JacobiNeRF: NeRF Shaping with Mutual Information Gradients (Appendix)

1. Mutual information approximated by Jacobian under scene perturbations

As defined in Sec. 3, p_i and p_j are different pixels with values $I(\mathrm{p}_i)$ and $I(\mathrm{p}_j)$, respectively, and they may or may not come from the same image (view). Generally speaking, p_i and p_j can also be the radiance value of 3D points. Without loss of generality, we derive with (gray-scale) 2D pixels in the following, and they can be expressed as:

$$I(\mathrm{p}_i) = \Phi(\mathbf{o}_i, \mathbf{v}_i; \theta)$$
$$I(\mathrm{p}_j) = \Phi(\mathbf{o}_j, \mathbf{v}_j; \theta)$$

Further, we denote θ^D as the set of parameters that will be perturbed by a random noise $\mathbf{n} \in \mathbb{R}^D$ sampled from a uniform distribution on the sphere $\mathbb{S}^{D-1}$. Please see Fig. 2 in the main text for different selection patterns of θ^D. The random variables representing the perturbed pixel values are then:

$$\hat{I}(\mathrm{p}_i) = \Phi(\mathbf{o}_i, \mathbf{v}_i; \theta^D + \mathbf{n})$$
$$\hat{I}(\mathrm{p}_j) = \Phi(\mathbf{o}_j, \mathbf{v}_j; \theta^D + \mathbf{n})$$

where we omit the parameters remain unchanged for clarity.

Now we characterize the mutual information between $\hat{I}(\mathrm{p}_i)$ and $\hat{I}(\mathrm{p}_j)$ under the perturbation-induced joint probability distribution $\mathbb{P}(\hat{I}(\mathrm{p}_i), \hat{I}(\mathrm{p}_j))$. However, as mentioned, calculating the joint distribution under a push-forward of the MLP is complicated due to non-linearities. Thus, we proceed by constraining the magnitude of the perturbations, namely, multiplying the random noise $\mathbf{n}$ by $\sigma \ll 1.0$. Again, this constraint improves compliance with the fact that a small perturbation in the physical scene is enough to reveal mutual information between scene entities. Moreover, it guarantees that the perturbed representation still represents a legitimate scene. With this constraint, we can explicitly write the random variables under consideration as:

$$\hat{I}(\mathrm{p}_i) = I(\mathrm{p}_i) + \sigma\mathbf{n} \cdot \frac{\partial \Phi(\mathbf{o}_i, \mathbf{v}_i; \theta)}{\partial \theta^D}$$
$$\hat{I}(\mathrm{p}_j) = I(\mathrm{p}_j) + \sigma\mathbf{n} \cdot \frac{\partial \Phi(\mathbf{o}_j, \mathbf{v}_j; \theta)}{\partial \theta^D}$$

following the Taylor expansion. We denote the Jacobians as $\partial\Phi_i/\partial\theta^D$ or $\partial\Phi_i$ for easy notation. We can show that the mutual information is:

$$\begin{aligned}\mathbb{I}(\hat{I}(\mathrm{p}_i), \hat{I}(\mathrm{p}_j)) &= \mathbb{H}(\hat{I}(\mathrm{p}_j)) - \mathbb{H}(\hat{I}(\mathrm{p}_j) \mid \hat{I}(\mathrm{p}_i)) \\ &= \mathbb{H}(\sigma\mathbf{n} \cdot \partial\Phi_j) - \mathbb{H}(\sigma\mathbf{n} \cdot \partial\Phi_j \mid \sigma\mathbf{n} \cdot \partial\Phi_i)\end{aligned}$$

leveraging the fact that entropy is translation-invariant.

From the above equation, we can see that the mutual information between the two perturbed pixels can be approximated by the mutual information between two random projections, e.g., $\sigma\mathbf{n} \cdot \partial\Phi_j$ and $\sigma\mathbf{n} \cdot \partial\Phi_i$. To further ease annotation, we let $\mathbf{A} = \partial\Phi_i$ and $\mathbf{B} = \partial\Phi_j$. And,

$$\mathbb{I}(\hat{I}(\mathrm{p}_i), \hat{I}(\mathrm{p}_j)) = \mathbb{H}(\sigma\mathbf{n} \cdot \mathbf{B}) - \mathbb{H}(\sigma\mathbf{n} \cdot \mathbf{B} \mid \sigma\mathbf{n} \cdot \mathbf{A})$$

To proceed, we write the Jacobians and the random (small) perturbation in the spherical form:

$$\mathbf{A} = \sigma_A \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \mathbf{B} = \sigma_B \begin{pmatrix} \cos\alpha_1^B \\ \sin\alpha_1^B \cos\alpha_2^B \\ \sin\alpha_1^B \sin\alpha_2^B \cos\alpha_3^B \\ \vdots \\ \sin\alpha_1^B \ldots \sin\alpha_{D-2}^B \cos\alpha_{D-1}^B \\ \sin\alpha_1^B \ldots \sin\alpha_{D-2}^B \sin\alpha_{D-1}^B \end{pmatrix},$$

$$\sigma\mathbf{n} = \sigma \begin{pmatrix} \cos\alpha_1 \\ \sin\alpha_1 \cos\alpha_2 \\ \sin\alpha_1 \sin\alpha_2 \cos\alpha_3 \\ \vdots \\ \sin\alpha_1 \ldots \sin\alpha_{D-2} \cos\alpha_{D-1} \\ \sin\alpha_1 \ldots \sin\alpha_{D-2} \sin\alpha_{D-1} \end{pmatrix} \tag{1}$$

Here we set (by rotation) the direction of $\mathbf{A}$ to be the unit vector in the first dimension. Thus, the cosine similarity between $\mathbf{B}$ and $\mathbf{A}$ is now the cosine value of the first angle (α_1^B) of $\mathbf{B}$ in the spherical form (similarly for the noise vector $\mathbf{n}$). Note that the above parameterization does not change the entropy or conditional entropy since $\sigma\mathbf{n}$ is uniformly distributed in each direction. Also, note that the scaling of a distribution shifts its entropy by a logarithm of the scaling factor. Thus, the first term is a constant given symmetry (randomness is in $\{\alpha_k\}_{k=1\ldots D-1}$), i.e.,

$$\mathbb{H}(\sigma\mathbf{n} \cdot \mathbf{B}) = \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-1}) + \log(\sigma_B \sigma) \tag{2}$$

Next, we calculate $\mathbb{H}(\sigma\mathbf{n} \cdot \mathbf{B} | \sigma\mathbf{n} \cdot \mathbf{A})$. First, let's check $\mathbb{H}(\sigma\mathbf{n} \cdot \mathbf{B} | \sigma\mathbf{n} \cdot \mathbf{A} = y)$, i.e., the entropy of $\sigma\mathbf{n} \cdot \mathbf{B}$ when the project of $\sigma\mathbf{n}$ on $\mathbf{A}$ is y. We have $y = \sigma_A \sigma \cos\alpha_1$, and

$$\begin{aligned}
\sigma\mathbf{n}\cdot\mathbf{B} &= \sigma_B \cos\alpha_1^B \sigma \cos\alpha_1 + \sigma_B \sin\alpha_1^B \sigma \sin\alpha_1 \cdot \\
&\begin{pmatrix} \cos\alpha_2^B \\ \vdots \\ \sin\alpha_2^B \ldots \sin\alpha_{D-1}^B \end{pmatrix}^T \begin{pmatrix} \cos\alpha_2 \\ \vdots \\ \sin\alpha_2 \ldots \sin\alpha_{D-1} \end{pmatrix} \\
&= y\cos\alpha_1^B \frac{\sigma_B}{\sigma_A} + \sigma_B \sin\alpha_1^B \sigma \sqrt{1-(\frac{y}{\sigma_A\sigma})^2} \cdot \\
&\begin{pmatrix} \cos\alpha_2^B \\ \vdots \\ \sin\alpha_2^B \ldots \sin\alpha_{D-1}^B \end{pmatrix}^T \begin{pmatrix} \cos\alpha_2 \\ \vdots \\ \sin\alpha_2 \ldots \sin\alpha_{D-1} \end{pmatrix} \qquad (3)
\end{aligned}$$

Thus, we have:

$$\begin{aligned}
\mathbb{H}(\sigma\mathbf{n}\cdot\mathbf{B}|\sigma\mathbf{n}\cdot\mathbf{A}=y) &= \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-2}) + \\
&\log(\sigma_B \sin\alpha_1^B \sigma \sqrt{1-(\frac{y}{\sigma_A\sigma})^2}) \qquad (4)
\end{aligned}$$

Suppose that the probability density function of y is f_Y, then we have:

$$\begin{aligned}
\mathbb{H}(\sigma\mathbf{n}\cdot\mathbf{B}|\sigma\mathbf{n}\cdot\mathbf{A}) &= \int f_Y(y)\mathbb{H}(\sigma\mathbf{n}\cdot\mathbf{B}|\sigma\mathbf{n}\cdot\mathbf{A}=y)\mathrm{d}y \\
&= \int f_Y(y)\mathbb{H}^{\text{proj}}(\mathbb{S}^{D-2})\mathrm{d}y + \\
&\int f_Y(y)\log(\sigma_B \sin\alpha_1^B \sigma \sqrt{1-(\frac{y}{\sigma_A\sigma})^2})\mathrm{d}y \\
&= \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-2}) + \log(\sigma_B \sin\alpha_1^B \sigma) + \\
&\int_{-\sigma_A\sigma}^{\sigma_A\sigma} f_Y(y)\sqrt{1-(\frac{y}{\sigma_A\sigma})^2}\mathrm{d}y \\
&= \log(\sigma_B \sin\alpha_1^B \sigma) + \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-2}) + \int_0^\pi f_{\alpha_1}(\alpha)\sin(\alpha)\mathrm{d}\alpha \qquad (5)
\end{aligned}$$

Combining Eq. (2) and Eq. (5), and denote the last term in Eq. (5) as $h(f_{\alpha 1})$ (*constant*) with f_{α_1} the probability density function of α_1, then we have:

$$\begin{aligned}
\mathbb{I}(\hat{I}(\mathrm{p}_i), \hat{I}(\mathrm{p}_j)) &= \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-1}) + \log(\sigma_B\sigma) - \\
&\log(\sigma_B \sin\alpha_1^B \sigma) - \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-2}) - h(f_{\alpha 1}) \\
&= \log(\frac{1}{\sin\alpha_1^B}) + \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-1}) - \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-2}) - h(f_{\alpha 1}) \\
&= \log(\frac{1}{\sqrt{1-\cos^2\alpha_1^B}}) + \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-1}) - \mathbb{H}^{\text{proj}}(\mathbb{S}^{D-2}) - h(f_{\alpha 1}) \\
&(\propto \|\cos\alpha_1^B\|) \qquad (6)
\end{aligned}$$

According to Eq. (6), we can see that the mutual information between the two pixels $\mathrm{p}_i, \mathrm{p}_j$ under the perturbation-induced joint distribution is positively correlated to the absolute value of the cosine similarity of their gradients with respect to the perturbed network weights.

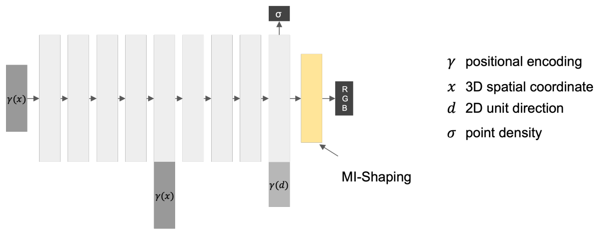

Figure 1. The NeRF architecture and the network parameters have been shaped. γ denotes positional encoding, x denotes 3D spatial coordinate, d represents 2D unit direction, and σ represents point density. The proposed mutual information shaping (MI-shaping) in our current implementation is applied on the 3×128-dimensional parameters of NeRF's RGB linear layer, shown as the orange layer in the figure.

2. Implementation details

2.1. Training details

The MI-shaping process follows Sec. 3.3 in the main text, where we first train the NeRF with only images, and then shape it via the proposed contrastive loss on the gradients. The gradients we use to shape NeRF are computed with the gray values of pixels (mean value of the 3 RGB channels), with respect to the 3×128-dimensional parameters of NeRF's RGB linear layer, as shown in Fig. 1.

For each epoch, we randomly sample a batch of 64 rays across all the training views. For each sampled ray, we select from the other rays whose corresponding DINO feature similarity with the sampled ray is higher than a threshold as the positive sample, and the others with lower similarity as negative samples (the total number of samples in the contrastive loss of a certain ray is also 64). The threshold for each scene is adaptively adjusted during the training process. We first set an interval $[0.5, 0.8]$ for the threshold, then for each step, we subtract 0.001 from the threshold if the ratio of positive samples is lower than 5%, and add 0.001 to the threshold if the ratio is larger than 15%, unless the threshold exceeds the interval. The loss is shown in Eq. (8), where $\lambda = 0.01, \gamma = 0.01$, optimized by the Adam [1] optimizer, with an initial learning rate of 5×10^{-4} for 10000 epochs.

The training runs for 24 hrs with color reconstruction and then 8 hrs with the shaping loss activated (mem: 1350 MB). Semantic-NeRF [4] and DINO-NeRF [2] take 28 hrs.

2.2. Label propagation with JacobiNeRF in 3D

Here we elaborate on the implementation details of the label propagation method denoted as J-NeRF 3D in the main text. The only difference between J-NeRF 3D and J-NeRF 2D is that we collect the perturbation responses of sampled points in 3D instead of 2D pixel difference, and then volume render them into segmentation logits for pixels.

To render a 2D segmentation label, we first sample points along the rays following the hierarchical sampling strategy of NeRF [3]. Then we render the points' unperturbed radiance values and K (K is the number of classes) times the perturbed values, thus obtaining K-dimensional differences for each point. We integrate the differences of the sampled points along each ray following the volume render process in Eq. (1) to get a K dimensional segmentation logits in 2D pixel space.

2.3. Label propagation under dense setting

2.3.1 Adaptive gradient sampling

For efficiency, we have to choose representative gradients for each class to perform perturbation rather than perturbing along all the given labels' gradients. To avoid loss of label information, we employ an adaptive gradient sampling strategy to make the gradients as sufficient as possible to reconstruct the given dense label.

For the first round, we randomly select 20 combinations of labeled pixels, each combination containing K pixels (K is the number of classes). We perturb along these gradients and get the response of the source view (whose dense label is known), thereby getting label prediction of the source view from the Argmax of the response. We evaluate the quality of the predicted label by calculating gain (mIoU) with the given dense label and choose the combination of gradients that yields the largest gain. For the next iteration, we sample another 20 combinations of gradients, and choose the one with the largest gain. We discard these gradients if the gain from these 20 combinations is not larger than zero. We repeat the above process until we get 5 qualified selections, i.e., each selection gives us K labels, in total 5$\times$K labels. Fig. 2 shows how we adaptively sample representative gradients. As the iteration of adaptive sampling goes on, the selected gradients can better reconstruct the given dense label, indicating that the gradients encode more and more information about the dense label.

2.3.2 Aggregation MLP

We leverage a lightweight aggregation/denoising MLP to further denoise the perturbed response with information from the dense label. The aggregation MLP maps from each pixel's K-dimensional (K is the number of classes) perturbation responses to K-dimensional segmentation logits. After we adaptively select the gradients and get the responses on the source view by perturbing along the selected gradients, we train the aggregation MLP from scratch under aggregation loss L_{agg}:

$$L_{agg} = -\sum_{r\in\mathcal{R}}\sum_{k=1}^{K} p^k(r)\log p^k_{agg}(r), \tag{7}$$

where $\mathcal{R}$ are the rays within the source view, p^k is the probability at class k of the ground-truth label, and p^k_{agg} is the probability at class k predicted by the aggregation MLP from the perturbed response. L_{agg} is a multi-class cross-entropy loss to encourage the denoised predicted label to be consistent with the given ground-truth dense labels. The MLP has 3 linear layers and ReLU activation, with $K \times 256 + 256 \times 128 + 128 \times K$ parameters. The MLP is optimized with the Adam [1] optimizer with an initial learning rate of 1×10^{-3}. We train it for 200000 iterations, approximately 30 minutes. On the right of Tab. 2, we show the quantitative effect of the lightweight aggregation MLP (*Please note that J-NeRF+ represents the propagation with the denoising MLP.*). Fig. 2 shows the denoising effect qualitatively.

3. Additional results

3.1. Novel view synthesis quality

We also report the novel view synthesis quality of the plain NeRF and the proposed JacobiNeRF in Tab. 1. As confirmed, the MI-shaping process does not degrade the performance of view synthesis, i.e., the image quality is similar to the one without shaping.

Method	NeRF	J-NeRF
PSNR $\uparrow$	18.88	18.79

Table 1. The performance of novel view synthesis on the Replica dataset measured by the peak signal-to-noise ratio (PSNR).

3.2. Propagation capability of plain NeRF

To further validate the effectiveness of our NeRF shaping method, we replace JacobiNeRF with plain NeRF (w/o MI-shaping) in our propagation pipeline and propagate in 2D by perturbing along the gradients of labeled pixels. The semantic segmentation propagation results, in the sparse setting, averaged over the 7 scenes from Replica are shown in Tab. 2 (left). As observed, J-NeRF 2D significantly outperforms plain NeRF, proving that our method shapes NeRF effectively to make it more suitable for information propagation.

Method	NeRF	J-NeRF
mIoU↑	0.067	**0.263**
Avg Acc↑	0.182	**0.489**
Total Acc↑	0.199	**0.483**

Method	J-NeRF	J-NeRF+
mIoU↑	0.411	**0.524**
Avg Acc↑	**0.724**	0.689
Total Acc↑	0.633	**0.864**

Table 2. Effectiveness of the proposed shaping on a plain NeRF (left), and of the lightweight perturbation response decoder (right).

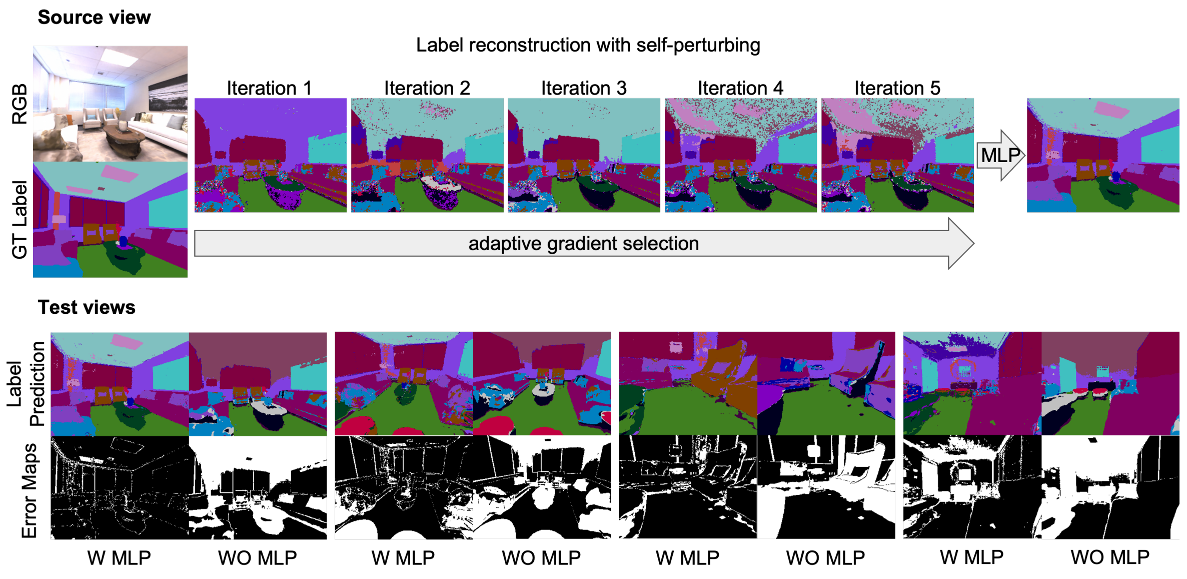

Figure 2. The effect of adaptive gradient sampling and aggregation MLP. Top: as more and more qualified gradients are selected, the reconstruction quality of the dense input label also gets better. Bottom: the denoising MLP can make better decode the perturbation responses.

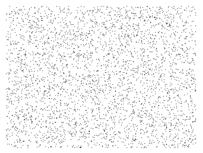

Figure 3. Distribution of selected regularization points.

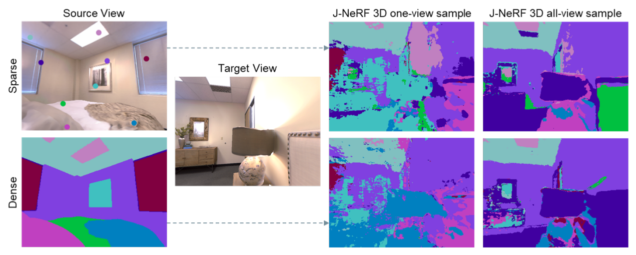

Figure 4. Qualitative comparison: one-view v. all-view regularization.

3.3. Learnable "argmax" for dense label propagation

We study the effectiveness of the denoising MLP used in the dense setting by comparing to those labels from the Argmax of the perturbation response. Results averaged over 7 scenes are shown in Tab. 2 (right), indicating that the lightweight MLP improves the utilization of the dense labels.

3.4. Distribution of selected regularization points

Points sampled during the contrastive training (MI-shaping) process are uniformly random as in Fig. 3.

3.5. Fewer regularization samples.

In Fig. 4, we observe that when the selected samples are constrained to one view, the clustering effect reduces compared to all-view regularization.

4. Additional Visualization

Please see Fig. 5, 6, 7 for more qualitative label propagation results.

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2, 3

[2] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. 2

[3] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 3

[4] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15838–15847, 2021. 2

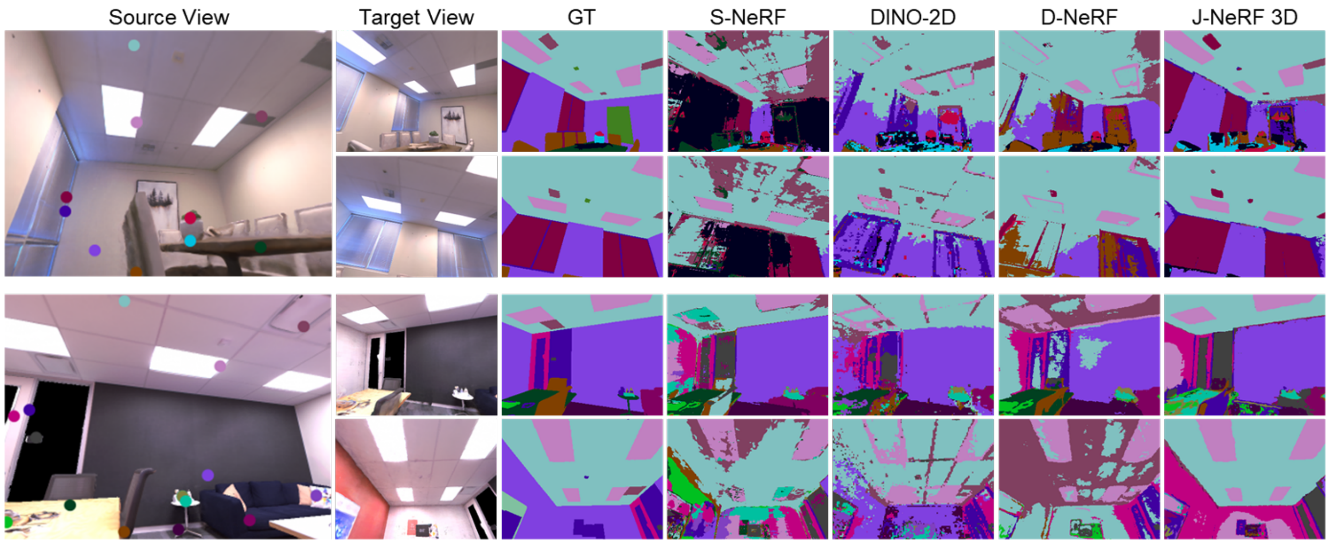

Figure 5. Qualitative results of semantic segmentation propagation under the sparse setting. Examples are from the Replica dataset.

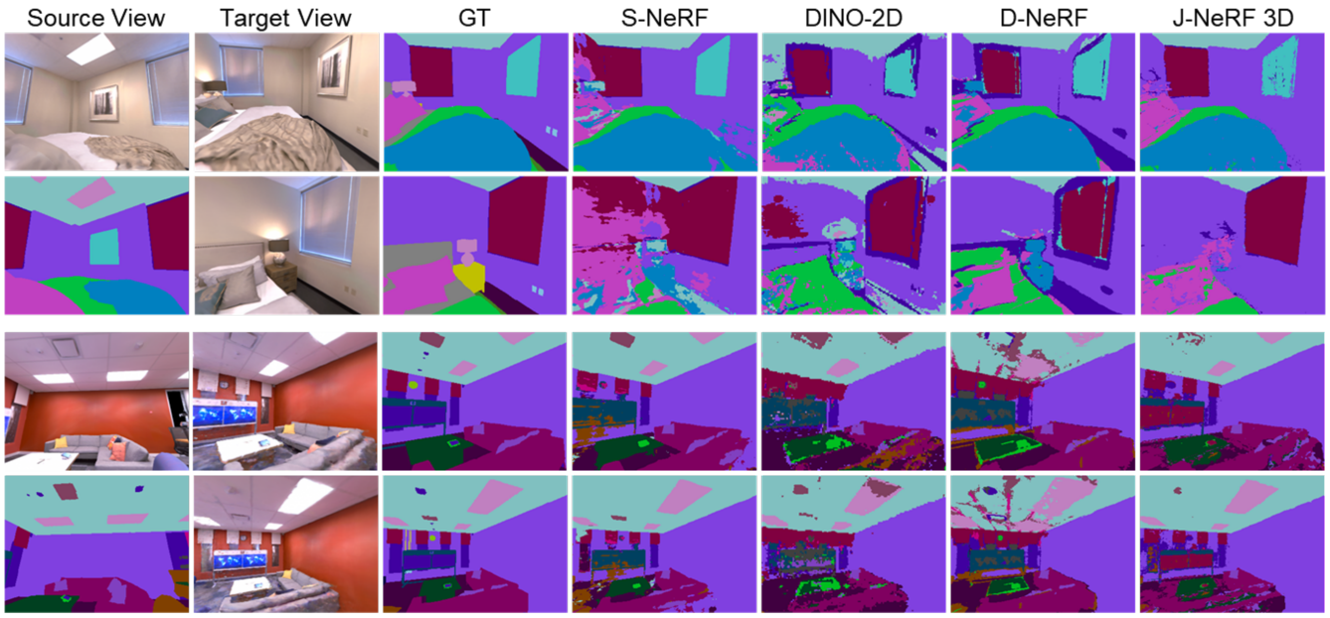

Figure 6. Qualitative results of semantic segmentation propagation under the dense setting. Examples are from the Replica dataset.

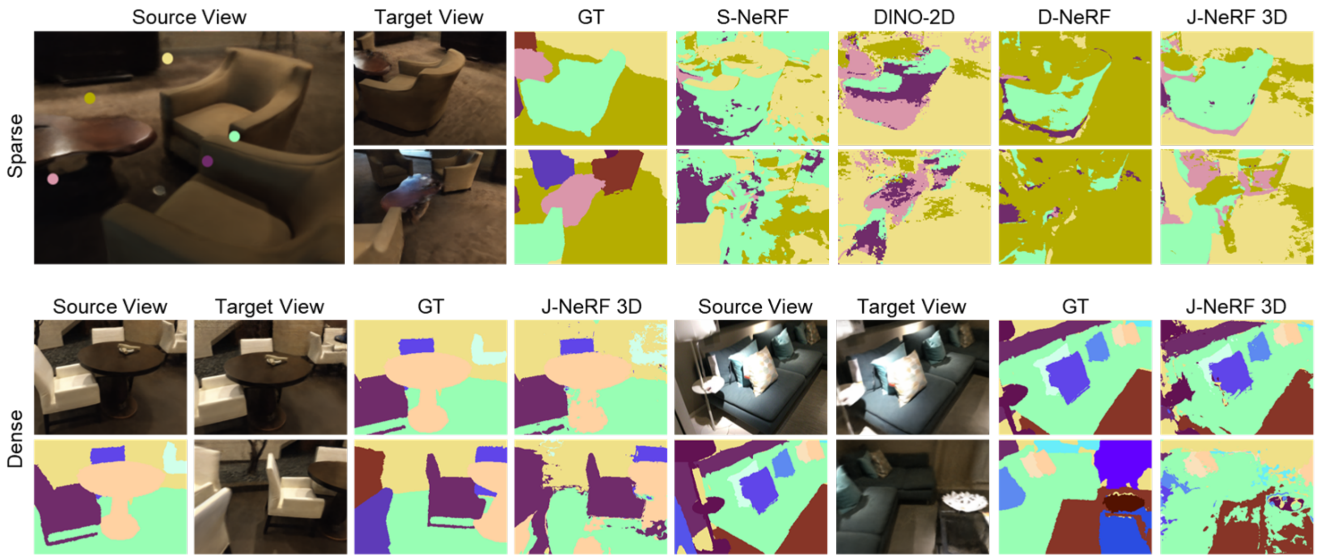

Figure 7. Qualitative results of instance segmentation propagation under both sparse and dense settings. Examples are from the ScanNet dataset.