CVPR
#7643

MonoSelfRecon: Purely Self-Supervised Explicit Generalizable 3D Reconstruction of Indoor Scenes from Monocular RGB Views

Supplementary Material

6. Relationship with MonoNeRF [8]

We discuss the relationship between our strongest baseline MonoNeRF[8] and our proposed method MonoSelfRecon as follows: 1) We share the same idea of SFM-based 3DR with monocular RGB sequence as input, and we both jointly train SFM and a generalizable NeRF, where the NeRF is used to boost SFM performance. 2) Although using SFM as the core of framework design, we regress to different 3D representations, where MonoNeRF regress to view-based 2D depth map while we regress to 3D voxel-based SDF values. 3) While MonoNeRF also jointly estimates camera poses, their 2D view-based depth representation restricts the ability to incrementally complete a whole scene in 3D mesh representation. Fusing TSDF from direct depth estimation is time-consuming, and will cause layered or sparse mesh due to depth inconsistency between each frame. By comparison, our direct voxel-SDF regression enables us to incrementally add the previous mesh to complete the whole scene consistently in mesh representation.

The mesh representation is a stricter 3D representation over 2D depth map. Theoretically, the depth map can be perfectly rendered from 3D mesh but cannot in reverse, which is further validated by our experiments. Table 2 and 3 show that although all using ground truth for supervised training, the one that directly regresses SDF (NeuralRecon) has a clear advantage on 2D depth metrics over other supervised methods that regresses depth. The reason that although both our method and MonoNeRF are based on SFM while ours outperforms theirs can be also partly attributed to this different 3D representation. Our visual results also reflect this point in Table 3, where although there is no much difference of 2D depth, the difference of 3D mesh is clear. In other words, the depth representation is more visually straightforward than 3D mesh. Consequently, our 3D mesh regressing is a stricter 3D geometric representation than MonoNeRF's 2D depth. **We will release our code soon after the paper acceptance**.

7. Evaluation Metrics

We follow the same evaluation metrics as [27, 36]. Details of the metrics are summarized in Table 5.

2D		3D	
Abs Rel	$\frac{1}{n}\sum \lvert d-d^*\rvert / d^*$	Acc	$\text{mean}_{p\in P}\left(\min_{p^*\in P^*}\lVert p-p^*\rVert\right)$
Abs Diff	$\frac{1}{n}\sum \lvert d-d^*\rvert$	Comp	$\text{mean}_{p^*\in P^*}\left(\min_{p\in P}\lVert p-p^*\rVert\right)$
Sq Rel	$\frac{1}{n}\sum \lvert d-d^*\rvert^2 / d^*$	Prec	$\text{mean}_{p\in P}\left(\min_{p^*\in P^*}\lVert p-p^*\rVert < .05\right)$
RMSE	$\sqrt{\frac{1}{n}\sum \lvert d-d^*\rvert^2}$	Recal	$\text{mean}_{p^*\in P^*}\left(\min_{p\in P}\lVert p-p^*\rVert < .05\right)$
$\sigma < 1.25$	$\frac{1}{n}\sum\left(\max\left(\frac{d}{d^*},\frac{d^*}{d}\right) < 1.25\right)$	F-score	$\frac{2\times\text{Prec}\times\text{Recal}}{\text{Prec}+\text{Recal}}$
Comp	% valid predictions		
RMSE log	$\sqrt{\frac{1}{n}\sum \lvert\log(d)-\log(d^*)\rvert^2}$		
Sc Inv	$\left(\frac{1}{n}\sum_i z_i^2-\frac{1}{n^2}\left(\sum_i z_i\right)^2\right)^{1/2}$		

Table 5. **Evaluation Metrics.**

8. Model Details

8.1. Attentional View Fusion

We use a standard Vision Transformer (ViT) Encoder, where we keep the original high-level architecture of the ViT encoder to be: A norm layer, a multi-head attention layer, a norm layer, and a MLP (2 heads are used). Originally the ViT takes image patch/features as input, while we adopted the input to be the nearest 2D features from the projected 3D voxels, which is of size $[N_{view}, N_{points}, C]$, where N_{view} is the number of views in a scene fragment, N_{points} is the number of voxels in a fragment, C is the feature channel. The input also takes the voxel mask as input to filter out the pixels which are invisible to the voxels, and the transformer only takes the visible pixel features. We stack two ViT encoders to update the features, where the output is still of size $[N_{view}, N_{points}, C]$. Then we use a multi-view weighted feature pooling to fuse the updated features at the view channel to 3D features of size $[N_{points}, C]$, where the weight is the number of visible views in a fragment for each voxel. Such design enables more flexibility to adjust the contribution of each view to the 3D voxels.

8.2. GRU

We directly use the GRU module from [36], which is elaborately designed for sparse 3D convolution. The 3D voxel features are obtained by attentional view fusions and fed to the GRU module, where the current 3D fragment features are conditioned on the previous fragment. Using the current 3D global voxel features G_t^l and the previous hidden state H_{t-1}^l at layer l, the current hidden state H_t^l can be obtained, and the SDF value at each level is regressed from the hidden state H_t^l. More specifically,

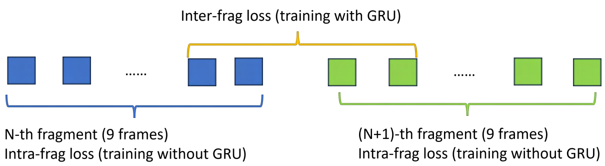


Figure 4. **Inter/Intra-fragment** losses illustration.

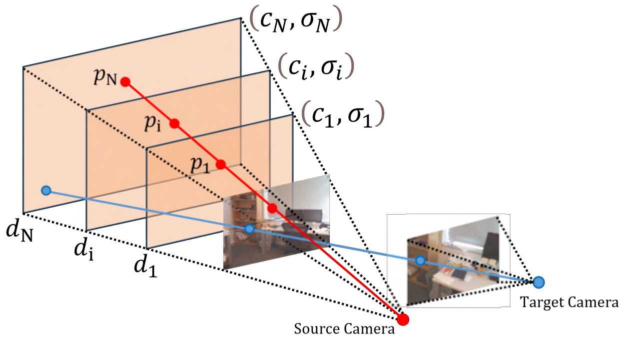


Figure 5. **Multi-Plane Image (MPI)** NeRF illustration.

$$
\begin{aligned}
z_t &= \sigma(SparseConv([H_{t-1}^l, G_t^l], W_z)) \\
r_t &= \sigma(SparseConv([H_{t-1}^l, G_t^l], W_r)) \\
\tilde{H_t^l} &= tanh(SparseConv([r_t \odot H_{t-1}^l, G_t^l], W_h)) \\
H_t^l &= (1 - z_t) \odot H_{t-1}^l + z_t \odot \tilde{H_t^l}
\end{aligned}
\tag{12}
$$

where z_t is the update gate, r_t is the reset gate, σ is the sigmoid function and W_* is the weight for sparse convolution.

We first train without GRU within each fragment to warmup the framework with our proposed self-supervised losses, where we call it **intra-fragment losses**. Because the GRU module is leveraged to enhance the consistency between fragments, the self-supervised learning strategy should be treated differently to intra-fragment losses. There is no need to change the training policy in purely supervised training because SDF ground truth is used, and there is no ground truth in our self-supervision except for the consistency clues between fragments. So we extend the **inter-fragment losses** to **intra-fragment losses**. While the model only takes input per fragment, backpropagating whole fragments brings memory challenges, so we only implement the inter-fragment losses on the frames around the boundary of fragments. Specifically, we use the last 4 and first 4 frames of the previous and current fragments to implement the inter-fragment loss.

8.3. NeRF

Since the SDF decoder is generalizable, the NeRF also must be generalizable to boost SDF decoder. For our work, we adopted MPI-NeRF[16, 57], which has been directly used by MonoNeRF[8] and proved to be generalizable. As Figure 5 shows, in Multi-Plane-Image (MPI) system, an image is represented by a set of parallel planes (orange planes) denoted as RGB-σ, specifically $(c_i, \sigma_i)_{i=1}^N$, where the i_{th} plane has d_i disparity (reverse of depth) to the camera. The shading points (red points) are selected as the intersection of the parallel planes and the rays shooting from pixels in the image, where c_i and σ_i are the RGB color and density of each shading points at i_{th} plane. In a standard MPI system, the source view RGB image $\hat{I}_s$ and depth map $\hat{D}$ can be composed using the "over" operation [31] as

$$
\begin{aligned}
\hat{I}_s &= \sum_{i=1}^{D} (c_i \sigma_i \prod_{j=i+1}^{D} (1 - \sigma_j)) \\
\hat{D}_s &= \sum_{i=1}^{D} (d_i^{-1} \sigma_i \prod_{j=i+1}^{D} (1 - \sigma_j))
\end{aligned}
\tag{13}
$$

To use MPI system in NeRF style, the composition operation above can be replaced by volumetric rendering [25] for both RGB and depth as

$$
\begin{aligned}
\hat{I}_s &= \sum_{i=1}^{N} T_i (1 - exp(-\sigma_i \delta_i)) c_i \\
\hat{Z}_s &= \sum_{i=1}^{N} T_i (1 - exp(-\sigma_i \delta_i)) z_i
\end{aligned}
\tag{14}
$$

where z_i is the rendered depth (reverse of disparity) $z_i = 1/d_i$, and $\delta_i = ||p_{i+1} - p_i||_2$ is the distance between the two neighbor shading points on a ray. Then we can extend volumetric rendering to target views. First, the corresponding pixels $[u_t, v_t]$ in the target view can be found by

$$
\begin{bmatrix} u_s \\ v_s \\ 1 \end{bmatrix} \sim K_s (R - t n^T d_i)(K_t)^{-1} \begin{bmatrix} u_t \\ v_t \\ 1 \end{bmatrix} \tag{15}
$$

Here, $[u_s, v_s]$ is the corresponding pixel locations in the source view, K_s and K_t are camera intrinsics of source and target views, R and t are rotation and translation from the target to source view, and n is the norm vector of the i_{th} plane. As the planes are parallel, the RGB c_i' and density σ_i' of shading points (blue points) on target rays (blue ray) are equal to those from source rays at the same disparity, as shown in Eq. 16,

$$
\begin{aligned}
c_i'(u_t, v_t) &= c_i(u_s, v_s) \\
\sigma_i'(u_t, v_t) &= \sigma_i(u_s, v_s)
\end{aligned}
\tag{16}
$$

Once we have RGB and density for target views, we can perform volumetric rendering on target views as:

$$
\begin{aligned}
\hat{I}_t &= \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i'\delta_i))c_i' \\
\hat{Z}_t &= \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i'\delta_i))z_i'
\end{aligned}
\tag{17}
$$

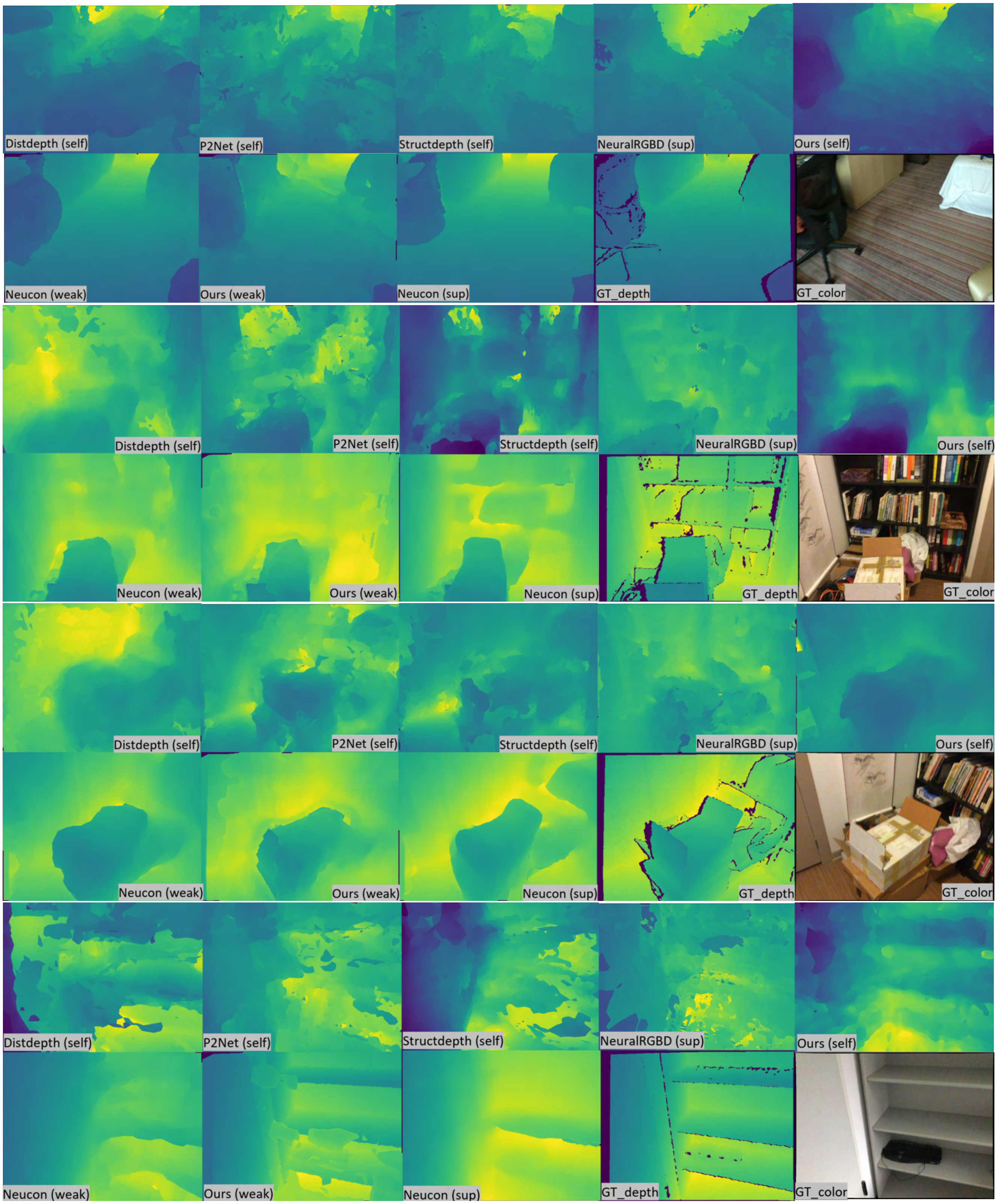
We use standard NeRF RGB loss, where $\hat{I}_s$ and $\hat{I}_t$ are self-supervised with their corresponding input images with a L1 loss. Since we directly use the reverse of disparity for depth, the depth value is scale-ambiguous. As mentioned in the paper, since there is no depth ground truth for pure self-supervision, we use SDF-depth as pseudo-depth to first recover the real scale of $\hat{Z}_s$ and $\hat{Z}_t$, then we impose a consistency loss between $\hat{Z}$ and SDF-depth to boost SDF decoder.

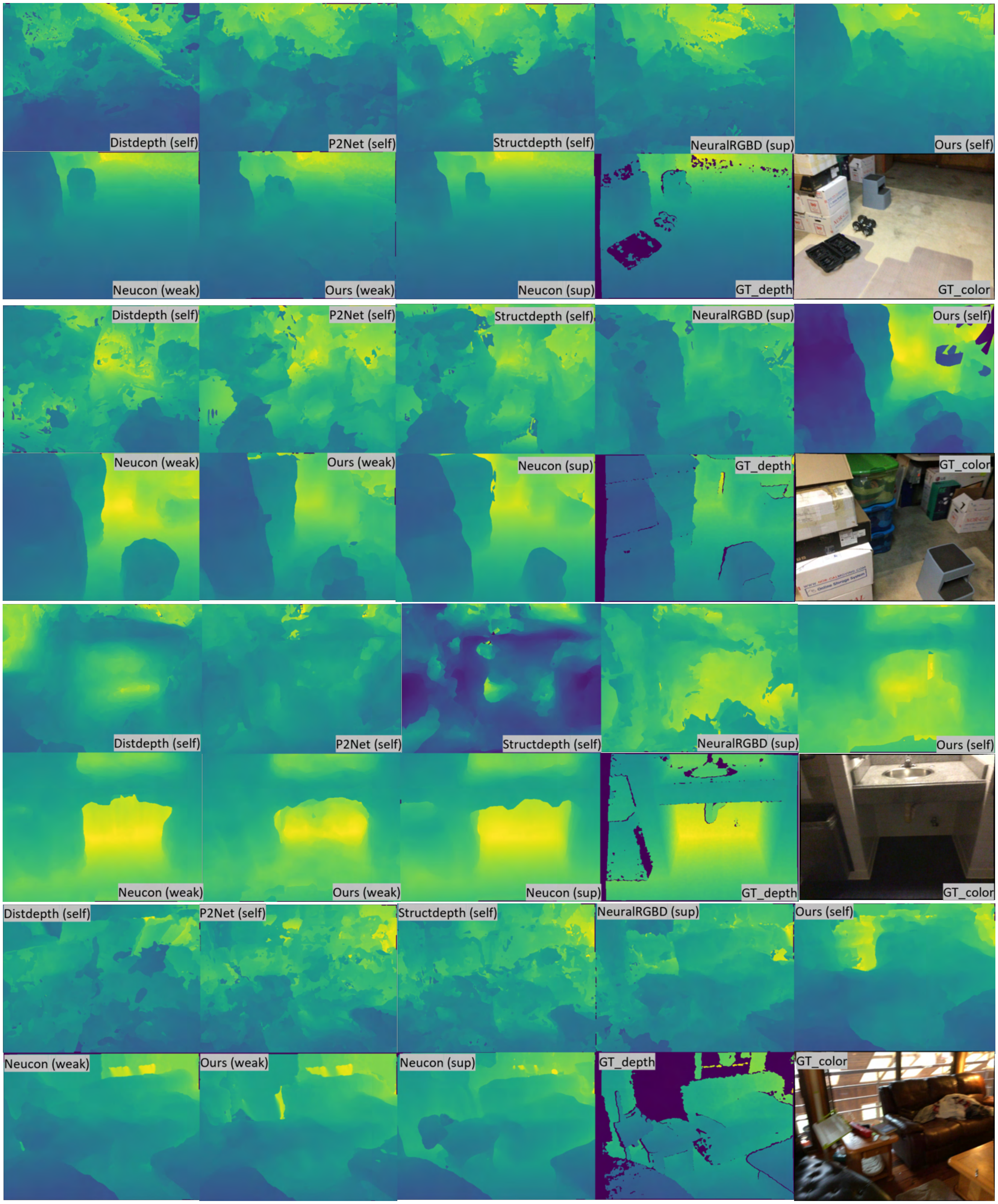
9. Visual Results

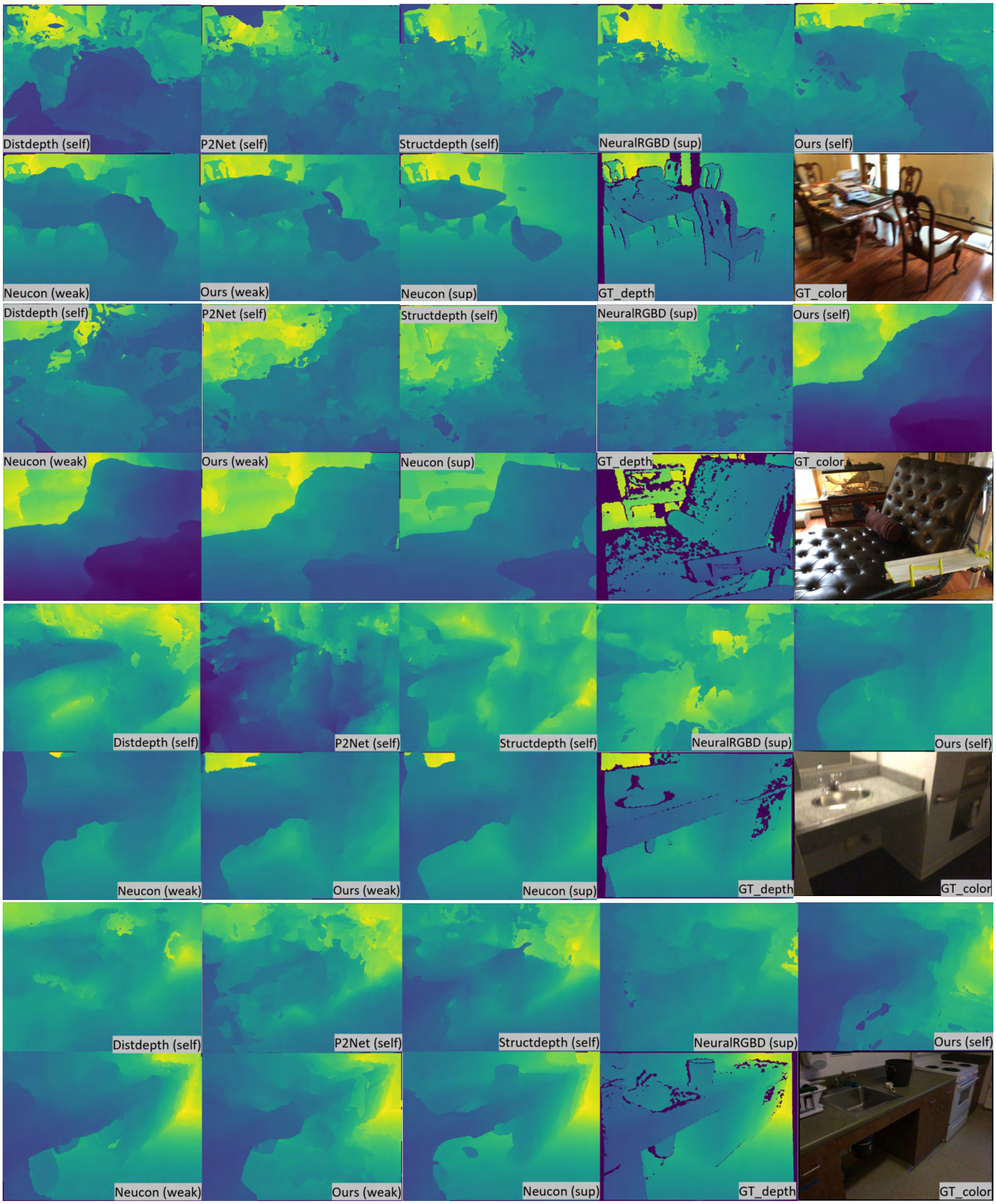
We show more visual results of 2D rendered depth and 3D mesh in Figure 6. **We also attach a PowerPoint file with visual results, where reviewers can rotate and zoom the 3D mesh to see the details**,

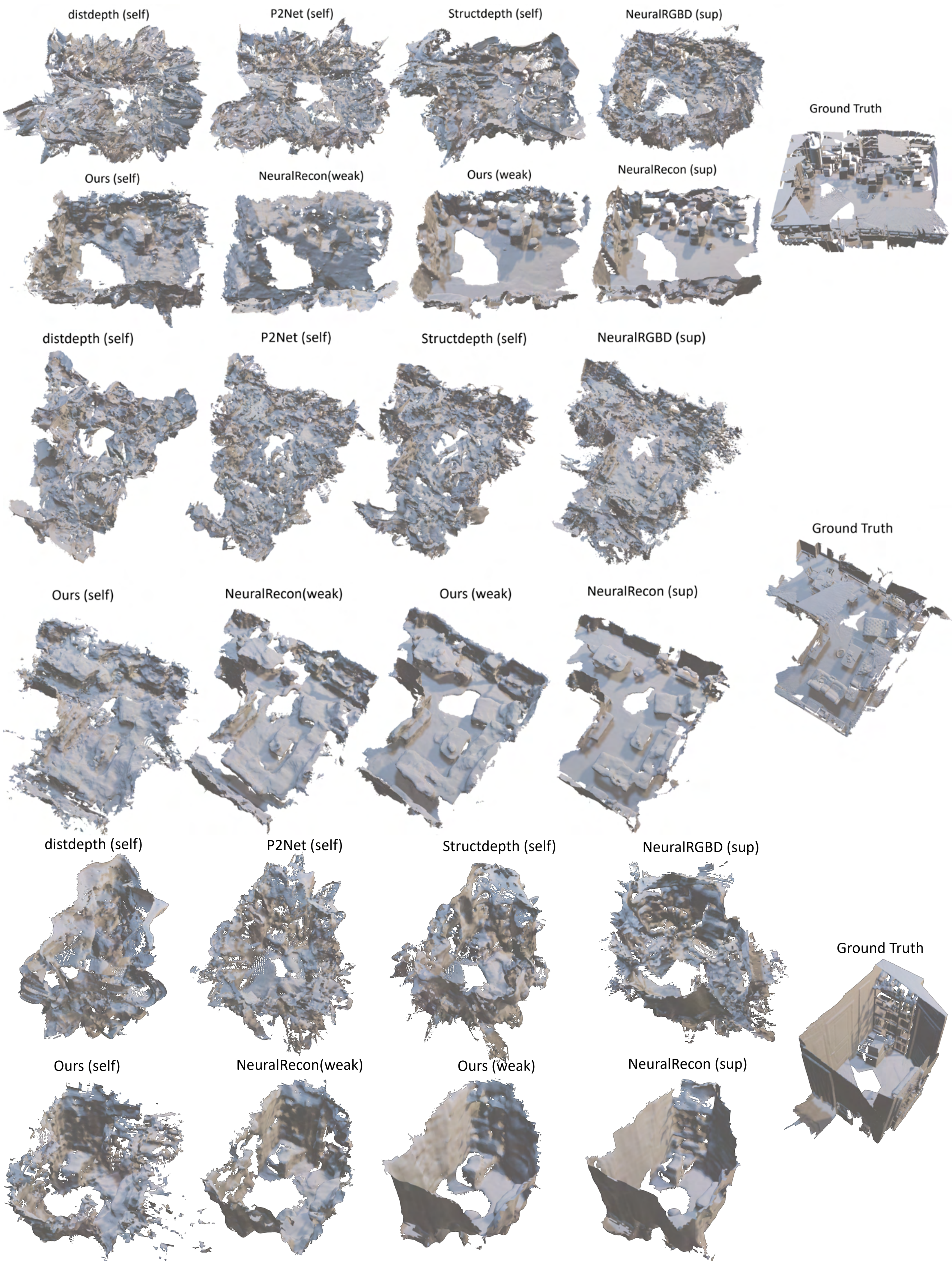
10. Limitation

Although our work combines the advantages of "self-supervised" "generalizable" and "3D explicit mesh" altogether, there are still limitations. So far our MonoSelfRecon can be only used for indoor environments, because we pre-define the 3D scene fragment with a fixed voxel number. Unlike indoor 2D images where depth vary within few meters, the depth can vary significantly just within a single 2D image in outdoor. It is applicable to keep the voxel number while increasing the voxel size, but it will lead to very poor resolution within voxels, which misses most of the details. Moreover, since we regress SDF corresponding to the discrete $N \times N \times N$ voxels of scene fragment, we cannot directly estimate SDF of a continuous 3D space, unless by interpolation. By contrast, SDF-NeRF based methods estimate SDF values in continuous 3D space but it is not generalizable to another scene. Our future works will explore to make SDF-NeRF generalizable, so that the 3DR can be "self-supervised", "generalizable", "explicit", "indoor/outdoor", and "continous in 3D space".


Distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ours (self)
Neucon (weak)
Ours (weak)
Neucon (sup)
GT_depth
GT_color
Distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ours (self)
Neucon (weak)
Ours (weak)
Neucon (sup)
GT_depth
GT_color
Distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ours (self)
Neucon (weak)
Ours (weak)
Neucon (sup)
GT_depth
GT_color
Distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ours (self)
Neucon (weak)
Ours (weak)
Neucon (sup)
GT_depth
GT_color


Distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ours (self)
Neucon (weak)
Ours (weak)
Neucon (sup)
GT_depth
GT_color


Distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ours (self)
Neucon (weak)
Ours (weak)
Neucon (sup)
GT_depth
GT_color
Distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ours (self)
Neucon (weak)
Ours (weak)
Neucon (sup)
GT_depth
GT_color
Distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ours (self)
Neucon (weak)
Ours (weak)
Neucon (sup)
GT_depth
GT_color
Distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ours (self)
Neucon (weak)
Ours (weak)
Neucon (sup)
GT_depth
GT_color


distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ground Truth
Ours (self)
NeuralRecon(weak)
Ours (weak)
NeuralRecon (sup)
distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ground Truth
Ours (self)
NeuralRecon(weak)
Ours (weak)
NeuralRecon (sup)
distdepth (self)
P2Net (self)
Structdepth (self)
NeuralRGBD (sup)
Ground Truth
Ours (self)
NeuralRecon(weak)
Ours (weak)
NeuralRecon (sup)

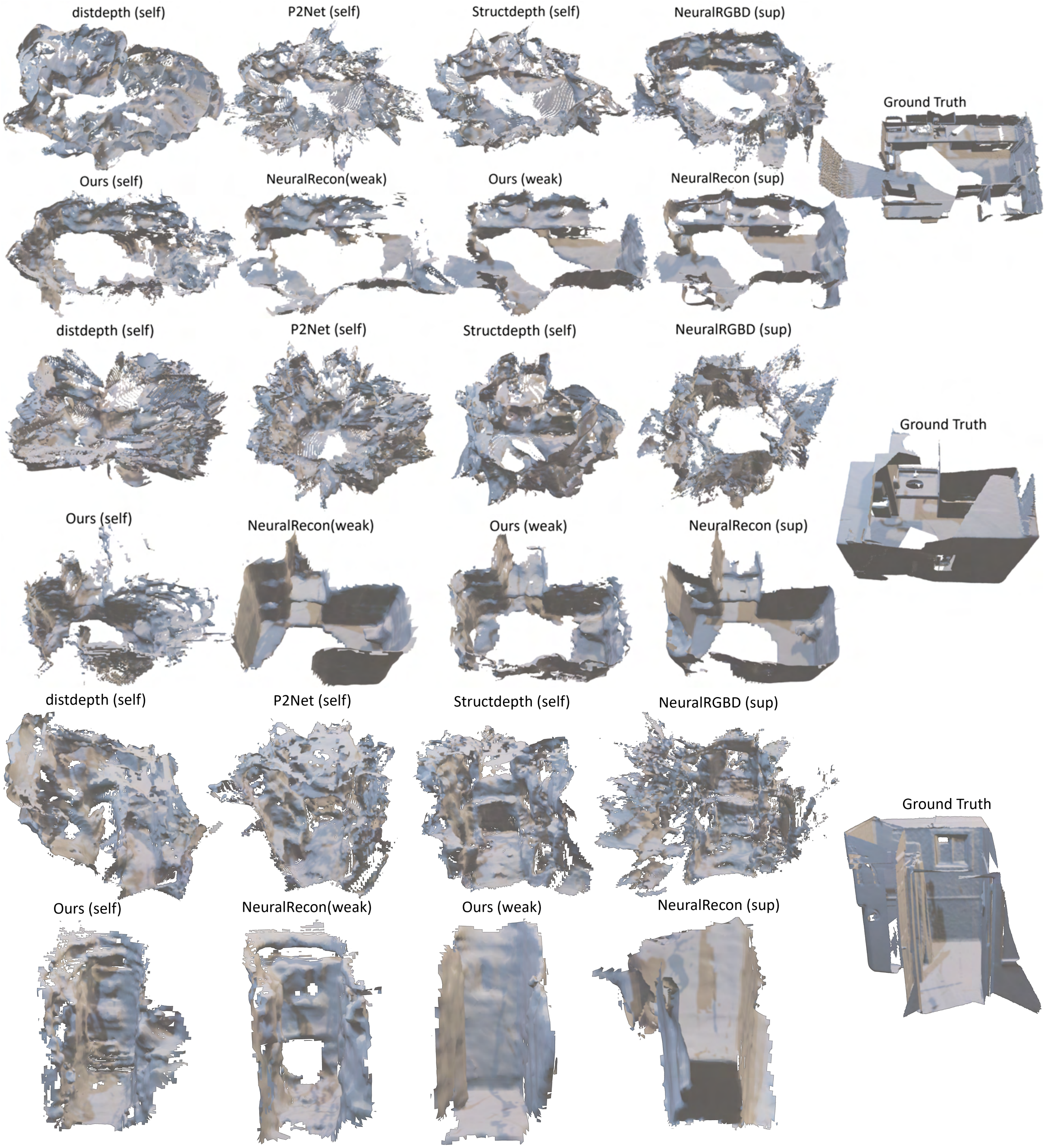


Figure 6. **Visual Results on ScanNet.**