Volumetric Bundle Adjustment for Online Photorealistic Scene Capture

Ronald Clark
Imperial College London
ronald.clark@imperial.ac.uk

Abstract

Efficient photorealistic scene capture is a challenging task. Current online reconstruction systems can operate very efficiently, but images generated from the models captured by these systems are often not photorealistic. Recent approaches based on neural volume rendering can render novel views at high fidelity, but they often require a long time to train, making them impractical for applications that require real-time scene capture. In this paper, we propose a system that can reconstruct photorealistic models of complex scenes in an efficient manner. Our system processes images online, i.e. it can obtain a good quality estimate of both the scene geometry and appearance at roughly the same rate the video is captured. To achieve the efficiency, we propose a hierarchical feature volume using VDB grids. This representation is memory efficient and allows for fast querying of the scene information. Secondly, we introduce a novel optimization technique that improves the efficiency of the bundle adjustment which allows our system to converge to the target camera poses and scene geometry much faster. Experiments on real-world scenes show that our method outperforms existing systems in terms of efficiency and capture quality. To the best of our knowledge, this is the first method that can achieve online photorealistic scene capture.

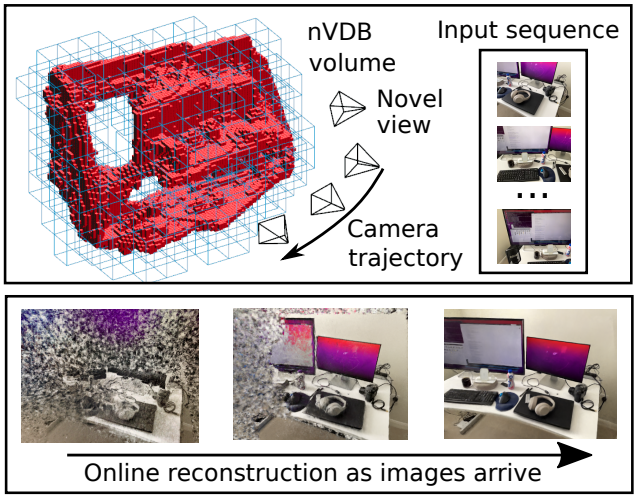

Figure 1. **Our method in operation.** We present an approach for online scene capture from a stream of monocular RGB images. The scene is represented using a neural volumetric dynamic B+Tree (nVDB) which stores a hierarchy of spatial features. The dynamic topology of the tree allows it to grow as the camera explores the scene. The camera and the volume parameters are optimized online using a novel volumetric bundle adjustment method (VBA). This allows our method to efficiently capture scene appearance and geometry. See the supplementary video for an online demo.

1. Introduction

Many applications require accurate online estimation of the geometry and appearance of 3D scenes. In augmented reality, for example, accurate geometry is necessary for proper handling of occlusions and physics interactions, while accurate appearance is necessary for rendering convincing novel views.

There are two main paradigms for achieving this goal. The first paradigm consists of visual simultaneous localization and mapping (SLAM) systems which are able to construct a dense model of a scene. These methods typically obtain depth maps either from a depth sensor, in the case of RGB-D methods [13, 15], or by estimating depth keyframes from multiple RGB images, in the case of monocular SLAM systems [11, 11, 14, 17]. These depth measurements are then fused into a consistent model in the form of a voxel grid representing a signed distance function (SDF) to surfaces which encodes the geometry of the scene. The colors from the RGB images are projected to the volume to model the appearance. These approaches are very efficient and operate in real time[1]. However, their main disadvantage is that they make simplified assumptions about scene appearance and can only reconstruct solid surfaces.

[1]For the purposes of this paper we define *real time* to mean the processing time for the reconstruction is comparable to the time taken to capture the image sequence. We define *online* to mean allowing for incremental processing as images arrive from the sensor.

Therefore they cannot capture photorealistic models of real-world scenes which are needed for real-time applications.

The second paradigm which has recently become very popular consists of neural volumetric rendering approaches [9,10]. These methods model the scene as a volume and use raymarching to render novel views. They estimate the volume parameters by directly optimizing photo-metric consistency using stochastic gradient descent (SGD) or momentum optimizers such as Adam [6]. The volume rendering, combined with the direct optimization of photometric error, allows these approaches to capture scenes in a photorealistic manner. However, the main disadvantage of these approaches is that they require a long per-scene training time. In the case of NeRF [10] this is up to a few days due to the neural network scene parameterisation.

In this paper, we aim to achieve photorealistic scene capture with high efficiency to get closer to real-time performance. This is accomplished though two novel components. The first is a tree-based scene representation that improves memory efficiency because information is only stored near occupied areas. The tree is very efficient to query which speeds up rendering and therefore also optimization. The second component is our volumetric bundle adjustment (VBA) approach that optimizes the pose and volume parameters to achieve photo consistency. VBA is based on a Levenberg-Marquadt type approach which enables much faster convergence by utilizing the curvature of the objective function. This allows it to traverse valleys of the objective function more quickly. By combining the nVDB scene representation and the VBA optimizer our system achieves more accurate results than RGB-D systems.

To summarize, the main contribution of in this paper is a system for online photorealistic reconstruction. Key to this system are two novel components:

1. A neural volumetric dynamic B+Tree, called nVDB, that can efficiently represent 3D scenes and can grow as more areas of the scene are explored

2. An efficient method for optimizing the volume properties, called volumetric bundle adjustment (VBA), that given a stream of input images can estimate the optimal nVDB parameters

2. Related work

RGB-D SLAM systems: These methods [2,13,20] typically take in RGB-D images from a depth sensor such as the Kinect or iPhone LiDAR sensor, estimate the camera pose using dense image alignment and fuse these into a volumetric representation of the scene. The volumetric representation typically consists of a dense voxel grid that stores geometry as the truncated signed distance value (TSDF) to the nearest surface and per-voxel color values that represent the scene appearance. These methods do not optimize the photoconsistency of the texture and are limited to representing solid Lambertian surfaces. Therefore, they cannot capture photorealistic models of complex scenes. Colormap optimization methods [7,29] aim to produce more photorealistic models by optimizing the camera poses along with non-rigid corrections to the images to ensure that the textures on the model are photometrically consistent. As in our method, the parameters are jointly optimized to maximize photometric consistency using non-linear least squares. However, unlike our method, they do not optimize the geometry.

Monocular reconstruction systems: There are many approaches [4, 11, 14, 18, 23] that have been designed to reconstruct scene geometry using only RGB images as input. The main difference between these methods and the RGB-D systems described above is that they have an extra step that reconstructs depth keyframes from multiple RGB images. This is typically done by constructing a photometric cost-volume and extracting per-pixel depth values that minimize this cost [14]. These methods suffer from photometric ambiguities present in textureless areas, for example as they rely on photometric information to compute the geometry. Recent learning based methods [4,11,18,23] have helped to make monocular reconstruction more robust, however, they generally do not model complex appearance effects and thus do not produce photorealistic models.

Neural volumetric rendering: Recent works [9, 10] have shown that it is possible to use neural networks in combination with volumetric rendering to capture and render photorealistic novel views of a scene. NeRF [10] uses a multi-layer perceptron (MLP) to model the volume properties at each point in space. Raymarching is then used to render images from arbitrary viewpoints. These methods are able to capture scenes with complex appearance as the volume rendering allows for modeling non-solid surfaces and the MLP can be conditioned on viewing angle to capture view-dependent effects. However, one of the main limitations of using a neural network to represent the volume is that it takes very long to render new frames. This makes both training and rendering very slow. NSVF [8] moved the representational capacity of the scene away from the neural network to a sparse voxel octree which can be queried very efficiently. The features stored in the SVO are processed by an MLP that outputs the volume rendering properties. However, the training of the NSVF model is still slow as the model makes use of a large MLP (similar to that of NeRF) and is trained using stochastic gradient descent. In our method, we also use an efficient tree structure for representing spatial feature of the scene. However, we use a hierarchical tree structure that allows us to store multi-scale features along with a smaller MLP. Furthermore, we propose a second-order optimizer that allows us to estimate the volume parameters more efficiently.

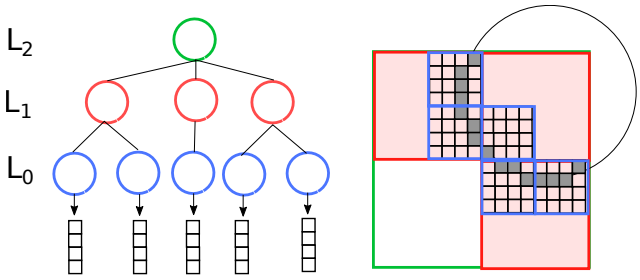

Figure 2. **Example of a VDB grid [12]**. This example shows a 2D scene consisting of a circle. VDB grids are based on B+Trees and allow for a different branching factor at each level of the tree and a variable number of children per node. This example shows a tree with a 2^1 branching at L_2 and L_1 and a 2^4 branching factor at L_0.

View synthesis methods: Local view synthesis methods [1,24,28] learn a general network to predict the volume rendering parameters from one or a few images near the target view. These methods require no optimization when capturing new scenes, and thus can quickly synthesize novel views. However, they only construct local representation typically using 3-5 images near the target view and thus do not build global representation of the scene. They are therefore best suited only for synthesizing novel views. In contrast, our method efficiently captures a global model of the scene and can render photorealistic novel views.

3. Background: VDB Trees

In general, representing a scene using a dense voxel grid is not memory efficient as many elements tend to be zero thereby wasting a lot of memory. Tree-based representations are a good solution as they allow storing only occupied voxels. Sparse voxel octree's (SVO) are a popular choice and have been used for RGB-D SLAM systems [22] and for speeding up neural volume rendering [8, 19, 27]. However, a major limitation of SVO's is that, in order to maintain the spatial resolution, the depth of the tree has to grow with the extent of the scene. VDB trees [12] were designed to overcome these limitations. VDB trees also represent voxels as the leaf nodes of an acyclic graph, but unlike octrees, have a variable branching factor at each level, which keeps the tree shallow. The configuration of the VDB tree is specified as $[B_1, ..., B_D]$ where B_d denotes the log_2 of the branching factor at level L_d of the tree. Figure 2 shows a 2D example of a $[4, 1, 1]$ VDB tree. The leaf nodes (also called "blocks") split into $2^4 = 16$ voxel elements and upper two layers split into at most 2^1 nodes. The variable branching factor means that not all these nodes at a particular level need to be active.

The data stored in the voxel elements is usually a scalar value representing the signed distance to the closest surface (for solid surface rendering) or a 4D opacity + colour value for volume rendering. However, the VDB tree can store data or features of arbitrary dimensionality. In addition, although VDB trees typically store voxel data at the leaf-nodes, data can be stored at any of the non-leaf (internal) nodes. Storing data at multiple levels of the tree allows one to create a multi-scale representations of a scene. The structure of the VDB tree makes it possible to efficiently access voxel values of arbitrarily complex scenes at multiple scales in constant, i.e. $O(1)$, time.

4. Method

In this section, we present our approach for reconstructing a photorealistic representation of a scene in near real time. Our system takes as input a sequence of images I_i, a rough estimate of the depth D_i at each frame and camera poses: $\mathrm{T}_i = \begin{bmatrix} \mathrm{R}_i & \mathbf{t}_i \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{SE}_3$, where $i = 1, \ldots, M$. Our system efficiently constructs a dense volumetric representation of the scene from which photorealistic novel views can be rendered. The two key components of our approach are a new volumetric representation that combines a dynamic sparse voxel grid with a novel bundle adjustment method called volumetric bundle adjustment, that is able to efficiently find the optimal parameters for the camera poses and volumetric representation.

An overview of our approach is shown in Figure 3. The base representation of the scene consists of a dynamic feature volume which learns spatial features representing the scene appearance that are projected to the volumetric rendering parameters using a shallow network (Section 4.1). The scene is rendered using raymarching to produce a novel view of the scene (Section 4.2). The rendered images are compared to the observed images to form a nonlinear least squares cost function which is optimized using a novel volumetric bundle adjustment approach (Section 4.3).

4.1. Neural VDB representation

We construct a novel continuous 3D representation of a scene that maps each point and viewing direction to a color and opacity value, $F_\theta : (V(\boldsymbol{x}), \boldsymbol{v}) \rightarrow (\boldsymbol{c}, \sigma), \forall \boldsymbol{x} \in V$. These values are queried during the raymarching to render novel views of the scene from a particular viewpoint. Our representation combines a VDB tree with a neural network interpolator, which we call an nVDB tree. The base of the nVDB is a VDB tree that stores N_f^L dimensional features at each level, L, along the tree hierarchy. We use the nVDB tree to learn an explicit hierarchy of features that represent the scene at each point in space. This is represented by the map $V : \boldsymbol{x} \rightarrow \boldsymbol{f}$ which takes a point $\boldsymbol{x}$ and queries the VDB grid to produce one feature for each level of the tree at that point in space. Similar to [8], the sampling is done using trilinear interpolation $\chi(\cdot)$ of the 8 neighbours of the voxel in which the point $\boldsymbol{x}$ lies, i.e.

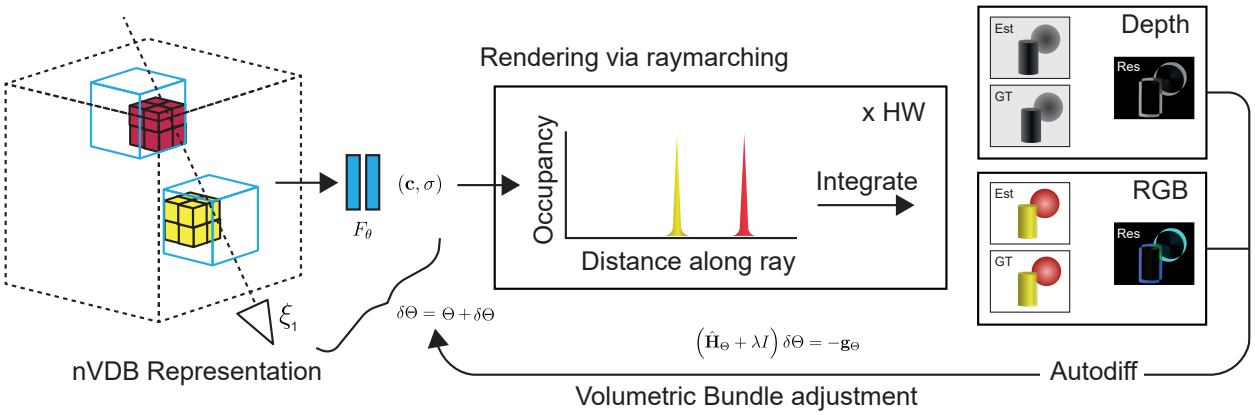

Figure 3. **Overview of our method**. We represent the scene as a neural dynamic volumetric B+Tree (nVDB) that stores spatial scene features. These features encode the shape and appearance of the scene in a high-dimensional feature space. Raymarching is used to render views of the volume specified by a camera pose. During rendering the features are sampled from the volume using trilinear interpolation and a shallow MLP is used to project these features to color and occupancy values. The estimated pixel color is computed by integrating the color values along the ray weighted by the occupancy and visibility. This is repeated for each pixel in the image. Once all the rays are rendered, the residuals are computed as the difference between the estimated and ground truth images. The residuals are then minimized using volumetric bundle adjustment (VBA), which efficiently refines the volume and camera pose parameters.

$V(\boldsymbol{x}) = \chi\left(\widetilde{V}_i\left(\boldsymbol{x}_1^*\right), \ldots, \widetilde{V}_i\left(\boldsymbol{x}_8^*\right)\right)$. The features $\boldsymbol{f}^l$ sampled at each level of the tree are then concatenated to form a feature vector $\boldsymbol{f}$ of dimension $\sum_{l=1}^{L} N_f^l$. This concatenation of features at each level of the tree produces a multi-scale feature representation that makes it easier to learn consistent appearance across large entities in the scene. The function F_θ, projects the features sampled from the VDB grid to the color and opacity outputs. This projection is modelled using a shallow fully-connected neural network.

4.2. Volume Rendering

We use volume rendering to synthesize views of the scene from a particular viewpoint given by the camera pose, T_j. For each pixel in the target image, we generate the corresponding ray with $\boldsymbol{x}_o$ and direction $\boldsymbol{v}$. The color of the pixel corresponding to the ray is computed by raymarching through the volume. The raymarching starts at the ray origin and steps along the ray, sampling the volume at each distances, $\boldsymbol{z}_i$, along the ray. The colors are accumulated along the ray $\bar{c}_t = \sum_k \alpha_k \left(1 - \exp\left(-\sigma_k\right)\right) c_k$ weighted by the visibility, $\alpha_k = \exp\left(-\sum_{j=1}^{k-1} \sigma_j\right)$. The output of the raymarching is a single color value for the pixel. In addition, we raymarch an estimated depth image. The depth are accumulated along the ray $\bar{d}_t = \sum_k \alpha_k \left(1 - \exp\left(-\sigma_k\right)\right) \boldsymbol{z}_k$ where α_k is computed as above. This depth image is used for the depth supervision, allowing us to utilize RGB-D data from sensors such as the Kinect, ARKit or from a monocular depth prediction network.

4.3. Volumetric Bundle Adjustment

Bundle adjustment iteratively adjusts the camera poses and the scene structure parameters by minimizing the photometric error between the rendered and captured images. Traditionally, the scene structure parameters optimized in bundle adjustment are sparse 3D points [21,25]. In our case, the structure parameters correspond to the parameters of our volumetric representation of the scene which consists of the features stored in the volume and the shallow MLP network. For purposes of optimization, we use a minimal representation of the camera parameters $\xi_j = (\log(R_j), \boldsymbol{t}_j)$ where log is the logarithmic map. The aggregated structure parameters consisting of the MLP parameters and voxel features as described in Section 4.1 are denoted by θ. We denote the set of all parameters by $\Theta = (\xi, \theta)$. The least squares objective function that we minimize takes the form:

$$\min_{\Theta \in \mathbb{R}^m} L(\Theta) = \frac{1}{2} \sum_{i=1}^{n} \left(\bar{c}_i - y_i\right)^2 + \left(\bar{d}_i - d_i\right)^2, \quad (1)$$

where $\boldsymbol{y}_i$ is the target color of that ray and d_i is the target depth. We define $\boldsymbol{r} = \left[\bar{c}_i - y_i,\ \bar{d}_i - d_i\right]_{i=1}^{n}$ as the residual vector formed by stacking the individual color and depth residuals. We minimize this objective function using the Gauss-Newton method. Defining the Jacobian $\mathbf{J}_{\boldsymbol{a}}$ as $[\mathbf{J}_{\boldsymbol{a}}\boldsymbol{r}]_{ij} = \partial[\boldsymbol{r}]_i / \partial[\boldsymbol{a}]_j$, we obtain the gradient of this objective function as: $\boldsymbol{g}_\Theta = \nabla_\Theta L(x; \Theta) = \mathbf{J}_\Theta^T \boldsymbol{r}$. The Gauss-Newton approximation of the Hessian takes the form: $\hat{\mathbf{H}}_\Theta \approx \mathbf{J}_\Theta^T \mathbf{J}_\Theta$. The parameter update is then computed by solving the following linear system:

$$\left(\hat{\mathbf{H}}_\Theta + \lambda I\right) \delta\Theta = -\boldsymbol{g}_\Theta. \quad (2)$$

This update takes into account both the gradient and curvature of the objective function. As $\hat{H}_\theta$ is very large, solving this equation directly is only feasible for very few parameters. However, our problem has a unique structure that can be exploited to allow for a more efficient solution. As in [25], we split the Hessian and the gradient into components that are governed by the camera variables ξ and the structure variables θ:

$$\mathbf{H}_\Theta = \begin{bmatrix} \mathbf{H}_{\xi\xi} & \mathbf{H}_{\xi\theta} \\ \mathbf{H}_{\xi\theta}^T & \mathbf{H}_{\theta\theta} \end{bmatrix}, \quad \boldsymbol{g} = \begin{bmatrix} \boldsymbol{g}_\xi \\ \boldsymbol{g}_\theta \end{bmatrix}. \tag{3}$$

We now use the nonlinear variable projection method (see [3] for an overview of its use in bundle adjustment) and convert the original update into two separate, smaller problems. Firstly, we can solve the camera variables with a reduced camera system [25]. The camera variables are updated using the Levenberg-Marquadt update, $\delta\xi = -\left(\mathbf{H}_{\text{rcs}} + \lambda\mathbf{I}\right)^{-1}\mathbf{g}_{\text{rcs}}$. The structure parameters are updated using the standard Gauss-Newton iteration: $\delta\theta = -\mathbf{J}_\theta^+ \boldsymbol{r}$. However, as the Jacobian of the structure parameters is still rather large, we use an approximation $\mathbf{J}_\theta^T\mathbf{J}_\theta \approx diag(\mathbf{J}_\theta^T\mathbf{J}_\theta)$. This speeds up the computation significantly as the diagonal elements can be efficiently computed. In summary, our optimizer splits the problem as in VarPro methods but uses a stochastic subset of terms for the reduced camera system and a diagonal approximation for the volume variables to make the optimization tractable.

4.4. Volume Creation and Refinement

A major advantage of the VDB tree used in our method is that the tree structure is inherently designed to be dynamic. This means, unlike standard voxel grids and SVOs, the tree's structure can be changed online without the need to rebuild the tree itself. This major advantage in our application, as it enables the feature volume to grow and refine itself as the user explores more areas of the scene.

To create the initial tree structure, we use the initial camera poses and depth map to allocate voxel bricks at all the locations where a surface is observed. However, as the camera poses and depth maps are not required to be accurate, this initial allocation is not necessarily representative of where the final occupied voxels will lie. Therefore, we use a splitting rule to dynamically refine the tree during operation, as specified below:

$$\text{refine} \rightarrow \begin{cases} \text{insert block at } \boldsymbol{x}, & \text{if } \sigma(\boldsymbol{x}) > \tau_h, \\ \text{remove block at } \boldsymbol{x}, & \text{if } \sigma(\boldsymbol{x}) < \tau_l \end{cases} \tag{4}$$

For the experiments in this paper we always use $\tau_h = 0.9$ and $\tau_l = 0.4$ unless stated otherwise.

5. Experimental setup

Implementation details: We use a [6,4,3] VDB tree which gives an available resolution of 8192^3 that stores features of dimension $N_f^L = 8$ for all levels $L = 1, 2, 3$. For the network F_θ, we use a shallow, 2 layer MLP with 32 hidden units with an output of dimension of 4. We extract keyframes from the RGB stream when the view overlap with the closest keyframe drops below 40% or the translation from the previous keyframe exceeds $15cm$ for indoor scenes and $40cm$ for outdoor scenes. We apply the volumetric bundle adjustment only to these keyframes. We run all our experiments on a machine with an NVIDIA A6000 GPU and an AMD Ryzen 3990X CPU.

Initialization data: Our method requires as input RGB images along with an estimate of the camera pose and depth of the scene. The estimate of the camera pose can be obtained using a SLAM system such as ORB-SLAM, or by using a tracking system such as the one in ARKit on the iPhone or ARCore on Android devices. For the experiments in this paper we use the pose available in the dataset (eg. the poses from ARKit in our ARKit dataset). For depth input, we require only a rough estimate of the depth of each pixel. For all the experiments in this paper we use an off-the-shelf monocular depth prediction network, MiDAS [16], to get the base depth map corresponding to the RGB images. As this network only outputs relative depth, we use the initial poses to solve for a scale factor for each depth image to convert to consistent metric depths.

Comparisons: We compare our method against 3 types of methods: RGB-D reconstruction systems, monocular reconstruction systems, recent neural volume rendering approaches. Specifically, in terms of RGB-D systems we compare against KinectFusion [13], Colormap Optimization [29] and PolyCamAI [20]. For monocular systems we compare against MVDepthNet [23], ATLAS, [11] and NeuralRecon [18] In terms of neural volume rendering approaches we compare against NeRF [10], NSVF [8], MVS-NeRF [1] and IBRNet [24]. For fairness, we initialize the sparse voxel octree in NSVF using the same monocular depth prediction as in our method as described in the NSVF paper [8].

Datasets: We conduct our experiments on two real-world datasets. *The DTU dataset* [5] consists of 104 scenes of table top objects. There are 64 views per scene captured using a robot arm with associated camera poses. High quality depth images are captured using a structured light sensor. *ARKit scenes dataset*. To test the real-world performance of the proposed method in the target setting, we collect a dataset using an iPhone 12. The dataset contains raw camera poses along with RGB-D images obtained from ARKit on the iPhone.

Evaluation metrics: We evaluate the visual image quality of rendered views using the commonly used PSNR metric. We evaluate the quality of the estimated geometry by reporting the precision, recall and F-score of the global model (see [11]).

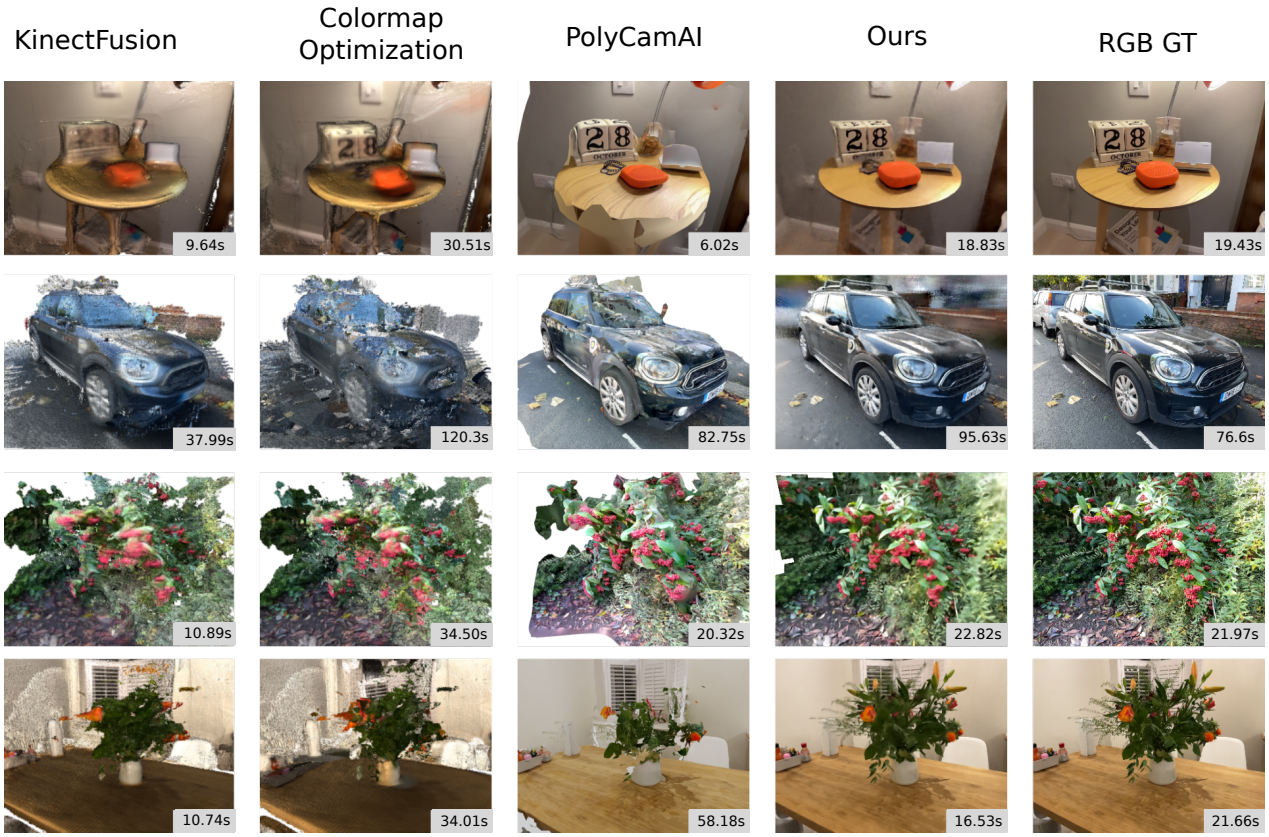

Figure 4. **Comparison to three real-time RGB-D reconstruction systems**. We show three RGB-D systems (a) KinectFusion [13] which fuses depth readings into a dense voxel grid, (b) the colormap optimization approach of [29] which optimizes camera poses to improve texture consistency, (c) PolyCamAI which is a commercial iPhone ARKit-based app and (d) our method. The inset shows the processing time for each method, and the time taken to capture the video. Our method achieves the best quality scene reconstruction.

6. Results and discussion

Comparison to RGB-D systems. We compare the reconstruction quality of our method to three RGB-D reconstruction systems. For these three systems, we use the ground-truth depth images as input. For our system, we take as input the less-accurate monocular depth prediction from MiDAS [16]. See Figure 4 for a qualitative comparison of the reconstruction. Among the four methods, our method achieves the most photorealistic quality of the rendering at the test pose. This is because our method handles view dependent effects and jointly optimizes the color and geometry for photometric consistency. In comparison, KinectFusion [13] (first column) suffers from blurry textures and does not recover from inaccuracies in the depth images used as input. Colormap Optimization [29] (second column), which optimizes the camera poses to ensure photometric consistency,obtains sharper textures than KinectFusion. However, it fails to produce a good reconstruction when the initial geometry is bad such as in the second scene. The PolyCamAI LiDAR app [20] (third column) achieves better texture quality than KinectFusion and Colormap Optimization. However, this method tends to give overly smoothed geometry and it does not account for view dependent effects, causing artifacts on the reflective and specular surfaces, such as the table in the first scene and the car in the second scene.

Table 1. **Novel view rendering quality.** Comparison of the rendering quality of our method to neural volume rendering and view synthesis approaches on the DTU dataset.

Scan	#1	#8	#21	#103
	PSNR↑			
PixelNeRF	21.64	23.70	16.04	16.76
IBRNet	25.97	27.45	20.94	27.91
NeRF$_{10.2h}$	26.62	28.33	23.24	30.40
NSVF$_{\approx 10.2h}$	28.77	**29.21**	**27.79**	32.79
MVSNeRF$_{ft-15min}$	28.05	28.88	24.87	32.23
Ours$_{15min}$	**29.72**	28.90	25.43	**33.3**

Comparison to neural volume rendering methods. In this comparison, we focus on the convergence time needed

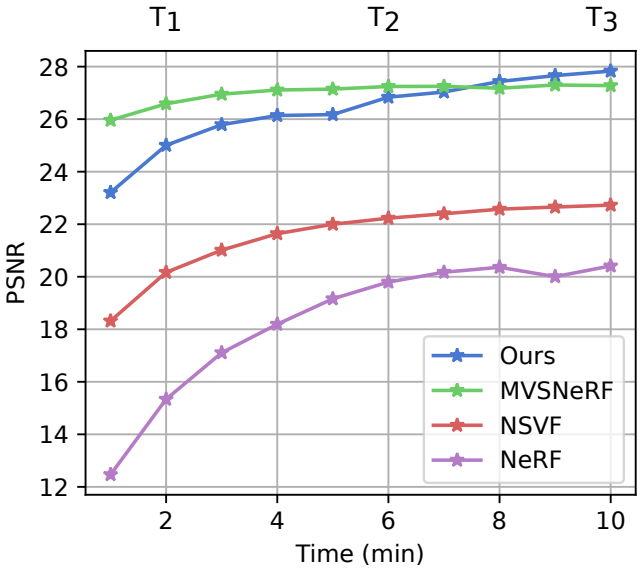

Figure 5. **Comparison to neural volumetric rendering approaches**. MVSNeRF [1] achieves good performance but uses only local information (3 neighboring views) for view synthesis. In contrast, our method constructs a global scene representation thus outperforming MVSNeRF after a few minutes of training. NSVF and NeRF need hours of training to reach the same quality of reconstruction.

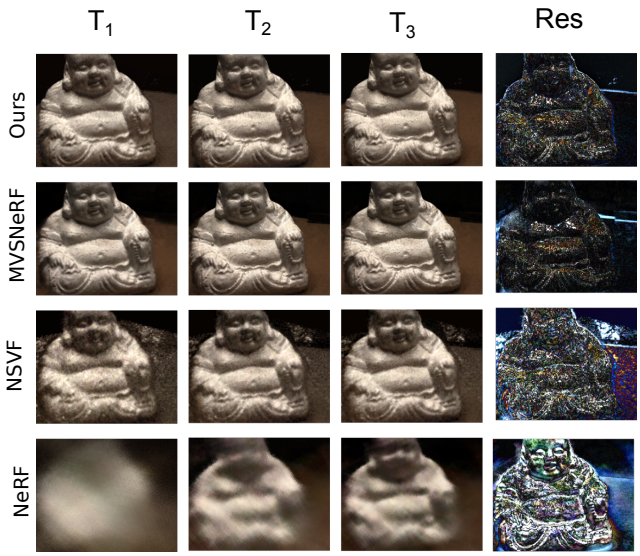

Figure 6. **Qualitative comparison to neural volumetric rendering approaches.** We show the results of our method, NeRF [10], MVSNeRF [1] and NSVF [8] at three different times during optimization corresponding to the times in Figure 5. The last column shows the error image. Our method and MVSNeRF perform the best in terms of visual quality, however, MVSNeRF only constructs a local (view-based) representation of the scene wheras our method constructs a global representation. This allows our method to achieve the best performance at T_3.

for a method to render good quality images. Figure 5 shows the PSNR of validation views from a scene of the DTU dataset over training time and Figure 6 shows a rendered test view at 3 timestamps: T_1=2min, T_2=6min and T_3=10min. NeRF [10] exhibits the slowest convergence of all four methods. By the end of 10min, the rendered image from NeRF still remains quite blurry. NSVF [8] uses an explicit feature volume in the form of a sparse voxel octree and thus achieves faster convergence than NeRF. However, our method outperforms both NeRF and NSVF by a large margin in terms of PSNR metric across all time scales, indicating our method can converge much faster and is able to generate photorealistic views in a very short time frames. The closest contender to our method is MVSNeRF [1], which learns a "generalizable" network and is able to achieve a very high PSNR with little training. However, unlike ours and the other methods, MVSNeRF does not construct a global representation of the scene. Instead, it forms a view-centric volume which it uses to synthesize novel views near the target-view. This behavior of only adopting local information limits MVSNeRF's ability to fuse information from further views. Therefore, our method is able to converge to a better PSNR as training goes on. In Table 1, we show the quantitative view quality results on the DTU dataset. Even when only optimizing for a short period of time, our method outperforms both NeRF and NSVF that have been trained for $\approx 10h$. Our method also outperforms MVSNeRF which has been fine-tuned for the same amount of time.

VBA convergence: In Figure 8 we visualize the difference in the convergence rate between our novel optimizer used in the volumetric bundle adjustment compared to ADAM. By utilizing the Hessian as in LM-based optimization, our method is able to converge much faster.

Evaluation of geometry and depth accuracy. In this experiment, we investigate whether our method is able to improve on the base depth estimate and how the depth estimated by our method compares to the LiDAR-based depth from iPhone ARKit. Figure 7 shows the results of this comparison. Our method takes as input the base depth obtained using MiDAS [16] (left image) and estimates the final depth prediction (middle image). The inpainted LiDAR depth produced by ARKit on an iPhone 12 Pro is shown in the right. We can see that the depth estimation produced by our method improves considerably on the initial depth map that is taken as an input by our approach to guide the volumetric optimization. This indicates that our method is able to correct the imperfections in the inaccurate depth readings. Secondly, our method achieves more detailed reconstruction of delicate structures than the LiDAR data from ARKit, as shown in the highlight boxes.

Ablation study. In this ablation study, we first examine the impact of different scene representations and different optimizers on the quality of the scene reconstruction. We

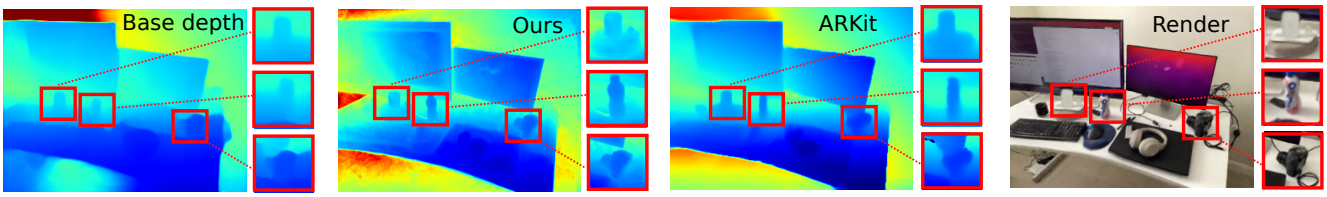

Figure 7. **Depth reconstruction quality**. We show the depth image estimated using (a) an off-the-shelf monocular depth prediction method [16], (b) the final depth estimated by our method and (c) the depth obtained from the iPhone 12 Pro LiDAR for comparison.

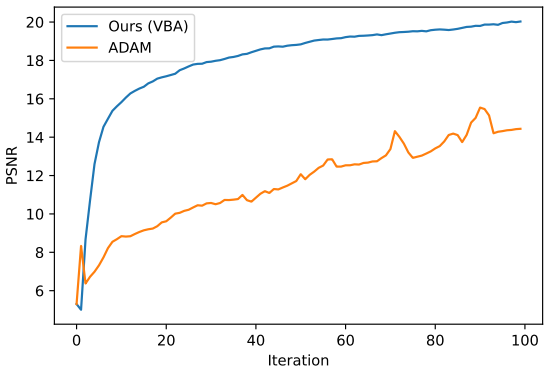

Figure 8. **Optimizer efficiency**. Visualization of the convergence of the novel second-order optimizer used in our volumetric bundle adjustment (VBA) compared to ADAM.

show the results in the top half of Table 2. We see that both our bundle adjustment method, VBA, and the nVDB representation contribute significantly to the output quality for the view synthesis task. Therefore, both components are essential to our goal of fast photorealistic scene capture. Next, we evaluate how sensitive our method is to the quality of the base depth image. We show the quality of the rendered image given the base depth produced from three different methods: 1) MiDAS_s (the small version of MiDAS), 2) MiDAS [16] and 3) MVSNet [26]. We see that our method is able to produce good quality reconstructions with the depth from all three approaches.

7. Limitations and social impact

There are some limitations to our approach. Firstly, our method is reliant on a guiding depth image to initialize the volume efficiently. As we have shown, this is not a major issue as our method is not very sensitive to the initial depth input and there are many depth estimation methods available to provide this depth input. Secondly, our method does not handle infinitely far background elements as the nVDB needs to cover the whole scene. This could be overcome by adding an extra background appearance feature that is parameterized like an environment map. As our method involves generating synthetic images, there is the possibility for its use in creating fake media and spreading disinformation. However, this cannot be done using our system alone and therefore the direct impact our method can have on society is mainly positive.

Table 2. **Impact of the optimizer, representation and depth guidance.** We show the performance using our nVDB representation compared to the popular ADAM [6] optimizer used for training NeRF and MVSNeRF. In terms of the representation we show our nVDB and NSVF [8] which is the closest related approach. We also show the impact of different depth guidance methods.

Base Depth	Rep.	Opt.	PSNR
MiDAS_s	nVDB	ADAM	26.70
MiDAS_s	NSVF	ADAM	22.94
MiDAS_s	nVDB	VBA	28.54
MiDAS	nVDB	VBA	28.906
MiDAS_s	nVDB	VBA	28.54
MVSNet	nVDB	VBA	29.45

8. Conclusion

In this paper, we have proposed a novel system for online photorealistic scene capture. This is achieved through two contributions: The first is a neural tree-based representation of the scene, called nVDB. The nVDB efficiently stores scene features and can dynamically adjust its topology for online reconstruction. The second is an optimization strategy that improves the efficiency of the bundle adjustment allowing us to jointly refine the volume parameters and camera poses to ensure photometric consistency. Together these two contributions allow our method to capture photorealistic models of scenes very efficiently. The results show that our method outperforms current reconstruction systems in terms of view quality and neural volume rendering methods in terms of efficiency. We believe our method can bring new levels of realism and immersion to AR applications.

Acknowledgements This research was supported by an Imperial College Research Fellowship. The authors would like to thank Shuyu Lin and Coconut for continuous support and discussions.

References

[1] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Int. Conf. Comput. Vis.*, 2021. 3, 5, 7

[2] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Trans. Graph.*, 2017. 2

[3] Je Hyeong Hong. *Widening the basin of convergence for the bundle adjustment type of problems in computer vision*. PhD thesis, University of Cambridge, 2018. 5

[4] Yuxin Hou, Juho Kannala, and Arno Solin. Multi-view stereo by temporal nonparametric fusion. In *Int. Conf. Comput. Vis.*, 2019. 2

[5] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014. 5

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2, 8

[7] Joo Ho Lee, Hyunho Ha, Yue Dong, Xin Tong, and Min H Kim. Texturefusion: High-quality texture acquisition for real-time rgb-d scanning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[8] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Adv. Neural Inform. Process. Syst.*, 2020. 2, 3, 5, 7, 8

[9] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 2019. 2

[10] Ben Mildenhall, Pratul P Srinivasan, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis.*, 2020. 2, 5, 7

[11] 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 *Eur. Conf. Comput. Vis.*, 2020. 1, 2, 5

[12] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.*, 2013. 3

[13] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE Int. Symp. on Mixed Aug. Reality*, 2011. 1, 2, 5, 6

[14] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Int. Conf. Comput. Vis.*, 2011. 1, 2

[15] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 2013. 1

[16] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. 5, 6, 7, 8

[17] Thomas Schöps, Torsten Sattler, Christian Häne, and Marc Pollefeys. 3d modeling on the go: Interactive 3d reconstruction of large-scale scenes on mobile devices. In *Int. Conf. 3D. Vis.*, 2015. 1

[18] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 2, 5

[19] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. 2021. 3

[20] Polycam Team. Polycamai, 2014. https://poly.cam/. 2, 5, 6

[21] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, 1999. 4

[22] Emanuele Vespa, Nikolay Nikolov, Marius Grimm, Luigi Nardi, Paul HJ Kelly, and Stefan Leutenegger. Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 2018. 3

[23] Kaixuan Wang and Shaojie Shen. Mvdepthnet: Real-time multiview depth estimation neural network. In *Int. Conf. 3D. Vis.*, 2018. 2, 5

[24] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3, 5

[25] Oliver J Woodford and Edward Rosten. Large scale photometric bundle adjustment. 2020. 4, 5

[26] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Eur. Conf. Comput. Vis.*, 2018. 8

[27] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. *Int. Conf. Comput. Vis.*, 2021. 3

[28] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3

[29] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Trans. Graph.*, 2014. 2, 5, 6