Supplementary Material

A. Overview

This document provides additional information that we did not fully cover in the main paper, ***VolumeFusion: Deep Depth Fusion for 3D Scene Reconstruction***. All references are consistent with the main paper.

B. Evaluation Metrics

In this section, we describe the metrics for depth and 3D reconstruction evaluation. Depth evaluation involves four different metrics: Abs Rel, Abs Diff, Sq Rel, and RMSE. Each of these metrics is calculated as:

$$\text{Abs Rel} = \frac{1}{n}\sum_{(u,v)} \left|z_{(u,v)} - \tilde{z}_{(u,v)}\right|_1 / \tilde{z}_{(u,v)}, \tag{8a}$$

$$\text{Abs Diff} = \frac{1}{n}\sum_{(u,v)} \left|z_{(u,v)} - \tilde{z}_{(u,v)}\right|_1, \tag{8b}$$

$$\text{Sq Rel} = \frac{1}{n}\sum_{(u,v)} \left|z_{(u,v)} - \tilde{z}_{(u,v)}\right|^2 / \tilde{z}_{(u,v)}, \tag{8c}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{(u,v)} \left|z_{(u,v)} - \tilde{z}_{(u,v)}\right|^2}, \tag{8d}$$

where n is the number of pixels within both valid ground-truth and predictions, $z_{(u,v)}$ is the depth value at the pixel (u, v) in the ground-truth depth map z, $\tilde{z}_{(u,v)}$ is the depth value at the pixel (u, v) in the inferred depth map $\tilde{z}$, and $|\cdot|_1$ represents absolute distance. Note that the inferred depth map of our method is the final fused depth results after the scene reconstruction.

Regarding the 3D geometry evaluation, we propose four different metrics: $\mathcal{L}_1$, accuracy (Acc), completeness (Comp), and F-score. $\mathcal{L}_1$ is the absolute difference between the ground-truth TSDF and the inferred TSDF, the accuracy is the distance from predicted points to the ground-truth points, completeness is the distance from the ground-truth points to the predicted points, and F-score is the harmonic mean of the precision and recall. Each of these metrics is calculated as:

$$\mathcal{L}_1 = \text{mean}_{a<1} \left|a - \tilde{a}\right|_1 \tag{9a}$$

$$\text{Acc} = \text{mean}_{\tilde{\mathbf{p}}\in\tilde{\mathcal{P}}}(\text{min}_{\mathbf{p}\in\mathcal{P}} \left|\mathbf{p} - \tilde{\mathbf{p}}\right|_1) \tag{9b}$$

$$\text{Comp} = \text{mean}_{\mathbf{p}\in\mathcal{P}}(\text{min}_{\tilde{\mathbf{p}}\in\tilde{\mathcal{P}}} \left|\mathbf{p} - \tilde{\mathbf{p}}\right|_1) \tag{9c}$$

$$\text{F-score} = \frac{2 \times \text{Prec} \times \text{Recall}}{\text{Prec} + \text{Recall}} \tag{9d}$$

where $\tilde{a}$ is the predicted TSDF value, a is the ground-truth TSDF value, $\mathbf{p}$ is a point within a set of the ground-truth point clouds $\mathcal{P}$, $\tilde{\mathbf{p}}$ is a point within a set of predicted point clouds $\tilde{\mathcal{P}}$, Prec is the precision metric (*i.e.*, Prec=$\text{mean}_{\tilde{\mathbf{p}}\in\tilde{\mathcal{P}}}(\text{min}_{\mathbf{p}\in\mathcal{P}} |\mathbf{p}\text{-}\tilde{\mathbf{p}}|_1 <0.05)$), and Recall is the recall metric (*i.e.*, Recall=$\text{mean}_{\mathbf{p}\in\mathcal{P}}(\text{min}_{\tilde{\mathbf{p}}\in\tilde{\mathcal{P}}} |\mathbf{p}\text{-}\tilde{\mathbf{p}}|_1 <0.05)$)

C. Implementation and Training Scheme

In this section, we provide clarifications and precision about the implementation and training scheme of our volume fusion network. The size of the n-th initial feature volume $\mathcal{V}^{\mathcal{I}}_{\text{n}}$ is $\mathbb{R}^{96(3\text{C})\times 48(\text{D})\times 120(\text{H})\times 160(\text{W})}$. Note that the size of the volumes in the first stage is consistent during the training and testing session, however, we set different resolutions of the feature volume $\mathcal{V}^{F}_{\text{n}}$, the pose-invariant scene volume $\mathcal{V}^{R}_{\text{n}}$, the unified scene volume$\mathcal{V}^{U}$, and the TSDF scene volume $\tilde{\mathcal{V}}^{\text{TSDF}}$. The sizes of both the feature volume $\mathcal{V}^{F}_{\text{n}}$ and the pose-invariant scene volume $\mathcal{V}^{R}_{n}$ are $\mathbb{R}^{32(C)\times 160(\text{V}_\text{x})\times 160(\text{V}_\text{y})\times 48(\text{V}_\text{z})}$ for training and $\mathbb{R}^{32(\text{C})\times 640(\text{V}_\text{x})\times 640(\text{V}_\text{y})\times 128(\text{V}_\text{z})}$ for testing and evaluation.

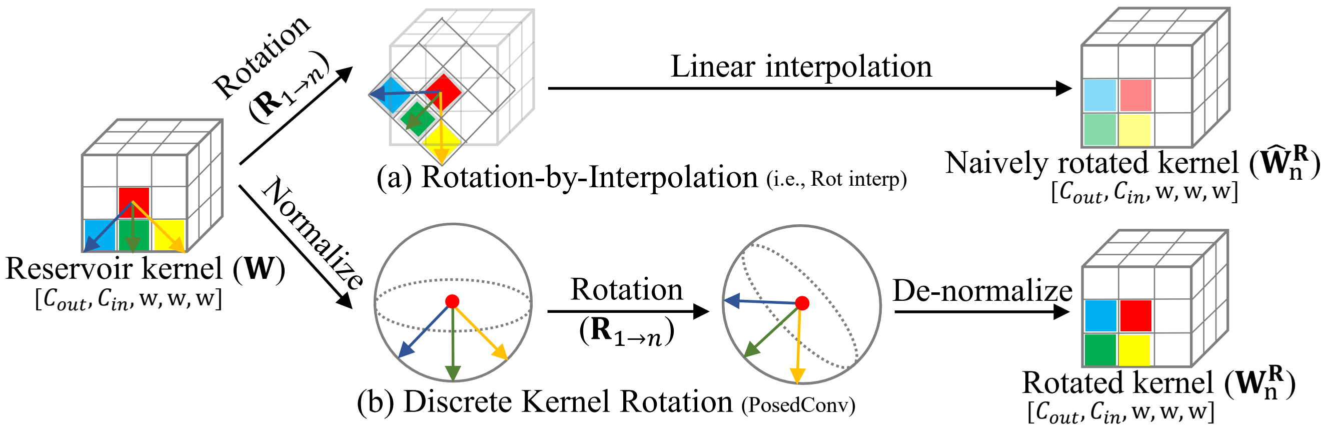
As depicted in Fig. 2 of the manuscript, the unified scene volume $\mathcal{V}^{U}$ obeys $\mathbb{R}^{33(\text{C}+1)\times 160(\text{V}_\text{x})\times 160(\text{V}_\text{y})\times 48(\text{V}_\text{z})}$ for a training session and $\mathbb{R}^{33(\text{C}+1)\times 640(\text{V}_\text{x})\times 640(\text{V}_\text{y})\times 128(\text{V}_\text{z})}$ for a testing period. Note that the unified scene volume has one more channel than the pose-invariant scene volume $\mathcal{V}^{R}_{\text{n}}$. This is because we embed back-projected point clouds from masked depth maps $\hat{\mathcal{Z}}$. After we propagate $\mathcal{V}^{U}$ for depth fusion, the dimensions of the TSDF scene volume $\tilde{\mathcal{V}}^{\text{TSDF}}$ are $\mathbb{R}^{160(\text{V}_\text{x})\times 160(\text{V}_\text{y})\times 48(\text{V}_\text{z})}$ for a training period and $\mathbb{R}^{640(\text{V}_\text{x})\times 640(\text{V}_\text{y})\times 128(\text{V}_\text{z})}$ for a testing session. Note that we set the identical resolution of the final TSDF scene volume as proposed in Murez *et al.* [32] for a fair comparison. The size of the batch per GPU is set as 1. For training, each batch consists of 45 multi-view images (*i.e.*, 15 reference views and 30 neighbor views). We use all images for test.

D. Discrete Kernel Rotation

This section describes the details of *PosedConv*. The idea of the *PosedConv* is to rotate the reservoir kernel by using the known camera poses to extract rotation-invariant features (Sec. 3.3 of the manuscript).

The rotated kernel can be simply computed by rotation followed by linear interpolation, called Rotation-by-Interpolation, as shown in Fig. 7-(a). However, the naively rotated kernels $\hat{\mathbf{W}}^{R}_{n}$ often fail to interpolate properly rotated reservoir kernel values because of the different radius from the center to the boundary. Thus, we design a discrete kernel rotation performed on a unit sphere to minimize the loss of boundary kernel information as shown in Fig. 7-(b), called *Discrete Kernel Rotation*.

In detail, we first transform the 3D cube into a unit sphere by normalizing its distance from the center of the kernel. Within this unit sphere, the rotated sphere also be-



Rotation ($\mathbf{R}_{1\to n}$)
Linear interpolation
(a) Rotation-by-Interpolation (i.e., Rot interp)
Naively rotated kernel ($\widehat{\mathbf{W}}_{\mathrm{n}}^{\mathbf{R}}$)
$[C_{out}, C_{in}, \mathrm{w}, \mathrm{w}, \mathrm{w}]$
Reservoir kernel ($\mathbf{W}$)
$[C_{out}, C_{in}, \mathrm{w}, \mathrm{w}, \mathrm{w}]$
Normalize
Rotation ($\mathbf{R}_{1\to n}$)
De-normalize
(b) Discrete Kernel Rotation (PosedConv)
Rotated kernel ($\mathbf{W}_{\mathrm{n}}^{\mathbf{R}}$)
$[C_{out}, C_{in}, \mathrm{w}, \mathrm{w}, \mathrm{w}]$

Figure 7. **Discrete Kernel Rotation in our *PosedConv*.** From the reservoir kernel $\mathbf{W}$, (a) Rotation-by-Interpolation produces naively posed kernel $\widehat{\mathbf{W}}_{\mathrm{n}}^{\mathbf{R}}$, (b) discrete kernel rotation (ours) produces the rotated kernel $\mathbf{W}_{\mathrm{n}}^{\mathbf{R}}$.

Algorithm 1 Discrete Kernel Rotation

```
Require: Reservoir kernel $\mathbf{W} \in \mathbb{R}^{C_{\mathrm{out}}\times C_{\mathrm{in}}\times w\times w\times w}$, rotation matrix $\mathbf{R}_{\mathrm{n}\to 1}$.
1: procedure DISCRETE KERNEL ROTATION($\mathbf{W}$, $\mathbf{R}_{\mathrm{n}\to 1}$)
2:     Declare rotated kernel $\mathbf{W}_{\mathrm{n}}^{\mathbf{R}} \in \mathbb{R}^{C_{\mathrm{out}}\times C_{\mathrm{in}}\times w\times w\times w}$
3:     for $[i,j,k]$ in $\mathbf{W}_{\mathrm{n}}^{\mathbf{R}}$ do
4:         $[X_{\mathrm{s}}, Y_{\mathrm{s}}, Z_{\mathrm{s}}] \leftarrow$ NORM($[i,j,k]$, $w$)    ▷ $NORM(\cdot)$ in Alg. 2
5:         $[\hat{X}_{\mathrm{s}}, \hat{Y}_{\mathrm{s}}, \hat{Z}_{\mathrm{s}}, 1]^{\intercal} \leftarrow \mathbf{R}_{\mathrm{n}\to 1}[X_{\mathrm{s}}, Y_{\mathrm{s}}, Z_{\mathrm{s}}, 1]^{\intercal}$
6:         $[\hat{i}, \hat{j}, \hat{k}] \leftarrow$ DeNorm($[i,j,k]$, $w$, $[\hat{X}_{\mathrm{s}}, \hat{Y}_{\mathrm{s}}, \hat{Z}_{\mathrm{s}}]$) ▷ DENORM($\cdot$) in Alg. 3
7:         $\mathbf{W}_{\mathrm{n}}^{\mathbf{R}}(i,j,k) \leftarrow$ Bilinear($\mathbf{W}$, $[\hat{i}, \hat{j}, \hat{k}]$)
8:     end for
9: end procedure
```

Algorithm 2 Normalization Function for Discrete Kernel Rotation

```
Require: Voxel coordinate $[i,j,k]$, kernel size $w$.
Ensure: Normalized image coordinate $[X_{\mathrm{s}}, Y_{\mathrm{s}}, Z_{\mathrm{s}}]$.
1: procedure NORM($[i,j,k]$, w)
2:     $r \leftarrow \frac{w\text{-}1}{2}$                                   ▷ Kernel radius $r$
3:     $\ell \leftarrow \sqrt{((i\text{-}r)^2 + (j\text{-}r)^2 + (k\text{-}r)^2)}$
4:     $[X_{\mathrm{s}}, Y_{\mathrm{s}}, Z_{\mathrm{s}}] \leftarrow [\frac{i\text{-}r}{\ell}, \frac{j\text{-}r}{\ell}, \frac{k\text{-}r}{\ell}]$ where $0 \le X_{\mathrm{s}}, Y_{\mathrm{s}}, Z_{\mathrm{s}} \le 1.0$
       return $[X_s, Y_s, Z_s]$
5: end procedure
```

Algorithm 3 DeNormalization Function for Discrete Kernel Rotation

```
Require: Voxel coordinate $[i,j,k] \in \mathbf{W}_{\mathrm{n}}^{\mathbf{R}}$, kernel size $w$, normalized image coordinate $[\hat{X}_{\mathrm{s}}, \hat{Y}_{\mathrm{s}}, \hat{Z}_{\mathrm{s}}]$ where $0.0 \le \hat{X}_{\mathrm{s}}, \hat{Y}_{\mathrm{s}}, \hat{Z}_{\mathrm{s}} \le 1.0$.
Ensure: Rotated voxel coordinate $[\hat{i}, \hat{j}, \hat{k}] \in \mathbf{W}$.
1: procedure DENORM($[i,j,k]$, $w$, $[\hat{X}_{\mathrm{s}}, \hat{Y}_{\mathrm{s}}, \hat{Z}_{\mathrm{s}}]$)
2:     $r \leftarrow \frac{w\text{-}1}{2}$                                   ▷ Kernel radius $r$
3:     $\ell \leftarrow \sqrt{(i\text{-}r)^2 + (j\text{-}r)^2 + (k\text{-}r)^2}$
4:     $[\hat{i}, \hat{j}, \hat{k}] \leftarrow [\ell\cdot\hat{X}_{\mathrm{s}}{+}r,\ \ell\cdot\hat{Y}_{\mathrm{s}}{+}r,\ \ell\cdot\hat{Z}_{\mathrm{s}}{+}r]$ where $0 \le i,j,k \le w$
       return $[\hat{i}, \hat{j}, \hat{k}]$
5: end procedure
```

comes a unit sphere so that we can alleviate the information drops during the rotation. After rotating the unit sphere, we denormalize the radius of the sphere to form the 3D cube.

This process rotates the reservoir kernel $\mathbf{W}$ of the n-th camera view by using the rotation matrix $\mathbf{R}_{1\to\mathrm{n}}$, as depicted in Fig. 7 of the main paper. Concretely, we iteratively apply the discrete kernel rotation for each n-th feature volume $\mathcal{V}_{\mathrm{n}}^{\mathcal{F}}$, as in Fig. 2 of the main paper. Given the reservoir kernel $\mathbf{W}$ and the rotation matrix $\mathbf{R}_{1\to\mathrm{n}}$, the algorithm infers the rotated kernel $\mathbf{W}_{\mathrm{n}}^{\mathbf{R}}$ as in Alg. 1. To densely fill the rotated kernel $\mathbf{W}_{\mathrm{n}}^{\mathbf{R}}$ with the reservoir kernel $\mathbf{W}$, we need to repetitively apply the reverse warping process.

Each voxel $\mathbf{v}'$ at the voxel coordinate $[i,j,k]$, is transformed into its position on the unit sphere $[X_{\mathrm{s}}, Y_{\mathrm{s}}, Z_{\mathrm{s}}, 1]^{\intercal}$ through the NORM($\cdot$) function (Line 4 in Alg. 1). The details of the NORM($\cdot$) function is described in Alg. 2 and this is the fundamental difference between our Discrete Kernel Rotation (Fig. 7-(b) of the main paper and Alg. 1) and rotation-by-interpolation (Fig. 7-(a) of the main paper and Alg. 4). Since this is a reverse warping process, we use the given rotation matrix $\mathbf{R}_{\mathrm{n}\to 1}$ (Line 5 in Alg. 1) that is the inverse of the rotation matrix $\mathbf{R}_{1\to\mathrm{n}}$ as depicted in Fig. 7 of the main paper. To do so, we obtain the rotated location $[\hat{X}_{\mathrm{s}}, \hat{Y}_{\mathrm{s}}, \hat{Z}_{\mathrm{s}}, 1]^{\intercal}$ lying on the unit sphere (Line 5 in Alg. 1).

To find the corresponding location in the reservoir kernel, we denormalize the rotated location $[\hat{X}_{\mathrm{s}}, \hat{Y}_{\mathrm{s}}, \hat{Z}_{\mathrm{s}}, 1]^{\intercal}$ and obtain the rotated voxel coordinate $[\hat{i}, \hat{j}, \hat{k}]$ at the reservoir kernel $\mathbf{W}$ (Line 6 in Alg. 1) where DeNorm($\cdot$) function is precisely described in Alg. 3. Since the rotated voxel coordinate is not located outside of the boundary of the reservoir kernel $\mathbf{W}$, we directly apply bilinear interpolation to extract the kernel response at $[\hat{i}, \hat{j}, \hat{k}]$ at the reservoir kernel $\mathbf{W}$ (Line 7 in Alg. 1). Contrary to our discrete kernel rotation, naive rotation-by-interpolation must check whether the rotated voxel coordinate $[\hat{i}, \hat{j}, \hat{k}]$ is out of the boundary

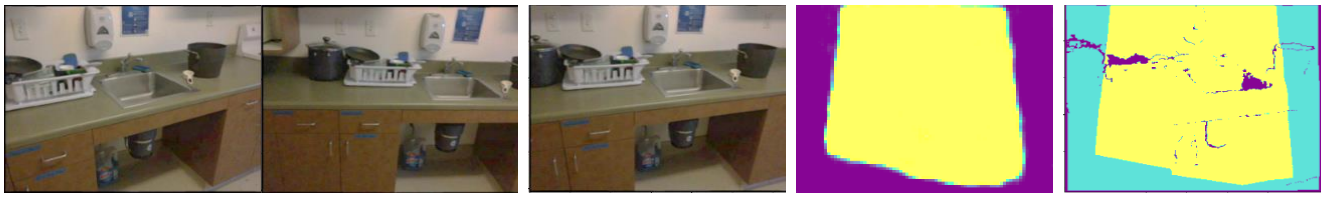

(a) A set of 2 neighbor views $\{\mathcal{I}_{\text{n-1}}, \mathcal{I}_{\text{n+1}}\}$ (b) Reference view $\mathcal{I}_{\text{n}}$ (c) Predicted overlapping mask $\widetilde{M}_n$ (d) Ground truth overlapping mask M_n

Figure 8. **Illustration of an overlapping mask.** Given (a) 2 neighbor views and (b) a reference frame, our multi-view stereo network (Fig. 2 of the manuscript) infers a depth map $\tilde{\mathcal{Z}}$ and (c) a overlapping mask $\tilde{\mathcal{M}}$. A true overlapping mask $\mathcal{M}$ is a binary mask that 1 represents valid overlapping pixels (yellow pixels in (d)) and 0 means non-overlapping pixels (cyan pixels in (d)). We can determine the overlapping pixels as pixels transformed from neighbor images (a) to the referential image (b) using true ground truth depth maps $\mathcal{Z}$ and ground truth camera poses $[\mathbf{R}|\mathbf{t}]$. Since there are unknown true depth values within the ground truth depth maps $\mathcal{Z}$, we cannot determine the overlapping or non-overlapping pixels that are colorized as purple in (d). After training our network, the obtained overlapping mask $\tilde{\mathcal{M}}_{\text{n}}$ is visualized in (c) where yellow pixels indicate predicted overlapping pixels and purple pixels mean estimated non-overlapping pixels.

Method	Evaluation			
	AbsRel	RMSE	$\mathcal{L}_1$	F-score
3D Conv	.058	.231	.166	.460
Rot interp	.056	.223	.159	.487
Disc K Rot	**.049**	**.164**	**.141**	**.508**

Table 5. **Ablation study of PosedConv.** We compare original convolution layer (*i.e.*, 3D Conv), and the Rotation-by-Interpolation (*i.e.*, Rot interp), and our PosedConv (*i.e.*, Disc K Rot). It shows that the kernel rotation is much effective thank the original 3D Conv, but our *PosedConv* further improve the quality of the 3D scene reconstruction.

Algorithm 4 Rotation-by-Interpolation

```
Require: Reservoir kernel $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times w \times w \times w}$, rotation matrix $\mathbf{R}_{n \to 1}$.
 1: procedure ROTATION-BY-INTERPOLATION($\mathbf{W}$, $\mathbf{R}_{\text{n}\to 1}$)
 2:     Declare rotated kernel $\mathbf{W}^{\mathbf{R}}_{\text{n}} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times w \times w \times w}$
 3:     for $[i, j, k]$ in $\mathbf{W}^{\mathbf{R}}_{\text{n}}$ do
 4:         $r \leftarrow \frac{w\text{-}1}{2}$                                  ▷ Kernel radius $r$
 5:         $[X, Y, Z] \leftarrow [i\text{-}r, j\text{-}r, k\text{-}r]$          ▷ Signed distance
 6:         $[\hat{X}, \hat{Y}, \hat{Z}, 1]^{\intercal} \leftarrow \mathbf{R}_{\text{n}\to 1}[X, Y, Z, 1]^{\intercal}$
 7:         $[\hat{i}, \hat{j}, \hat{k}] \leftarrow [\hat{X}+r, \hat{Y}+r, \hat{Z}+r])$
 8:         if $[\hat{i}, \hat{j}, \hat{k}] \notin \mathbf{W}$ then                ▷ Boundary check
 9:             $[\hat{i}, \hat{j}, \hat{k}] \leftarrow [\tilde{i}, \tilde{j}, \tilde{k}]$
10:             where $[\tilde{i}, \tilde{j}, \tilde{k}] \in \mathbf{W}$ and $[\hat{i}, \hat{j}, \hat{k}]$ is nearest to $[\tilde{i}, \tilde{j}, \tilde{k}]$.
11:         end if
12:         $\mathbf{W}^{\mathbf{R}}_{\text{n}}(i, j, k) \leftarrow$ Bilinear($\mathbf{W}$, $[\hat{i}, \hat{j}, \hat{k}]$)
13:     end for
14: end procedure
```

of the reservoir kernel $\mathbf{W}$, or it sometimes has difficulty in interpolating the reservoir kernel $\mathbf{W}$ into the rotated kernel $\mathbf{W}^{\mathbf{R}}_{\text{n}}$ (Lines 8-10 in Alg. 4).

Finally, we compute the rotated kernel $\mathbf{W}^{\mathbf{R}}_{\text{n}}$ for the n-th camera view. As described in Sec. 3.3 of the main paper, we identically apply the conventional 3D convolution operation that is equipped with our rotated kernels $\mathbf{W}^{\mathbf{R}}_{\text{n}}$.

To validate the necessity of our discrete kernel rotation, we conduct an ablation study as in Table 5. Alongside the results in Table 2 of the manuscript, we additionally report the accuracy when we use naive Rotation-by-Interpolation (*i.e.*, Rot interp). It shows that the rotating the kernel in both ways (Rot interp and discrete kernel rotation) improves the quality of depth and 3D geometry, but our PosedConv further achieves the higher accuracy than that of Rotation-by-Interpolation.

E. Overlapping Mask

In this section, we present an example figure of an overlapping mask as in Fig. 8. In the first stage of our network, multi-view stereo, we utilize three neighbor views to infer a depth map and an overlapping mask. This overlapping mask is used to filter out the uncertain depth values at the specific pixels that have no corresponding pixels in the neighbor views. As shown in Fig. 8, our network properly infers the overlapping mask in a referential camera viewpoint. Note that the referential image is in between two adjacent views, the overlapping region is usually located at the center of the referential images.

F. Combined Ablation Results.

To clearly show influences from our contributions, we merge ablation results in the manuscripts as in Table 6. The quality of the reconstruction is largely improved with both *PosedConv* and overlapping masks, simultaneously. This is because overlapping masks are designed to filter out the uncertain depth values that can hurt the quality of 3D reconstruction done by PosedConv. In conclusion, our two novel contributions, PosedConv and overlapping masks, are complementary and lead to precise 3D scene reconstruction.

Preserve (✓)				Performance			
Depth fusion	Conv type		Overlap	2D depth		3D geometry	
	Conv	PosedConv		AbsRel	RMSE	$\mathcal{L}_1$	F-score
				.061	.248	.162	.499
		✓		.060	.245	.160	.495
✓		✓		.060	.238	.162	.475
✓	✓		✓	.058	.231	.166	.460
✓		✓	✓	**.049**	**.164**	**.141**	**.508**

Table 6. **Merged ablation results on ScanNet dataset.**

Method	2D Depth		3D Geometry	
	AbsRel	RMSE	Acc	F-score
CNMNet (ECCV'20)	.161	.361	.398	.149
NeuralRecon (CVPR'21)	.155	.347	.100	**.228**
VolumeFusion (ours)	**.140**	**.320**	**.085**	.217

Table 7. **Quantitative results on 7-Scenes dataset.**

G. Additional Results

We conduct quantitative evaluations on the 7-Scenes dataset (Shotton *et al.*, CVPR'13) in Table 7. Similarly, our network achieves state-of-the-art performance against the recent approaches. For a fair comparison, we strictly follow the assessment pipelines of these concurrent approaches ([38] and Long *et al.* in ECCV'20).