VisFusion: Visibility-aware Online 3D Scene Reconstruction from Videos

Huiyu Gao, Wei Mao, Miaomiao Liu

Australian National University

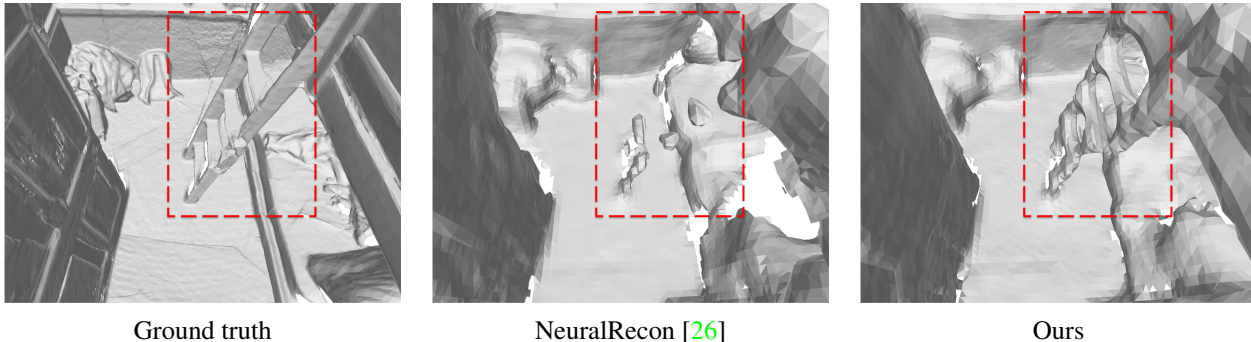{huiyu.gao, wei.mao, miaomiao.liu}@anu.edu.au

Figure 1. The 3D reconstruction by NeuralRecon [26] and the proposed method. Our method reconstructs the scene with more fine details.

Abstract

We propose VisFusion, a visibility-aware online 3D scene reconstruction approach from posed monocular videos. In particular, we aim to reconstruct the scene from volumetric features. Unlike previous reconstruction methods which aggregate features for each voxel from input views without considering its visibility, we aim to improve the feature fusion by explicitly inferring its visibility from a similarity matrix, computed from its projected features in each image pair. Following previous works, our model is a coarse-to-fine pipeline including a volume sparsification process. Different from their works which sparsify voxels globally with a fixed occupancy threshold, we perform the sparsification on a local feature volume along each visual ray to preserve at least one voxel per ray for more fine details. The sparse local volume is then fused with a global one for online reconstruction. We further propose to predict TSDF in a coarse-to-fine manner by learning its residuals across scales leading to better TSDF predictions. Experimental results on benchmarks show that our method can achieve superior performance with more scene details. Code is available at: https://github.com/huiyu-gao/VisFusion.

1. Introduction

3D scene reconstruction from RGB videos is a critical task in 3D computer vision, which finds its broad appli-

cations in augmented reality (AR), robot navigation and human-robot interaction. These applications require accurate, complete and real-time 3D reconstruction of scenes. While state-of-the-art SLAM systems [3, 31] can track the camera motion accurately by leveraging both visual and inertial measurements in an unknown environment, the reconstructed map from a SLAM system only contains sparse point clouds such that dense reconstruction from monocular videos remains as a challenging problem.

Many previous methods [1, 18] assume the observation of the whole video sequence for the reconstruction, which is not practical for online applications like VR games. In this paper, we follow [26] to propose an online 3D reconstruction method. Given input images, most earlier 3D reconstruction methods [23, 35] adopt a two-stage pipeline, which first estimates the depth map for each keyframe based on multi-view stereo (MVS) algorithms [11, 14, 29, 32] and then fuses the estimated depth maps into a Truncated Signed Distance Function (TSDF) volume [19]. The Marching Cubes algorithm [16] is then used to extract the 3D mesh. However, those two-stage pipelines struggle to produce globally coherent reconstruction since each depth map is estimated separately [26], especially for low texture regions like walls whose depth values are extremely hard to estimate with only several local views. To address this, more recent works [2, 26, 33] propose to fuse image features into a global 3D volume and directly regress TSDF [26, 33] or occupancy [2] given the feature volume. Such strategy allows for an end-to-end global surface reconstruction.

The problem of occlusion naturally arises for global feature fusion. Previous methods [2, 26] either completely ignore it by simply averaging the multi-view features [26] for each voxel or implicitly model the visibility via the attention mechanism [2]. However, without explicit supervision, such attention cannot guarantee to encode the correct visibility. In this paper, we thus propose to explicitly predict the visibility weights of all views for each voxel with ground truth supervision. In addition, voxels will be considered *visible* in at least one view in [2] due to the normalization of the attention mechanism, while in our method, empty voxels and fully occluded voxels are *invisible* in any view to avoid introducing noises. Specifically, given a fragment of a video sequence observing the same 3D region, we first project each 3D voxel onto different view images to obtain 2D features. We then compute the pair-wise similarities of these features. Since features of the same occupied voxel are often similar across views, such similarity map naturally encodes the information of whether a 3D voxel is visible at a particular camera view or not (see Fig. 4). We thus use this similarity map to predict visibility weights.

For volumetric-based methods, it is common practice to adopt a coarse-to-fine pipeline [2, 18, 25, 26]. One of its key steps is voxel sparsification which eliminates empty voxels at coarse level for better performance and smaller memory consumption. To the best of our knowledge, previous methods [2, 18, 25, 26] propose to globally sparsify the volume by removing voxels whose occupancy probabilities are lower than a predefined threshold. However, such fixed threshold tends to sparsify more voxels than necessary, especially to remove voxels covering thin structures such as chair legs. At coarse level where the thin structure only occupies a small portion of the voxel, the features of such thin structure are likely ignored leading to low occupancy probability prediction and resulting in the removal of such voxel. However, such voxel should rank highly, based on the occupancy probability, among voxels along the visual ray defined by the pixel observing this thin structure. Inspired by this, we introduce a novel ray-based sparsification process. In particular, for any image, we first cast a ray from every pixel to get the voxels this ray passes. For each ray, we then keep voxels with top occupancy scores to next level. Unlike previous works [2, 18, 25, 26] that sparsify the global volume, our ray-based sparsification is performed on local 3D volume. Our ray-based sparsifying strategy allows us to retain more surface voxels to the next level leading to a more complete reconstruction.

Furthermore, previous coarse-to-fine methods [2, 18, 25, 26] directly regress the TSDF at each level discarding the relationships between the TSDF predicted at coarse and that at fine level. In our method, at each fine level, we aim to predict a residual between the TSDF volume upsampled from the coarser level and that of the fine level, which is shown to be more accurate in TSDF estimation.

In summary, our contributions are (i) a visibility-aware feature fusion module which explicitly predicts visibility weights used for feature aggregation for voxels; (ii) a ray-based voxel sparsifying algorithm which leads to the reconstruction of more scene structure details. (iii) an easier way of TSDF regression by learning the residual to the upsampled coarse TSDF volume for improved TSDF estimation. Our model outperforms the existing online feature fusion based methods.

2. Related Work

Multi-view depth estimation. COLMAP [22] is one of the most popular methods for Multi-View Stereo which jointly estimate the depth and normal with photometric and geometric consistency. Although it is robust and accurate, COLMAP has difficulty in densely reconstructing areas without distinctive features. Deep learning-based methods [11, 12, 32] try to relax the dependency of the photo-consistency assumption with data-driven priors. Although some recent studies [9, 11, 12, 14, 36] show that constructing 3D cost volumes followed by 3D CNNs for the cost volume processing could achieve promising results, they have large gpu memory cost and struggle to achieve real-time inference speed. Recently, SimpleRecon [21] utilizes image priors and integrates the geometric metadata into the cost volume without using 3D convolutions. The scene geometry is then obtained by a depth fusion approach, such as volumetric TSDF fusion [7], which fuses the estimated depth maps into geometric 3D representation. While the incremental volumetric TSDF fusion method proposed by KinectFusion [19] is widely applied due to its simplicity and effectiveness, such two-stage approach suffers from computational redundancy and causes artifacts in the 3D reconstruction as the depth maps are estimated separately.

End-to-end 3D scene reconstruction. To achieve coherent reconstruction, several volumetric approaches are proposed to directly infer the 3D geometry from volumetric features in an end-to-end manner. SurfaceNet [13] is the first volumetric method to predict volumetric surface occupancy from two input images with a 3D convolutional network. Atlas [18] extends this work to multi-view setup by averaging the features back projected from all input images in the sequence and directly regressing the global TSDF volume of the scene. VolumeFusion [5] and 3DVNet [20] further improve Atlas by using the estimated depth maps of input views as intermediate representations whose training is supervised by ground truth depth supervision. VoRTX [25] adopts the transformer architecture [30] for multi-view feature fusion. While improving the reconstruction quality, these methods [5, 18, 20, 25] can only work in an offline fashion, requiring entire image sequences as input, thus cannot achieve online reconstruction.
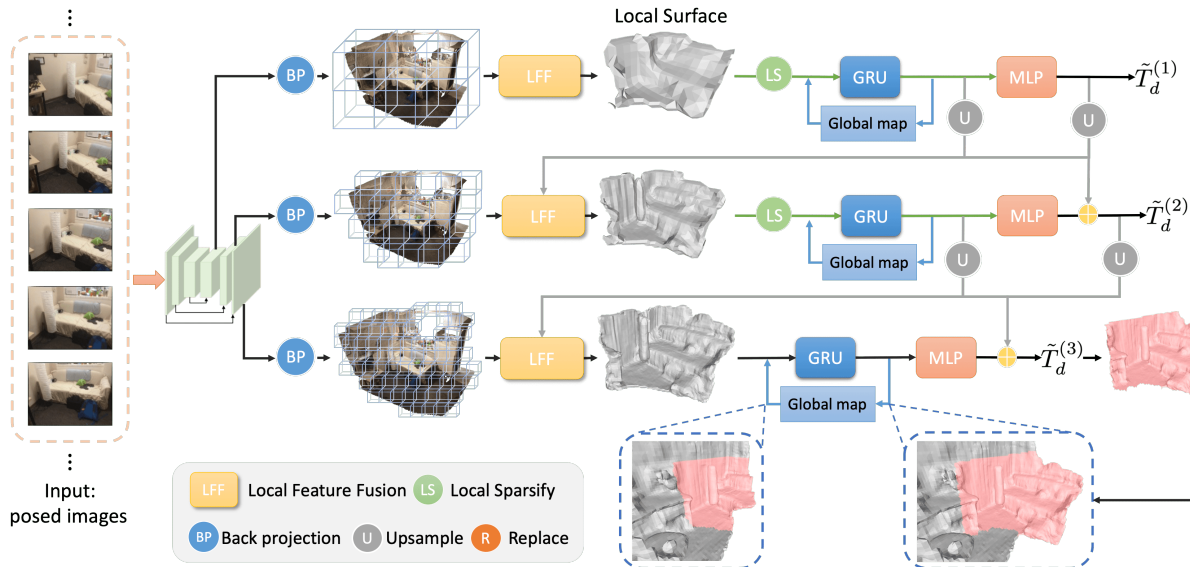
Figure 2. **Overview of our coarse-to-fine pipeline.** Given a fragment of a video, we first construct 3D feature volumes of different resolutions. The feature of each voxel is obtained by projecting it back to every camera view. The features from different camera views are then fused via the predicted visibility (local feature fusion). We then extract the local occupancy and TSDF from the fused feature followed by a ray-based sparsification process to remove empty voxels. The sparse local feature volume is finally fused to global via GRU and used to produce the final TSDF. The global feature and final TSDF of coarse level are further upsampled and fed to the next level for refinement.

For online end-to-end 3D reconstruction methods, NeuralRecon [26] proposes the first real-time framework for dense reconstruction from posed monocular videos. It converts Atlas [18] to an incremental reconstruction system by averaging features only within local video fragments and fusing across fragments using a 3D convolutional variant of Gated Recurrent Unit (GRU) [6] module. However, simply fusing image features by averaging in local fragments does not effectively model the visibility and solve occlusion issues. In addition, although NeuralRecon employs a coarse-to-fine approach to reduce memory consumption, it uses a global threshold to sparsify voxels between layers, resulting in losing fine structures. Similar to [25], Transformer-Fusion [2] also leverages the transformer architecture [30] to fuse features in the global voxel space. It performs in an online manner by saving features from multiple views and dropping saved features with the lowest attention weight. However, this design makes TransformerFusion suffer from high computing costs and storage consumption. Recently, PlanarRecon [33] proposes an incremental framework for 3D plane detection and fusion, which mainly focuses on planar region reconstruction. In contrast, we aim for the reconstruction of the scene structure in general.

Residual learning and refinement. In MVS task, many depth estimation works [4, 10, 34] predict a residual to an initial depth prediction and refine it gradually across multi-scales. Inspired by such strategy, we propose a cascaded TSDF learning framework that gradually learns the TSDF residual to the one learned from a coarser level and recon-

structed from previous fragments. By doing so, our model is able to achieve better reconstruction performance.

3. Method

Let us now introduce our approach to online 3D scene reconstruction from posed monocular videos. Given a sequence of monocular images $\{\mathbf{I}_i\}$ of a scene, where $\mathbf{I}_i \in \mathbb{R}^{H \times W \times 3}$ represents the ith RGB image in the video sequence, with their corresponding camera intrinsics, rotation matrices, and translation vectors $\{\mathbf{K}_i \in \mathbb{R}^{3 \times 3}, \mathbf{R}_i \in \mathbb{R}^{3 \times 3}, \mathbf{t}_i \in \mathbb{R}^3\}$ estimated by a SLAM system, our method incrementally reconstructs the dense 3D geometry of the scene. To achieve online reconstruction, following [11, 26], we sequentially select suitable keyframes. A new incoming frame is selected as a keyframe if the camera motion is greater than a predefined threshold [26]. We then split the keyframe stream into M non-overlapping fragments $\mathcal{F}_t = \{\mathbf{I}_i\}_{i=N(t-1)+1}^{Nt}, t = \{1, 2, \ldots, M\}$, each of which consists of N consecutive keyframes. The view-frustum of each keyframe image is computed with a fixed max depth range $d_{\max}$. Same as [26], we only reconstruct the cubic-shaped region that encloses the view-frustums of all images within this fragment. Such region is called fragment bounding volume (FBV) [26].

An overview of our framework is shown in Fig. 2. Our method predicts the TSDF representation of the scene, in a coarse-to-fine manner. It consists of three modules, namely, the local feature fusion module, the local sparsification
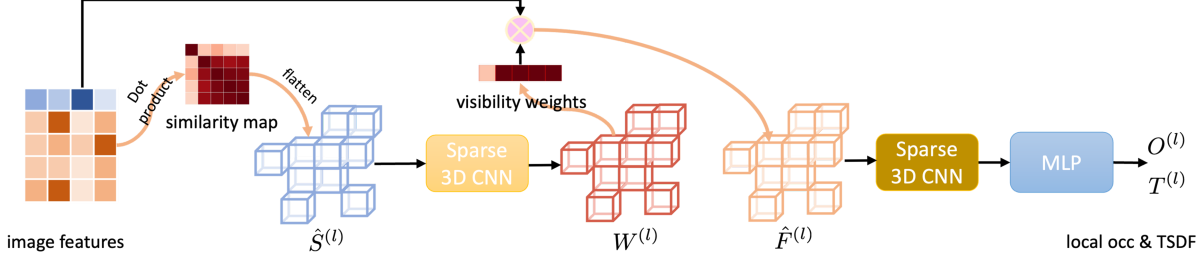
Figure 3. **Local feature fusion.** For each voxel, given the N features extracted from different camera views, we first compute their pairwise cosine similarities. Such similarity map is then flattened as a vector. The 3D volume of such vectors is processed by a 3D CNN to produce visibility weights $\mathbf{W}$, which is used to aggregate the back projected features. With another 3D CNN followed by one MLP layer, we obtain the local occupancy and TSDF for our ray-based sparsification.

module, and the global feature fusion module. Below, we introduce these modules in detail.

3.1. Local Feature Fusion

Here we discuss our proposed local feature fusion module at level l of our coarse-to-fine framework. Given N images in current fragment $\mathcal{F}_t$, our local feature fusion module aims to aggregate image features according to the visibility of voxels in the corresponding FBV. In particular, the local FBV is represented as a 3D voxel grid of $D^{(l)}$ voxels $\mathbf{V}^{(l)} \in \mathbb{R}^{D^{(l)} \times 3}$. Note that, the FBVs at different levels have different resolutions thus different numbers of voxels [1]. We follow [26] to use a variant of MnasNet [27] to extract 2D features from input images. Each volume is then projected to different camera views to obtain an initial local feature volume $\mathbf{F}^{(l)} \in \mathbb{R}^{D^{(l)} \times N \times C^{(l)}}$ as,

$$\mathbf{F}_{dn}^{(l)} = \mathbf{H}_n^{(l)}(\Pi(\mathbf{K}_n^{(l)}, \mathbf{R}_n \mathbf{V}_d^{(l)} + \mathbf{t}_n)) , \quad (1)$$

where $\mathbf{F}_{dn}^{(l)} \in \mathbb{R}^{C^{(l)}}$ is the feature vector of voxel $\mathbf{V}_d^{(l)}$ obtained from n^{th} image feature map $\mathbf{H}_n^{(l)} \in \mathbb{R}^{H^{(l)} \times W^{(l)} \times C^{(l)}}$. With a little abuse of notation, we also use $\mathbf{H}_n^{(l)}(\cdot)$ to represent 2D feature interpolation. $\Pi(\cdot, \cdot)$ is the camera perspective projection. Note that, we use different resolutions of 2D image features for different levels and the camera intrinsics are also changed accordingly. For each voxel, given the features from N different views $\{\mathbf{F}_{dn}^{(l)}\}_{n=1}^N$, our goal now is to fuse these local features into one. Unlike previous scene reconstruction methods [18, 26] that fuse those features by simply averaging them, we propose to directly predict the visibility weights for feature fusion.

Let us first define the visibility of a voxel. A voxel is considered to be occupied when its distance to the nearest surface is less than the truncation distance λ of the TSDF. A voxel is visible to a view if it is occupied and not occluded by other occupied voxels. This differs from the visibility

defined in [15, 25] where empty voxels are also treated as visible as long as they are not occluded [2]. As illustrated in Fig. 3, we proposed to predict the visibility from the pairwise feature similarities. In particular, given the features of a voxel $\mathbf{F}_{dn}^{(l)}$, we compute the cosine similarity between features from every two views as,

$$\mathbf{S}_{dmn}^{(l)} = \frac{\mathbf{F}_{dm}^{(l)T} \mathbf{F}_{dn}^{(l)}}{\|\mathbf{F}_{dm}^{(l)}\|_2 \|\mathbf{F}_{dn}^{(l)}\|_2} , \quad (2)$$

where $\mathbf{S}_d^{(l)} \in \mathbb{R}^{N \times N}$ is the similarity map of this voxel and $m, n \in \{1, 2, \cdots, N\}$. As illustrated in Fig. 4, such similarity maps provide important geometry heuristics for predicting the visibility of each voxel in the local fragment. We then eliminate the diagonal entries which are always ones and flatten the map to obtain a long similarity feature denoted as $\hat{\mathbf{S}}_d^{(l)} \in \mathbb{R}^{N(N-1)}$. For the whole FBV, we then have a similarity volume $\hat{\mathbf{S}}^{(l)} \in \mathbb{R}^{D^{(l)} \times N(N-1)}$. It is fed into a 3D CNN to obtain visibility weights $\hat{\mathbf{W}}^{(l)} \in \mathbb{R}^{D^{(l)} \times N}$. For each voxel, there are N weights that will be used to fuse the local feature volume $\mathbf{F}^{(l)}$ as,

$$\hat{\mathbf{F}}_d^{(l)} = \sum_{n=1}^N \hat{\mathbf{W}}_{dn}^{(l)} \mathbf{F}_{dn}^{(l)} , \quad (3)$$

where $\hat{\mathbf{F}}^{(l)} \in \mathbb{R}^{D^{(l)} \times C^{(l)}}$ is the fused local feature volume. Another 3D CNN followed by a linear layer is then used to generate local occupancy $\hat{\mathbf{O}}^{(l)} \in \mathbb{R}^{D^{(l)}}$ and local TSDF $\hat{\mathbf{T}}^{(l)} \in \mathbb{R}^{D^{(l)}}$.

Training loss. The training losses for the visibility, local occupancy and local TSDF are defined as follows.

$$\mathcal{L}_w^{(l)} = \frac{1}{N D^{(l)}} \sum_{d=1}^{D^{(l)}} \sum_{n=1}^N (\hat{\mathbf{W}}_{dn}^{(l)} - \frac{\mathbf{W}_{dn}^{(l)}}{\sum_{m=1}^N \mathbf{W}_{dm}^{(l)}})^2 , \quad (4)$$

---

[1]At the first (coarsest) level, the local volume will be a full grid while for the rest, the volumes will be a small subset of the full grid upsampled from the sparsified ones of the previous level.

[2]Based on the definition of visibility in VoRTX [25] and NeuRay [15], besides surface voxels (points), all empty voxels (points) along the ray before it hits the surface are also defined as *visible*. By contrast, we define those empty and occluded voxels as *invisible* to avoid introducing noises to our feature volume.
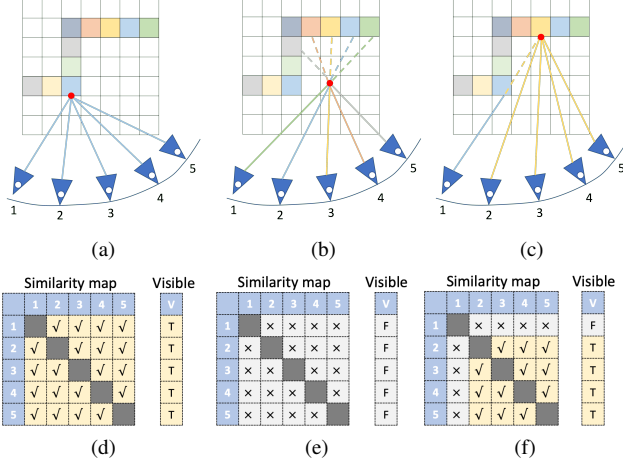
Figure 4. **Different kinds of voxels and their similarity maps.** $\sqrt{}$ denotes relatively similar and $\times$ denotes relatively dissimilar. T represents True and F represents False. (a) Surface voxel (b) Empty voxel (c) Surface voxel with occlusion (d), (e), (f) Similarity map and visibility definition corresponding to (a), (b), (c).

where $\mathbf{W}_{dn}^{(l)} \in \{0, 1\}$ is the ground truth visibility which indicates whether the dth voxel is visible in n^{th} view and $\hat{\mathbf{W}}_{dn}^{(l)} \in [0, 1]$. Note that, here we directly train our model to output the normalized visibility.

$$\mathcal{L}_o^{(l)} = \frac{1}{D^{(l)}} \sum_{d=1}^{D^{(l)}} \text{BCE}(\hat{\mathbf{O}}_d^{(l)}, \mathbf{O}_d^{(l)}) \,, \tag{5}$$

where $\text{BCE}(\cdot, \cdot)$ is the binary cross entropy; $\mathbf{O}_d^{(l)} \in \{0, 1\}$ is the ground truth occupancy and $\hat{\mathbf{O}}_d^{(l)} \in [0, 1]$.

$$\mathcal{L}_t^{(l)} = \frac{1}{D^{(l)}} \sum_{d=1}^{D^{(l)}} |\ell(\hat{\mathbf{T}}_d^{(l)}) - \ell(\mathbf{T}_d^{(l)})| \,, \tag{6}$$

where $\ell(x) = \text{sgn}(x) \log(|x|+1)$ is the log scale function as used in [26] and $\text{sgn}(\cdot)$ is the sign function; $\mathbf{T}_d^{(l)} \in [-1, 1]$ is the ground truth TSDF and $\hat{\mathbf{T}}_d^{(l)} \in [-1, 1]$.

With the local occupancy at level l, we would like to discard the empty voxels and only upsample the remaining ones to the next level $l + 1$. Existing methods [18, 26] that rely on a fixed threshold tend to sparsify more voxels than necessary leading to incomplete reconstruction especially for thin structures. In the next section, we introduce our ray-based local sparsification approach to address this.

3.2. Ray-based Local Sparsification

Given the camera parameters $\{\mathbf{K}_i^{(l)}, \mathbf{R}_i, \mathbf{t}_i\}_{i=N(t-1)+1}^{Nt}$ of images in current fragment $\mathcal{F}_t$, we first cast a ray from each pixel in these images. Given a ray, let us denote the voxels that are passed by this ray as $\{\mathbf{v}_i\}_{i=1}^R$ where $\mathbf{v}_i \in \mathbb{R}^3$ is the 3D coordinates of the voxel centre. These voxels are
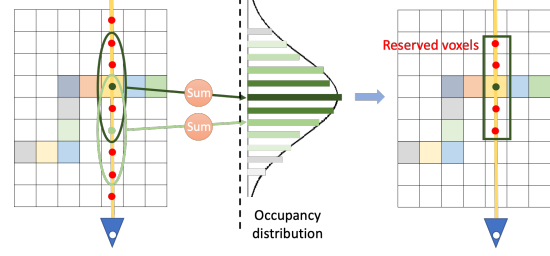

Figure 5. **Ray-based local sparsification.** Given the voxels passed by a visual ray and their local occupancies, we obtain a set of sliding windows, each of which covers several consecutive voxels. The sliding window with the highest sum of occupancies computed from covered voxels is reserved for the next level.

organized in ascending order by their depth to the corresponding camera view as shown in Fig. 5. Their occupancies are represented as $\{\hat{o}_i\}_{i=1}^R$ where $\hat{o}_i \in [0, 1]$ is obtained from the predicted local occupancy volume using the voxel coordinate $\mathbf{v}_i$.

For each ray, one can then simply choose voxels with top-k occupancies. However, such strategy is not robust enough and as will be shown in our experiments, is not optimal. Alternatively, we propose to use a sliding window along the ray and rank each sliding window according to the sum of all the voxel occupancies in it. Specifically, let us assume the size of such sliding window are K and $K < R$, the set of sliding windows can then be represented as $\{\mathcal{W}_i\}_{i=1}^{R-K}$ where $\mathcal{W}_i = \{\mathbf{v}_j\}_{j=i}^{i+K-1}$. The sum of occupancies for i^{th} sliding window is, $\hat{u}_i = \sum_{j=i}^{i+K-1} \hat{o}_i$. The sliding window with the highest sum of occupancy scores is then selected as $i^* = \text{argmax}_i \hat{u}_i$. By doing so, our sparsification strategy adaptively selects the most likely occupied voxels for every ray which is better than only relying on a fixed threshold. We use the same criteria to select sliding windows for each ray in every input image. A voxel is sparsified only if it is not in any of these selected sliding windows. Note that, our ray-based sparsification is performed at the local volume instead of the global one. This is to avoid the situation when a ray not only passes the current surface but also existing surfaces in the global volume. In this case, there are potentially multiple sliding windows with high sum of occupancies.

3.3. Global Feature Fusion

Our model maintains a global feature volume which stores the information from all historically reconstructed fragments [26]. The stored global feature volume may overlap with the FBV of the current fragment. We follow [26] to use a 3D convolutional variant of Gated Recurrent Unit (GRU) [6] to fuse the local feature volume to global. Specifically, after local sparsification, we have a set of local features denoted as $\mathbf{L}^{(l)}$ and the corresponding global features as $\mathbf{G}^{(l)}$. Note that, for the overlapping local volume, its

corresponding global feature is directly obtained from the stored volume. For the non-overlapping ones, the global feature will be initialized as zeros. The fused global feature is then updated as

$$\tilde{\mathbf{G}}^{(l)} = \text{GRU}(\mathbf{L}^{(l)}, \mathbf{G}^{(l)}) , \quad (7)$$

where $\mathbf{G}^{(l)}$ and $\mathbf{L}^{(l)}$ are used as the hidden state and input to the GRU, respectively. The fused $\tilde{\mathbf{G}}^{(l)}$ are then used for global TSDF prediction via a single layer MLP.

Unlike existing coarse-to-fine works [25,26] that directly regress the TSDF, we propose an easier way by taking advantage of the predicted TSDF at previous coarse level. In particular, given the global TSDF volume estimated from the above fused global feature $\tilde{\mathbf{T}}^{(l-1)}$ at level $l-1$, we first upsample it to match the resolution of current level as $\tilde{\mathbf{T}}^{(l-1)\uparrow}$. The output of current level is treated as a residual $\Delta\tilde{\mathbf{T}}^{(l)}$ and the final global TSDF at level l is,

$$\tilde{\mathbf{T}}^{(l)} = \tilde{\mathbf{T}}^{(l-1)\uparrow} + \Delta\tilde{\mathbf{T}}^{(l)} . \quad (8)$$

As shown in our ablation study in next section, such strategy leads to better performance. Similar to Eq. (5) and Eq. (6), we can then compute the global occupancy and TSDF loss as $\mathcal{L}_O^{(l)}$ and $\mathcal{L}_T^{(l)}$.

Overall training loss. The overall training loss for the whole model is then,

$$\mathcal{L} = \sum_{l=1}^{3} \omega^{(l)}(\mathcal{L}_w^{(l)} + \mathcal{L}_o^{(l)} + \mathcal{L}_t^{(l)} + \mathcal{L}_O^{(l)} + \mathcal{L}_T^{(l)}) , \quad (9)$$

where $\omega^{(l)}$ is the loss weight at level l.

3.4. Implementation Details

Our network is trained using Adam optimizer with batch size of 2 on an Nvidia RTX 3090 GPU for 50 epochs. The loss weights $\omega^{(1)}$, $\omega^{(2)}$, $\omega^{(3)}$ are set as 1, 0.8, 0.64. We choose the checkpoint of the last training epoch for evaluation. Following [26], we use torchsparse [28] for 3D sparse convolution and initialize the feature extraction backbone, a variant of MnasNet [27], with the pretrained weights from ImageNet. In each local fragment, the number of key frame images N is set to 9. We use 3 coarse-to-fine layers and set the voxel size for each layer as 16 cm, 8 cm, and 4 cm respectively. The TSDF truncation distance λ is set as three times the voxel size for each layer. The size of the sliding window K used in our local sparsification module is 9 across all layers with different resolutions.

4. Experiments

4.1. Datasets, Metrics & Baselines

Datasets. We evaluate our method on the ScanNet (V2) [8] and 7-Scenes dataset [24]. ScanNet consists of 2.5M images in 1613 scans across 807 distinct scenes with ground-truth depths, camera poses, surface reconstructions, and

	Method	Acc↓	Comp↓	Chamfer↓	Prec↑	Recall↑	F-score↑
Depth Fusion	COLMAP [22]	13.5	**6.9**	10.2	0.505	**0.634**	0.558
	MVDNet [32]	20.5	8.4	14.5	0.231	0.473	0.307
	GPMVS [11]	16.2	7.9	12.1	0.335	0.533	0.408
	DPSNet [12]	17.7	8.1	12.9	0.272	0.497	0.349
	DeepVMVS [9]	11.7	7.6	9.7	0.451	0.558	0.496
	CVD [17]	34.4	9.1	21.8	0.266	0.461	0.331
	SimRec [21]	**6.5**	7.8	**7.2**	**0.641**	0.581	**0.608**
Feature Fusion / Offline	Atlas [18]	8.4	10.2	9.3	0.565	0.598	0.578
	3DVNet [20]	22.1	**7.7**	14.9	0.506	0.545	0.520
	VoRTX [25]	**6.2**	8.2	**7.2**	**0.688**	**0.607**	**0.644**
Feature Fusion / Online	NeuRec [26]	**5.4**	12.8	9.1	0.684	0.479	0.562
	TF [2]	7.8	**9.9**	8.9	0.648	**0.547**	0.591
	Ours	5.5	10.5	**8.0**	**0.695**	0.527	**0.598**

Table 1. **Quantitative results of 3D metrics on ScanNet.** We show the results of two-stage depth fusion methods (top) and those for end-to-end feature fusion works (bottom) following the evaluation protocol in [26]. We highlight the best results for *Depth Fusion*, *Feature Fusion Offline* and *Feature Fusion Online* methods in blue, teal, and violet, respectively. Offline methods assume to observe the whole video sequence. Our method performs the best among all online feature fusion methods in the *Chamfer* metric.

Method	Acc↓	Comp↓	Chamfer↓	Prec↑	Recall↑	F-score↑
SimRec [21]	8.0	8.6	8.3	0.511	0.482	0.495
NeuRec [26]	6.1	19.4	12.8	0.588	0.347	0.431
Ours	**5.9**	**13.1**	**9.5**	**0.620**	**0.441**	**0.512**

Table 2. **Quantitative results of 3D metrics on 7-scenes.** We evaluate our method on the official test split of 7-scenes. All methods are trained on ScanNet and for baseline methods, we use their released pre-trained models.

		Acc↓	Comp↓	Chamfer↓	Prec↑	Recall↑	F-score↑
i	w/o any	5.6	14.1	9.9	0.661	0.441	0.527
ii	w resi	5.8	12.0	8.9	0.649	0.457	0.535
iii	w loc, resi	6.7	11.0	8.9	0.629	0.477	0.541
iv	w vis, resi	**5.4**	11.8	8.6	0.693	0.502	0.580
v	ours (top k)	5.6	11.6	8.6	0.673	0.495	0.569
vi	**ours** (sliding window)	5.5	**10.5**	**8.0**	**0.695**	**0.527**	**0.598**

Table 3. **Ablation study.** We ablate our visibility-based feature fusion ("vis"), ray-based local sparsification ("loc") and TSDF residual learning ("resi") on ScanNet.

instance-level semantic segmentations. We use the official train/val/test split to train and evaluate our method. For 7-Scenes, it contains 7 different scenes recorded from a handheld Kinect RGB-D camera. Each distinct scene is scanned several times by different users to generate multiple image sequences. We further evaluate our trained model on the official test split of 7-Scenes directly to demonstrate the generalization ability of our method. Details about datasets are included in the supplementary material.

Metrics. We follow the 3D geometry metrics used in [2, 18, 26] to evaluate the performance of our approach. The detailed definitions of these metrics are included in the supplementary material. Among these metrics, we regard the *Chamfer* distance as the most important metric because it is the average of accuracy, measuring the distance from predicted point clouds to the ground truth ones, and completeness, measuring the distance from ground truth point clouds
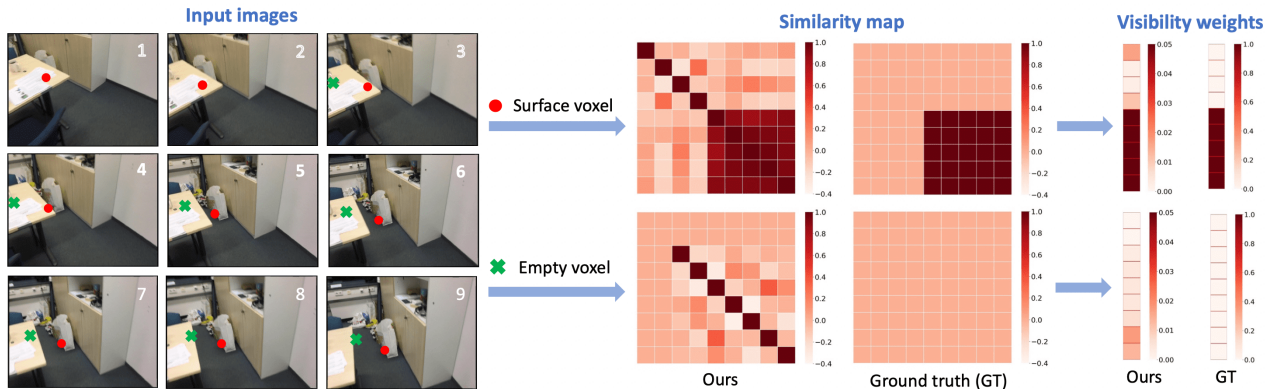
Figure 6. **Relations between the similarity map and the visibility weights.** Here, we illustrate this relationship using two kinds of voxels. The surface voxel (red dot) occupied by the paper bag on the floor is visible in the last 5 views. Thus the features extracted from those images have higher similarity with each other than those for the other views. So do the visibility weights. For the empty voxel (green cross), the features from different images are different leading to lower visibility weights.

to the predicted ones, for all 3D points. Although *F-score* also represents the balance between accuracy and completeness, it is also affected by a user-defined threshold (5cm).

Baselines. We compare our method with state-of-the-art depth-based methods and end-to-end reconstruction methods. For a fair comparison, these methods are trained and evaluated by following the official data split provided by the ScanNet dataset [8]. We use the pre-trained model for [9] and fine-tuned models for [11, 12, 32] provided by [9]. Following [21], the maximum fused depth is limited to 3m. The results of [2, 18, 20, 21, 25] are taken from [21] and [17, 22, 26] are taken from [26]. Among depth-based methods, MVDepthNet [32], GPMVS [11] and DeepVideoMVS [9] achieve real-time performance while GPMVS, DeepVideoMVS and CVD [17] are depth-fusion baselines with consistent video depth estimation. Neural-Recon [26] and TransformerFusion [2] are two end-to-end incremental volumetric reconstruction frameworks that directly predict the surface geometry and are the most relevant ones to our approach.

4.2. Results

ScanNet. The experimental results on the ScanNet dataset are reported in Tab. 1. Our method outperforms all existing online feature fusion methods [2, 26] in both *Chamfer* distance and *F-score* metrics. We achieve a 12.1% reduction in chamfer distance compared to NeuralRecon [26]. Although the concurrent work SimpleRecon [21] achieves slightly better numerical results, due to their two-stage pipeline, their results often include noises especially for low texture regions like walls as shown in Fig. 7. In contrast, our reconstruction results are more clean and coherent. We also compare our qualitative results to those of NeuralRecon [26] in Fig. 7. As highlighted in red boxes, our visibility-aware feature fusion and ray-based sparsification enable us to reconstruct more complete and detailed scene structures,

demonstrating the effectiveness of our approach.

7-Scenes. We also report the results on 7-Scenes in Tab. 2. The conclusion still holds. As the data split used by [26] is not available, we thus report the results of evaluating their pretrained model on the official test split of 7-Scenes. It may explain why the results of [26] are different from those reported in the original paper. More results on 7-Scenes are provided in the supplementary material.

Visibilities. In Fig. 6, we visualise the similarity map and learned visibility weights for surface voxels and empty voxels in the current local fragment, respectively. The similarity map shows that the photometric consistency between features from different views provides strong heuristics for voxel occupancy probability and view selection. From the visibility weights, we can find our local feature fusion module is able to select the most relevant views for each voxel to local occupancy prediction. More visualisations are shown in the supplementary material.

Ablation study. To demonstrate the effectiveness of each module of our method, we conduct several experiments on the ScanNet dataset and show the results in Tab. 3. Comparing row i and row ii, by adding the residual learning strategy, the chamfer distance is reduced by 10.1%. As shown in row iii and row iv, with our visibility predicting module and ray-based local sparsification strategy, our performance is further improved. Finally, we compare the effectiveness of using a sliding window to select consecutive K voxels, to simply choosing the top K voxels, along each ray in row v and row vi. It shows that utilising the sliding window for sparsification improves the performance.

Running time. Although due to the extra computation of our visibility-aware feature fusion and ray-based local sparsification, our method tends to be slower than neuralrecon [26], our model still achieves a real-time reconstruction of 25 key frames per second (FPS) on an NVIDIA RTX 2080Ti GPU and 45 FPS on an NVIDIA RTX 4090 GPU.
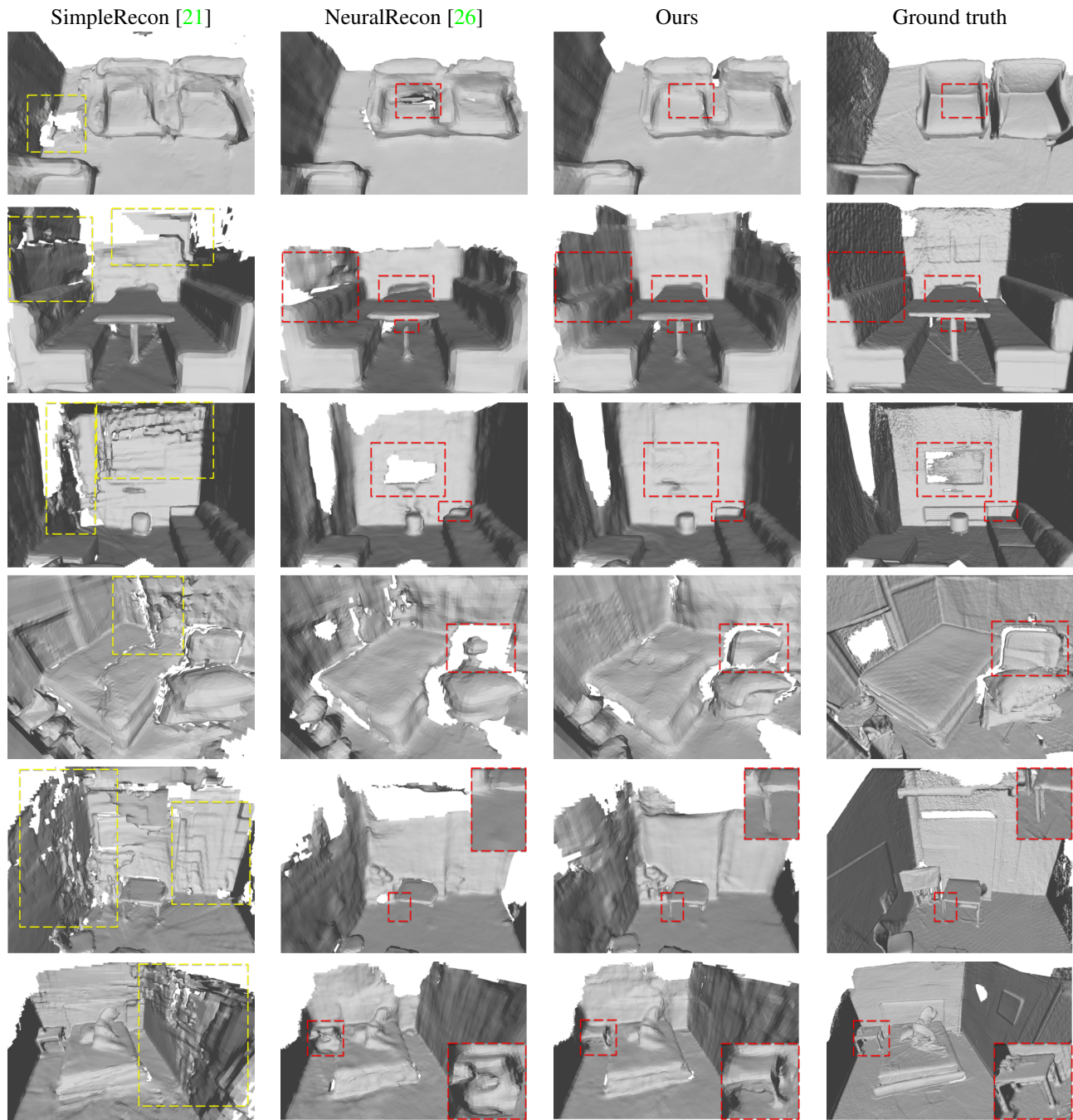
Figure 7. **Qualitative comparison on ScanNet.** Compared to NeuralRecon [26], our reconstruction results are more complete and contain more details (highlighted in the red boxes). Since SimpleRecon [21] is a two-stage depth-based method, it generates many artifacts and is not coherent (highlighted in the yellow boxes).

5. Conclusion

In this paper, we demonstrate that explicitly learning the visibility weights for feature fusion and adopting a ray-based local sparsification strategy will benefit the online incremental 3D scene reconstruction task. In the future, we will try to combine the advantages of depth estima-tion and end-to-end feature-based reconstruction to generate both sharper and coherent surface geometry.

References

[1] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11, pages 1–11, 2011. 1

[2] Aljaz Bozic, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. Transformerfusion: Monocular rgb scene reconstruction using transformers. *Advances in Neural Information Processing Systems*, 34:1403–1414, 2021. 1, 2, 3, 6, 7

[3] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 1

[4] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1538–1547, 2019. 3

[5] Jaesung Choe, Sunghoon Im, Francois Rameau, Minjun Kang, and In So Kweon. Volumefusion: Deep depth fusion for 3d scene reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16086–16095, 2021. 2

[6] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014. 3, 5

[7] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. 2

[8] 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. 6, 7

[9] Arda Duzceker, Silvano Galliani, Christoph Vogel, Pablo Speciale, Mihai Dusmanu, and Marc Pollefeys. Deepvideomvs: Multi-view stereo on video with recurrent spatiotemporal fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15324–15333, 2021. 2, 6, 7

[10] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2495–2504, 2020. 3

[11] Yuxin Hou, Juho Kannala, and Arno Solin. Multi-view stereo by temporal nonparametric fusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2651–2660, 2019. 1, 2, 3, 6, 7

[12] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. Dpsnet: End-to-end deep plane sweep stereo. *arXiv preprint arXiv:1905.00538*, 2019. 2, 6, 7

[13] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2307–2315, 2017. 2

[14] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10986–10995, 2019. 1, 2

[15] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7824–7833, 2022. 4

[16] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 1

[17] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (ToG)*, 39(4):71–1, 2020. 6, 7

[18] 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 *European conference on computer vision*, pages 414–431. Springer, 2020. 1, 2, 3, 4, 5, 6, 7

[19] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011. 1, 2

[20] Alexander Rich, Noah Stier, Pradeep Sen, and Tobias Höllerer. 3dvnet: Multi-view depth prediction and volumetric refinement. In *2021 International Conference on 3D Vision (3DV)*, pages 700–709. IEEE, 2021. 2, 6, 7

[21] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. Simplerecon: 3d reconstruction without 3d convolutions. In *European Conference on Computer Vision*, pages 1–19. Springer, 2022. 2, 6, 7, 8

[22] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European conference on computer vision*, pages 501–518. Springer, 2016. 2, 6, 7

[23] Thomas Schöps, Torsten Sattler, Christian Häne, and Marc Pollefeys. 3d modeling on the go: Interactive 3d reconstruction of large-scale scenes on mobile devices. *3DV*, 1(2):3–2, 2015. 1

[24] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2930–2937, 2013. 6

[25] Noah Stier, Alexander Rich, Pradeep Sen, and Tobias Höllerer. Vortx: Volumetric 3d reconstruction with transformers for voxelwise view selection and fusion. In *2021 International Conference on 3D Vision (3DV)*, pages 320–330. IEEE, 2021. 2, 3, 4, 6, 7

[26] 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. 1, 2, 3, 4, 5, 6, 7, 8

[27] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 4, 6

[28] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, pages 685–702. Springer, 2020. 6

[29] Julien Valentin, Adarsh Kowdle, Jonathan T Barron, Neal Wadhwa, Max Dzitsiuk, Michael Schoenberg, Vivek Verma, Ambrus Csaszar, Eric Turner, Ivan Dryanovski, et al. Depth from motion for smartphone ar. *ACM Transactions on Graphics (ToG)*, 37(6):1–19, 2018. 1

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3

[31] Lukas Von Stumberg, Vladyslav Usenko, and Daniel Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2510–2517. IEEE, 2018. 1

[32] Kaixuan Wang and Shaojie Shen. Mvdepthnet: Real-time multiview depth estimation neural network. In *2018 International conference on 3d vision (3DV)*, pages 248–257. IEEE, 2018. 1, 2, 6, 7

[33] 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. 1, 3

[34] Jiayu Yang, Wei Mao, Jose M Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4877–4886, 2020. 3

[35] Zhenfei Yang, Fei Gao, and Shaojie Shen. Real-time monocular dense mapping on aerial robots using visual-inertial fusion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4552–4559. IEEE, 2017. 1

[36] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783, 2018. 2