13] Chen Feng, Yuichi Taguchi, and Vineet R Kamat. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6218–6225. IEEE, 2014.

[14] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[15] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Manhattan-world stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1422–1429. IEEE, 2009.

[16] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1418–1425. IEEE, 2010.

[17] Yiming Gao, Yan-Pei Cao, and Ying Shan. Surfelnerf: Neural surfel radiance fields for online photorealistic reconstruction of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 108–118, 2023.

[18] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023.

[19] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *arXiv preprint arXiv:2404.15506*, 2024.

[20] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. *arXiv preprint arXiv:2403.17888*, 2024.

[21] Jingwei Huang, Angela Dai, Leonidas J Guibas, and Matthias Nießner. 3dlite: towards commodity 3d scanning for content creation. *ACM Trans. Graph.*, 36(6):203–1, 2017.

[22] Linyi Jin, Shengyi Qian, Andrew Owens, and David F Fouhey. Planar surface reconstruction from sparse views. In *Proc. of the IEEE/CVF International Conference on Computer Vision*, pages 12991–13000, 2021.

[23] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *IEEE transactions on pattern analysis and machine intelligence*, 26(2):147–159, 2004.

[24] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (ToG)*, 39(6):1–14, 2020.

[25] Zhi-Hao Lin, Wei-Chiu Ma, Hao-Yu Hsu, Yu-Chiang Frank Wang, and Shenlong Wang. Neurmips: Neural mixture of planar experts for view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15702–15712, 2022.

[26] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. Planercnn: 3d plane detection and reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4459, 2019.

[27] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588, 2018.

[28] Jiachen Liu, Pan Ji, Nitin Bansal, Changjiang Cai, Qingan Yan, Xiaolei Huang, and Yi Xu. Planemvs: 3d plane reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8665–8675, 2022.

[29] Xiaoxiao Long, Lingjie Liu, Wei Li, Christian Theobalt, and Wenping Wang. Multi-view depth estimation using epipolar spatio-temporal networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8258–8267, 2021.

[30] Hannes Möls, Kailai Li, and Uwe D Hanebeck. Highly parallelizable plane extraction for organized point clouds using spherical convex hulls. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7920–7926. IEEE, 2020.

[31] Zak Murez, Tarrence Van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3d scene reconstruction from posed images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 414–431. Springer, 2020.

[32] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[33] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[34] Yiming Qian and Yasutaka Furukawa. Learning pairwise inter-plane relations for piecewise planar reconstruction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 330–345. Springer, 2020.

[35] Carolina Raposo, Miguel Lourenço, Michel Antunes, and Joao Pedro Barreto. Plane-based odometry using an rgb-d camera. In *BMVC*, volume 2, page 6, 2013.

[36] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. Dense planar slam. In *2014 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 157–164. IEEE, 2014.

[37] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[38] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[39] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.

[40] Jingjia Shi, Shuaifeng Zhi, and Kai Xu. Planerectr: Unified query learning for 3d plane recovery from a single view. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9377–9386, 2023.

[41] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, pages 746–760. Springer, 2012.

[42] Sudipta Sinha, Drew Steedly, and Rick Szeliski. Piecewise planar stereo for image-based rendering. In *2009 International Conference on Computer Vision*, pages 1881–1888, 2009.

[43] Christiane Sommer, Yumin Sun, Leonidas Guibas, Daniel Cremers, and Tolga Birdal. From planes to corners: Multi-purpose primitive detection in unorganized 3d point clouds. *IEEE Robotics and Automation Letters*, 5(2):1764–1771, 2020.

[44] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[45] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[46] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15598–15607, 2021.

[47] Bin Tan, Nan Xue, Song Bai, Tianfu Wu, and Gui-Song Xia. Planetr: Structure-guided transformers for 3d plane recovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages

4186–4195, 2021.

[48] Bin Tan, Nan Xue, Tianfu Wu, and Gui-Song Xia. Nope-sac: Neural one-plane ransac for sparse-view planar 3d reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[49] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[50] Yiming Xie, Matheus Gadelha, Fengting Yang, Xiaowei Zhou, and Huaizu Jiang. Planarrecon: Real-time 3d plane detection and reconstruction from posed monocular videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6219–6228, 2022.

[51] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich 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.

[52] Fengting Yang and Zihan Zhou. Recovering 3d planes from a single image via convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[53] Shichao Yang and Sebastian Scherer. Monocular object and plane slam in structured environments. *IEEE Robotics and Automation Letters*, 4(4):3145–3152, 2019.

[54] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1029–1037, 2019.

A Appendix

A.1 More Details of AlphaTablets Optimization

Update of up vector. Note that the tablet's up vector should not be learned during the optimization process. By considering the tablet's motion as a rigid transformation, any in-plane rotation can be accounted for by optimizing the texture and alpha values. However, we need to establish an update rule for the up vector to keep the rigid transformation characteristics of the tablet. Our design is to apply the same rotation to the up vector as the one applied to the normal vector during the update. Given the normal vectors n and n' (before and after the update), and the up vector u, we can acquire the new up vector u' by:

$$\theta_r = \mathbf{n} \cdot \mathbf{n}', \tag{12}$$

$$\mathbf{r} = \mathbf{n} \times \mathbf{n}', \tag{13}$$

$$K = \begin{pmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0, \end{pmatrix} \tag{14}$$

$$R = I \cos\theta_r + K \sin\theta_r + \mathbf{r}\mathbf{r}^T(1 - \cos\theta_r), \tag{15}$$

$$\mathbf{u}' = R\mathbf{u} \tag{16}$$

Merging Scheme. As explained in Section 3.3, we first construct a KD-tree to find the tablet neighborhoods, and initialize a union-find data structure for all tablets. Here we actually use the unit tablets, which are defined as the projections of all initial 3D tablets to the current tablets, to build the KD-tree. In other words, we maintain the affiliations of initial and current tablets, and use the updated initial tablets to perform the merging. The reason is that using 3D center point distances among tablets with different 3D sizes is ambiguous. For example, a small tablet can have a smaller center point distance than a non-adjacent small tablet, compared to a spatially adjacent big tablet. Using the unit tablets with similar sizes, the neighboring adjacency can be easily determined by checking the center point distances.

These unit tablets are used only because they have easily defined neighborhoods. Since the unit tablets of one current tablet come from the projection on this tablet, they share the same surface normal and adjacent positions. Thus, merging with unit tablets will definitely produce larger (or the same) tablets than current tablets. After each merging, all the unit tablets are updated by the projection onto the merged new tablets, and the merged new tablets are fed into the next optimization.

Using SLIC superpixel on ScanNet 1296x968 resolution image results in around 10k superpixels for each keyframe, leading to a large number of initial tablets. To address the issue, We conduct an initial merge process after AlphaTablets initialization. In practice, we find it is beneficial to the accuracy and convergence speed. Table 5 shows the ablations of the initial merge.

Table 4: **Ablation studies on initial merge.**

Method	F-score ↑	VOI ↓	RI ↑	SC ↑
w/o in-training merge and init merge	0.188	6.991	0.939	0.098
w/o in-training merge	0.438	5.171	0.941	0.138
w/o init merge	0.454	3.754	**0.944**	0.273
w/ all merge schemes	**0.456**	**3.466**	**0.944**	**0.284**

Weight Check Scheme. During the optimization process, there may be cases where some tablets are nearly invisible from all viewpoints, yet they have a relatively large transparency value. In such situations, these tablets should be removed. Additionally, there could be instances where certain regions of a tablet are not visible from any viewpoint. In these cases, those specific regions of the tablet should be excluded.

To address these scenarios, we designed a weight check mechanism: We perform a rasterization step at all viewpoints and extract the points where the alpha blending weight exceeds a certain threshold (we choose 0.3 in our implementation). We record the tablet index corresponding to each of these points. Before the merging step, we perform the weight check by removing tablets with an excessively

low number of associated points. Furthermore, for each tablet, we recalculate its boundary based on the texture map coordinate ranges of all the points associated with that tablet.

Tablet-camera assignment. We always maintain affiliations between the initial tablets and the current (merged) tablets (as stated in Sec A.1), and we keep track of the camera index that initially generated each initial tablet. When tablets are merged, we count the number of each camera index corresponding to all affiliated initial tablets and assign the most frequently occurring camera to the newly merged tablet.

A.2 Additional Implementation Details

Baselines. For 3D volume-based methods including Atlas, NeuralRecon, PlanarRecon, and Metric3D with TSDF fusion, we followed PlanarRecon to use their enhanced version of Seq-RANSAC. We refer to PlanarRecon for detailed descriptions. For point-based methods such as SuGaR, since PlanarRecon's Seq-RANSAC requires 3D TSDF volume as inputs and cannot be easily adapted to points or meshes, we use the classical vanilla Seq-RANSAC, which iteratively applies RANSAC to fit planes. Here we used Open3D plane RANSAC implementation for each iteration. The hyper-parameters are carefully tuned for optimal performance. For the Metric3D baseline, We used the official Metric3D v2 implementation and pre-trained weights (v2-g) running on each keyframe to get depth maps, followed by TSDF fusion to fuse into 3D volume. Finally, PlanarRecon's Seq-RANSAC is applied to the 3D TSDF volume to get the planar results. We adopted the original implementation for the SuGaR baseline, during the COLMAP pre-processing, we feed ground-truth camera poses into the pipeline, which provides better initial sparse points. After optimization, SuGaR outputs the mesh model, and we uniformly sampled 100k surface points and applied vanilla Seq-RANSAC on top of sampled points to get the 3D planar results. Quantitative results for other baselines (Atlas, NeuralRecon, PEAC, PlanarRecon) were taken from PlanerRecon.

Details of tablet properties. For pixel range (r_u, r_v), each tablet's geometry is located in 3D space, while its texture is stored in 2D. The pixel range represents the resolution at which the texture is stored: it is derived directly from the range in the source image for initial tablets; for merged tablets, the pixel range is calculated as the average of all corresponding initial tablets. The distance ratios (λ_u, λ_v) establish the relationship between the 2D texture resolution and the 3D size of the tablet. For initial tablets, the distance ratio is calculated by dividing the camera's focal length by the average initial distance of the tablet. For merged tablets, the distance ratio is the average of all corresponding initial tablets' ratios. The alpha channel of tablets is a learnable single-channel map with the same shape as the texture map.

A.3 Additional Discussions

Different initialization for SuGaR baseline. We further experiment on our ablation subset to compare the COLMAP initialization with Metric3D's dense depth-based initialization similar to our method. For Metric3D init, we use the same keyframes as our method and randomly sample a total of 100,000 points as initial points. The results are shown in the table:

Table 5: **Ablation studies on different initialization of SuGaR.**

Method	F-score ↑	VOI ↓	RI ↑	SC ↑
SuGaR+COLMAP Initialization	0.300	5.759	0.797	0.090
SuGaR+Metric3D Initialization	0.326	5.670	0.789	0.102
Ours	**0.456**	**3.466**	**0.944**	**0.284**

The ScanNet dataset presents significant challenges like numerous blurry and textureless regions, which are especially problematic for Gaussian-based methods like SuGaR when reconstructing clear geometry. Also, SuGaR heavily relies on COLMAP reconstruction to initialize, but the COLMAP reconstruction on ScanNet is sometimes noisy, affecting the final performance. The Metric3D initialization method does indeed enhance the reconstruction quality of SuGaR (as shown in Fig. 6), but the overall reconstruction quality remains constrained, with noticeable jitter and challenges in accurately delineating planar regions, leading to an inferior performance to our approach.

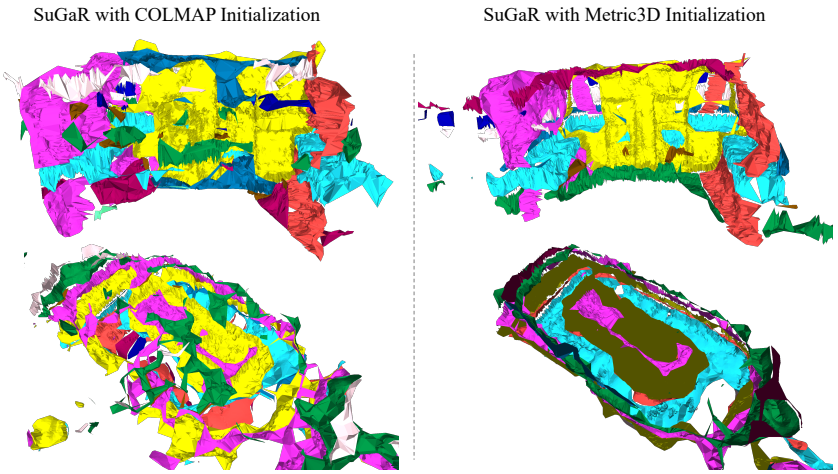


Figure 6: **Qualitative Comparison of Initialization Methods for SuGaR.**

Breakdown of Time Budget. Below is a breakdown of the time budget for the optimization process of a single scene:

Table 6: **Breakdown of the time budget of a single scene.**

Stage	Task	Time (s)
Initialization	texture init	**1517.38**
	geometry init	**1672.57**
Render	pseudo mesh	10.39
	rasterization	316.62
	alpha composition	2.15
Loss Calculation	photometric loss	1.07
	depth loss	28.28
	normal loss	102.44
	distortion loss	5.90
Training	backward	**3347.83**
Merge	kd-tree,union-find set	96.41
	geometric calculation	23.14
	tablet projection	22.26
	weight check	62.14

The merge and rendering pipeline is relatively efficient, while the initialization process (which includes converting every superpixel to an initial tablet, and texture initialization) consumes a significant amount of time. This is due to the current naive demonstration implementation, where tens of thousands of Python loops are called, which can be improved to enable parallelized initialization in future work. Furthermore, the NVDiffRast renders more than ten layers to perform alpha composition every forward pass, but most of the scene's structure is single-layered, resulting in a substantial backward computation burden during training. We regard this as another potential area for considerable optimization in the future work.

3D reconstruction accuracy. The difference in 3D accuracy (termed as Acc in Tab. 1 of the main paper) between our method and PlanarRecon on the ScanNet dataset can be attributed to several factors. First is the scope of reconstruction: PlanarRecon often only reconstructs large planar regions. This allows for easier localization and high accuracy in these specific areas, but it limits overall coverage and performance. Our method enables more comprehensive reconstruction, including smaller planar regions, which can impact the accuracy metrics but provide a more complete representation of the scene. Another is the ground-truth coverage: It is worth noting that the 3D ground truth planes in ScanNet only partially cover the scene within the camera's view. Even after

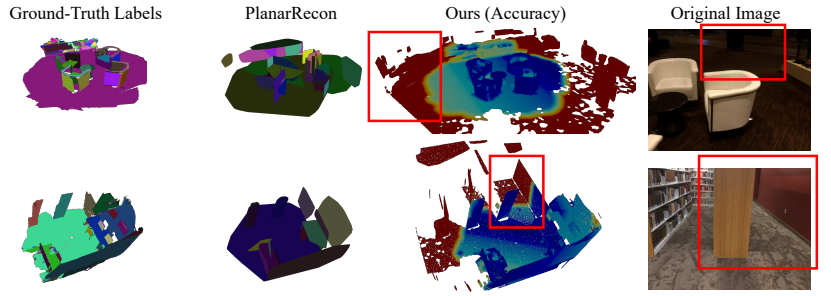


Figure 7: **Demonstration of Insufficient Coverage of 3D Ground-Truth Labels**: The 3D ground truth labels only partially cover the range within the camera's view. Most of the red regions in the figure highlight this issue. While these uncovered areas reduce accuracy, they should not be considered a negative outcome.

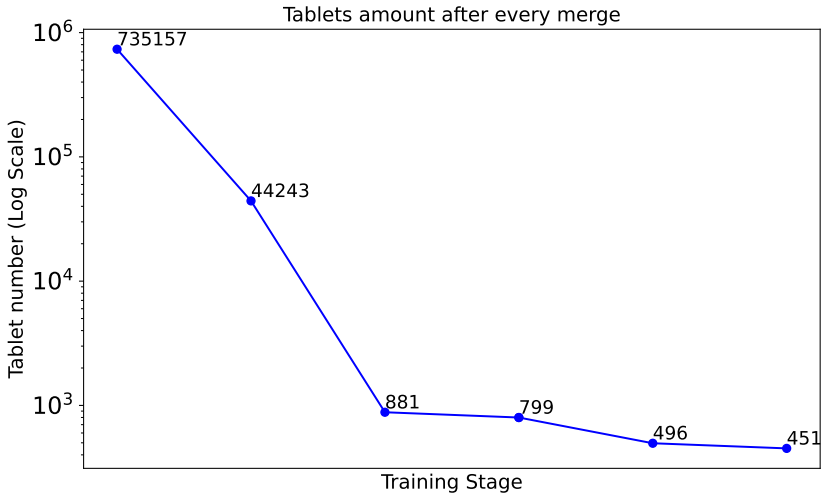


Figure 8: **Visualization of Tablet Count Evolution.**

excluding areas too distant to be relevant using the camera frustum, significant portions remain uncovered. PlanarRecon learns to exclude distant reconstructions during its training stage, leading to improved accuracy metrics. Our method, however, is capable of identifying planar regions for all visible areas (evident in Fig. 7 where most of the red regions highlight this phenomenon). While these uncovered areas affect the evaluation accuracy, they should not necessarily be considered a negative outcome. Our method provides a more complete reconstruction of the scene, including areas not represented in the ground truth data.

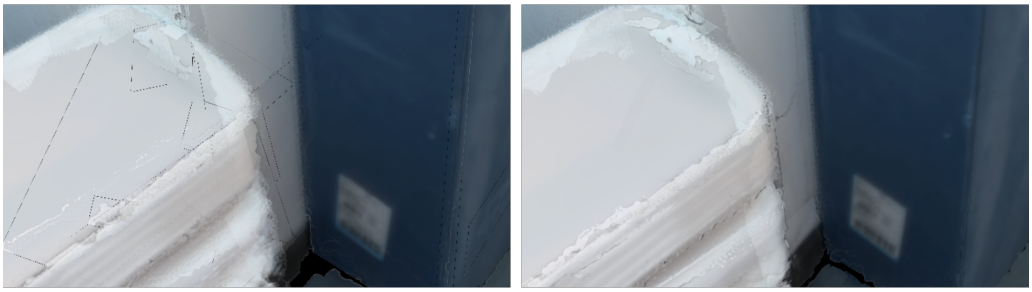
Tablet count evolution. We demonstrate the tablet count evolution of a single scene in Fig. 8. The number of tablets decreases rapidly in the early merging stages and gradually converges into several hundred. Notably, the final tablets contain a large portion of small tablets representing non-planar regions, while the primary planar scene structure is adequately represented with fewer tablets.

A.4 Additional Qualitative Results

We provide more qualitative results in Fig. 10 and Fig. 11.

A.5 Broader Impacts

3D planar reconstruction and editing have the potential to revolutionize numerous fields such as entertainment, media, accessibility, manufacturing, etc, by enhancing visualization, interaction, and



Naive Anti-aliasing Our Tablet Anti-aliasing

Figure 9: **Qualitative comparison of our tablet anti-aliasing scheme.** Naive anti-aliasing will lead to wrong strip artifacts, while our anti-aliasing scheme effectively mitigates those artifacts.

understanding. However, it may raise concerns about privacy and data security, necessitating robust policies and safeguards.

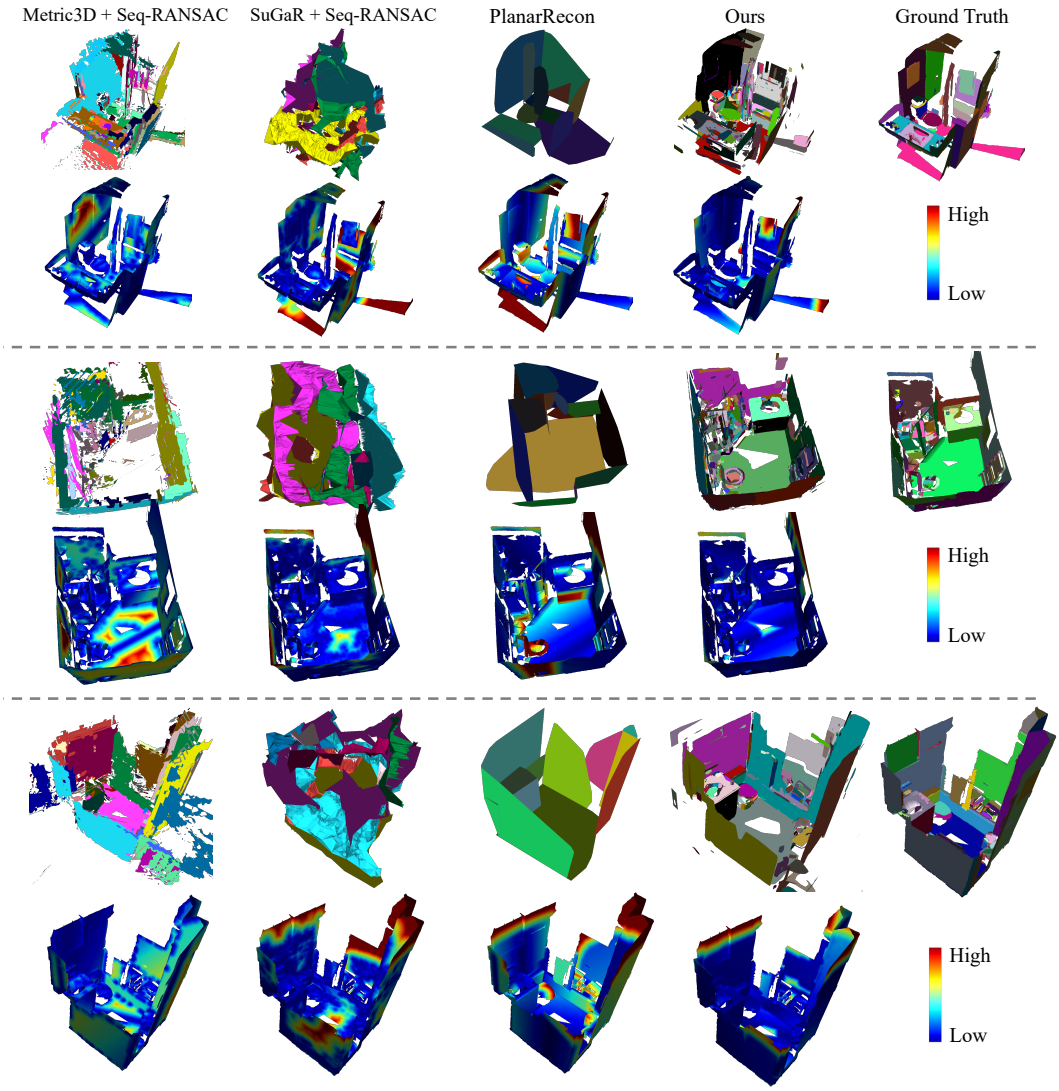


Figure 10: **More qualitative results on ScanNet.** Error maps are included. Better viewed when zoomed in.

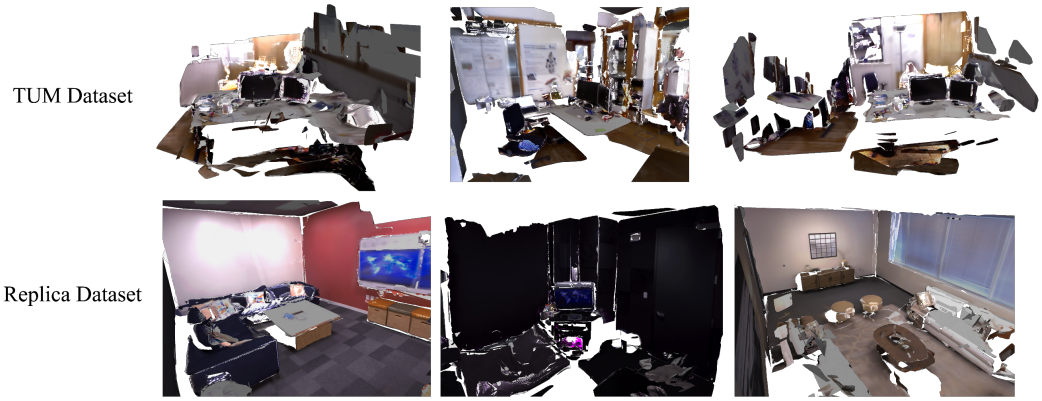


Figure 11: **More qualitative results on TUM-RGBD and Replica datasets.**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 4.4 for limitation discussions.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not include or claim theoretical contributions or results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 4.1 for implementation details about hyperparameters, optimization settings, etc.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Our code will be released once the paper is public.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 4.1 for implementation details about hyperparameters, optimization settings, etc.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our evaluation is deterministic and too costly to calculate the statistical significance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section 4.1 for running time and resources.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research in this paper conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section A.5 for the broader impact discussion.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The original papers that produced the code package or dataset are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.