# A SUPPLEMENTARY VIDEO

We recommend the reviewers to refer to the additional results presented in the supplementary video.

# B DETAILS OF SEGMENTATION AND ALIGNMENT

We present the specific calculations for the segmentation and alignment of the background scene and foreground objects, as introduced in Sec. 3.2 of the main paper. The overall process can be summarized as follows:

**Background scene alignment.** Utilize highly overlapping static background information and corresponding point clouds to perform pose registration between different scenes. (B.1)

**Background scene/foreground objects segmentation.** Use the proposed fusion method to obtain a clean background, and compare its differences with the point clouds from each scene to obtain the union of all point clouds corresponding to foreground objects. Then employ a clustering algorithm to obtain point clouds corresponding to each individual foreground object. (B.2)

**Foreground objects matching and alignment** Perform matching and registration of the point clouds for each foreground object, thereby obtaining the relative poses between the foreground objects. (B.3)

The illustration is shown in Fig. 6. Note that a part of our method leverages existing point cloud calculations, which can be implemented with minimal code using Open3D Zhou et al. (2018).

## B.1 BACKGROUND SCENE ALIGNMENT

As a preparatory step for radiance field reconstruction, we independently run COLMAP Schönberger & Frahm (2016); Schönberger et al. (2016) for $N$ scenes to obtain camera registrations. In such cases, the camera poses and optimized radiance fields are represented in different coordinate systems belonging to their respective scenes. To enable comparison of radiance fields across scenes, we introduce a method for aligning the scenes to a reference scene. Without loss of generality, we choose the first scene as the reference scene. Due to the lack of scale information in COLMAP, the alignment between scenes is a $\mathrm{Sim}(3)$ registration[1] problem.

**Naïve solution.** Given the scene point cloud $\{\mathcal{P}_i\}$ computed from each radiance fields, the scene alignment task can be formulated as finding an appropriate transformation $Q_i \in \mathrm{Sim}(3)$ for the point cloud $\mathcal{P}_i$ that minimizes the error $||\mathcal{P}_{\text{ref}} - Q_i\mathcal{P}_i||$ where $\mathcal{P}_{\text{ref}}$ is the point cloud of the reference scene. Note that although the foreground object placements varies across scenes, we assume that the static background provides sufficient information for obtaining reasonable scene registration results. Then, $Q_i$ can be determined using traditional point cloud registration algorithms, *e.g.*, FPFH feature matching Rusu et al. (2009) with RANSAC Fischler & Bolles (1981) solver, followed by Iterative Closest Point (ICP) Arun et al. (1987) refinement. Unless otherwise specified, the term "point cloud registration" in the following text uses this two-step algorithm.

**More robust solution.** The above naïve solution can yield reasonable results in some scenes. However, for some challenging scenes, we have observed that relying solely on point cloud information can lead to failure, since the geometric information of the background may not be unique enough, leading to unstable registration results. Hence, we propose to leverage the available multi-view image information to facilitate the registration process. Specifically, for scene $i$, we consider the multi-view images $\{\mathcal{I}_l\}_i$ and the 3D position $\{\mathbf{q}_l\}_i$ of its registered camera, where $\mathbf{q}_l \in \mathbb{R}^3$ is expressed in the coordinate system of scene $i$. Using COLMAP registration of the reference scene, we register images $\{\mathcal{I}_l\}_i$ to the reference scene to obtain the camera position $\{\mathbf{q}_l\}_i'$ expressed in the reference coordinate system. Although the object placements in scene $i$ differ from those in the reference scene, the RGB information in the background provides sufficient information to complete the registration. Here, since $\{\mathbf{q}_l\}_i$ and $\{\mathbf{q}_l\}_i'$ represent the same camera position expressed in the coordinate systems of scene $i$ and the reference scene, respectively, given the relative pose between the scenes $Q_i$, it should satisfy that $\{\mathbf{q}_l\}_i' = Q_i\{\mathbf{q}_l\}_i$ for all $l$. Therefore, we solve for the

[1] $\mathrm{Sim}(3)$ transformation represents $\mathrm{SE}(3)$ transformation with an additional scale factor.

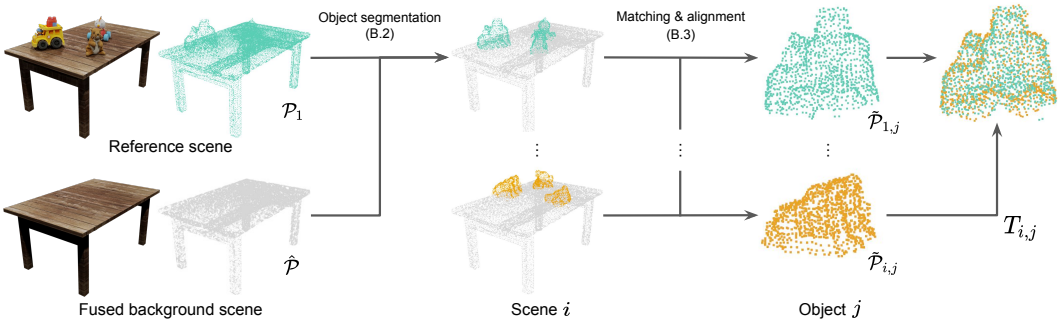


Figure 6: **Pipeline of foreground objects segmentation and alignment**. We use the point clouds obtained from the original scenes and the reconstructed clean background to create a set of point clouds representing all foreground objects. Then, we utilize point cloud matching and registration algorithms to determine the correspondences between objects and their relative poses.

transformation $Q_i \in \mathrm{Sim}(3)$ that minimizes the registration error between camera positions, given by

$$\tilde{Q}_i = \arg\min_{Q_i} \sum_l ||\{\mathbf{q}_l\}_i' - Q_i\{\mathbf{q}_l\}_i||. \tag{7}$$

Similarly, we solve for $Q_i$ using RANSAC followed by ICP refinement.

### B.2 Background scene/foreground objects segmentation

Using the alignment obtained from B.1, we use the fusion method described in Sec. 3.4 to obtain a clean background scene. Then, we extract the point cloud corresponding to the background scene, denoted as $\hat{\mathcal{P}}$, and we denote the surface point clouds of each scene $\{\mathcal{S}_i\}$ as $\{\mathcal{P}_i\}$. Here we represent all of the point clouds in the reference coordinate system.

For each scene, we compare it with the background scene and calculate the distance between the point clouds $\mathcal{P}_i$ and $\hat{\mathcal{P}}$. As shown in Fig. 6, the difference between these point clouds corresponds to the foreground objects. By setting an appropriate threshold $\delta > 0$, we can obtain the foreground point clouds $\tilde{\mathcal{P}}_i$ as follows:

$$\tilde{\mathcal{P}}_i = \{\mathbf{p} | \mathrm{dist}(\mathbf{p}, \hat{\mathcal{P}}) > \delta,\ \mathbf{p} \in \mathcal{P}_i\} \tag{8}$$

where $\mathrm{dist}(\mathbf{p}, \hat{\mathcal{P}})$ denotes the Euclidean distance between $\mathbf{p}$ and the nearest point in the point clouds $\hat{\mathcal{P}}$. Note that $\tilde{\mathcal{P}}_i$ is a concatenation of point clouds of all foreground objects in scene $i$. To obtain individual point clouds for each object, we apply a clustering method. Specifically, with user-specified object number $M$, we use the DBSCAN Ester et al. (1996) clustering algorithm to identify $M$ object point clouds as:

$$\{\tilde{\mathcal{P}}_{i,j}\} = \mathrm{DBSCAN}_M(\tilde{\mathcal{P}}_i) \tag{9}$$

which satisfies the equation $\sum_j \tilde{\mathcal{P}}_{i,j} = \tilde{\mathcal{P}}_i$, where $\sum$ is the concatenation operation for point clouds.

### B.3 Foreground object matching and alignment

So far, we have obtained $M$ point clouds corresponding to foreground objects in each scene. However, since the $M$ point clouds obtained from the DBSCAN algorithm are unordered, object matching is required to obtain the correct registration of corresponding object point clouds.

Here, we propose an algorithm that simultaneously solves the issues of cross-scene object point cloud registration and matching. Without loss of generality, we take the coordinates and clustering order of the objects in the first scene as the reference and align the objects in scene $i = 2, ..., N$ with the corresponding objects in the reference scene. We approach this as a sequential point cloud registration problem. For simplicity, we consider the matching and registration of the foreground objects in the second scene to the reference scene. Since the correspondence of $M$ object point clouds in these two scenes is unknown, we pair and register them one by one, computing the registered pose

and fitness score $s_{jj'}$. Here the subscript $jj'$ denotes the registration between the point cloud pair $(\tilde{\mathcal{P}}_{1,j}, \tilde{\mathcal{P}}_{2,j'})$ where $j, j' \in \{1, ..., M\}$. We solve the object matching as a bipartite matching problem between the second scene and the reference scene, aiming to maximize the overall registration fitness. More specifically, for the $j$-th object in the reference scene, we pair it with $j'$-th object in the second scene, align them, and calculate the aligned pose $Q$ with the corresponding fitness score $s$. This process is computed for all $j, j' \in \{1, \cdots, M\}$, resulting in a total of $M \times M$ relative poses and corresponding fitness values, which are then recorded in $\{\mathbf{Q}\}$ and $S$, respectively. Next, we use bipartite matching on the obtained cost matrix $S \in [0, 1]^{M \times M}$ to calculate the object matching that maximizes the overall fitness, and finally obtain the optimal correspondences and relative poses. Based on the optimal matching results, we rewrite the objects point clouds in a consistant indexing order as $\{\tilde{\mathcal{P}}_{i,j}\}$. Also the relative poses of the objects of the second scene w.r.t. the reference scene $\{T_{2,j}\}$ are obtained.

For the rest of scenes of $i \geq 3$, we repeat the above procedure to get the registered poses of all objects in all scenes $\{T_{i,j}\}$ where $T_{i,j} \in \mathrm{SE}(3)$. Since we take the first scene as the reference coordinates so that $T_{1,j} = \mathbb{I}^{4\times4}$ is an identity matrix.

The pseudo-code is shown in Alg. 1. We denote the point cloud registration algorithm as a function `register`$(\cdot, \cdot)$, which takes two point clouds $\mathcal{P}_1$ and $\mathcal{P}_2$ as inputs and outputs their relative pose $Q \in \mathrm{SE}(3)$ and the registration fitness score $s \in [0, 1]$.

**Algorithm 1:** Foreground object matching and alignment

**Input :** Foreground object point clouds $\{\tilde{\mathcal{P}}_{i,j}\}$
**Output:** Foreground object poses $\{T_{i,j}\}$
$T_{1,j} \leftarrow \mathbb{I}^{4\times4}$
**for** $i = 2, \cdots, N$ **do**
  $S \leftarrow \mathbf{0}^{M\times M}$
  $\{\mathbf{Q}\} \leftarrow \{\mathbb{I}^{4\times4}\}_{j=1,\cdots,M;\ j'=1,\cdots,M}$
  **for** $j = 1, \cdots, M$ **do**
    **for** $j' = 1, \cdots, M$ **do**
      $(Q, s) \leftarrow$ `register`$(\tilde{\mathcal{P}}_{1,j}, \tilde{\mathcal{P}}_{i,j'})$
      $S_{jj'} \leftarrow s \in [0, 1]$
      $\{\mathbf{Q}\}_{j,j'} \leftarrow Q \in \mathrm{SE}(3)$
    **end**
  **end**
  $index \leftarrow$ `bipartite_matching`$(S)$
  **for** $j = 1, \cdots, M$ **do**
    $T_{i,j} \leftarrow \{\mathbf{Q}\}_{j,index[j]}$
  **end**
**end**
**return** $\{T_{i,j}\}$

## C IMPLEMENTATION DETAILS

**Radiance field representation** We implement our method using Instant-NGP Müller et al. (2022), a state-of-the-art radiance field representation with fast optimization, where each scene can be optimized within a few minutes in order of magnitude.

**Visibility field representation** Given the recent advances in using an explicit grid to represent neural field, we also use an explicit grid to model the proposed visibility field. Specifically, for the target region we discretize the space into $64^3$ grid points, each of which holds the computed visibility of that point. Trilinear interpolation is used to compute the visibility of any point in the continuous 3D space. We observe that, in the original formulation of visibility (2), the discontinuous nature of $V(x)$ (*i.e.*, the visibility may abruptly change from visible to invisible near the surface of an object) may leads to discontinuous scene fusion and poor rendering results. Therefore, we apply smoothing to the computed visibility field. Please refer to the supplementary material for more discussion.


Figure 7: **Visualization of scene labels.** We show the correspondence between different parts of the fused scene and the original scenes. (Top) three original input scenes. (Bottom) background and foreground objects with fused scene, the point cloud labels, and the segmented original scenes. We can observe that, artifacts can appear when rendered from certain viewpoints due to occlusions in the input scenes, while our method accurately segments the parts of the space with higher visibility based on the proposed visibility field, thus synthesizing scenes without occlusion.

## D FUSED SCENE LABELS VISUALIZATION

As our method fuses multiple scenes into one, we show in Fig. 7 which parts of the fused scene come from which original scene. Specifically, for the background and foreground point clouds obtained as Fig. 6, we compute the visibility of each point at their corresponding positions in each original scene, and assign the label of the scene with the highest visibility to that point. The results demonstrate that our method can accurately capture the unoccluded parts in each scene, leading to clean background and 360° foreground object rendering. Moreover, it also implies that our simple yet versatile concept of visibility field can accurately quantify visibility information in 3D space, which may benefit future research in various fields.

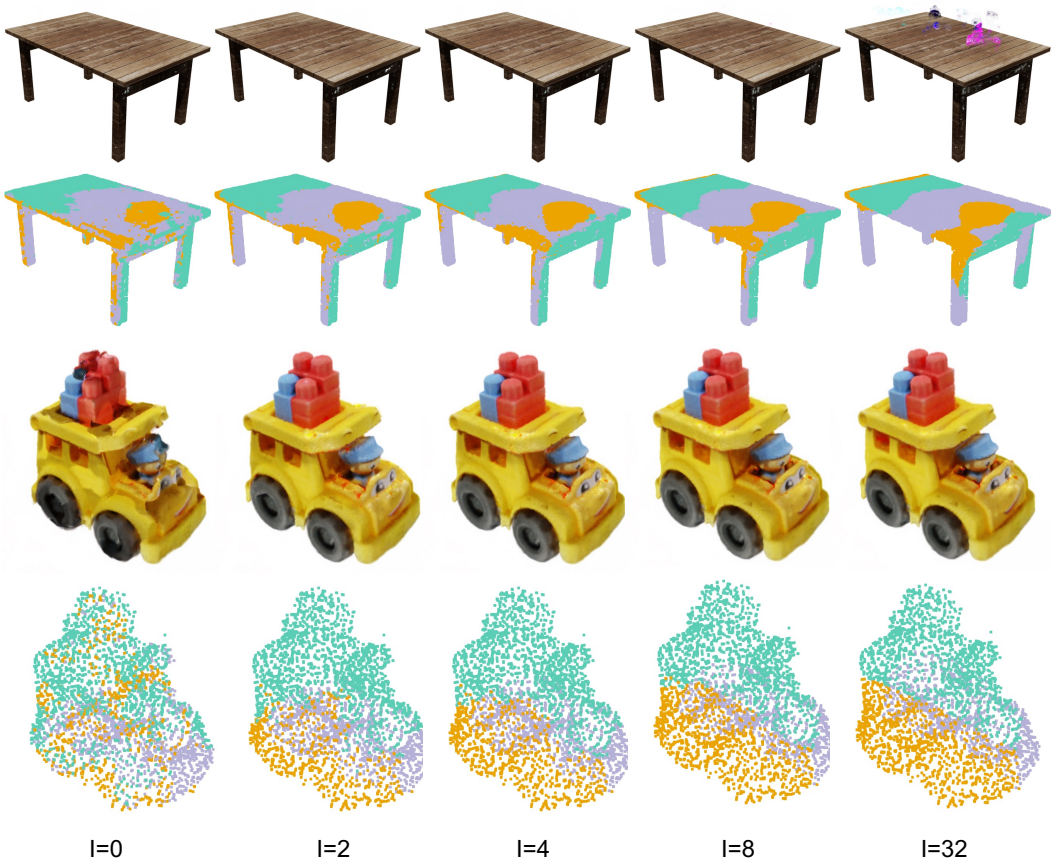


Figure 8: **Ablation on number of iteration for visibility field smoothing.** Also refers to Fig. 7 for the details of labels.

## E VISIBILITY FIELD SMOOTHING

In the formulation of visibility (Eq. (4) in the main paper), $V_l(\mathbf{x})$ exhibits a step function near the object surface (*e.g.*, from visible to invisible), resulting in discontinuous changes in visibility near the surface. Such discontinuous values make the visibility of the object surface ambiguous between different scenes, leading to an ineffective comparison of visibility between scenes. To address the issue of discontinuity, we smooth the visibility field in our implementation. Specifically, the visibility field is implemented by 3D grid, we apply a discrete Laplacian smoothing to its grid points. For the visibility $v_{d,h,w}$ on the grid point, at each iteration its is updated as:

$$v_{d,h,w} = \frac{1}{6} \sum v_{\mathcal{N}(d,h,w)}, \tag{10}$$

where $\mathcal{N}(d,h,w)$ denotes the six adjacent grid points to the grid $(d,h,w)$, *i.e.*, $(d \pm 1, h, w), (d, h \pm 1, w), (d, h, w \pm 1)$. We repeat the above update $I$ times to obtain the smoothed visibility field for subsequent fusion computation.

We present ablation study on the smoothing iterations $I$ of the visibility field in Fig. 8. The visibility field is implemented as a $64^3$ grid. We can observe that without smoothing, the scene labels near the object surface are very noisy, which also indicates that the visibility values near the surface are ambiguous without smoothing, resulting in the incorrect selection of scenes with higher visibility. As we increase the number of iterations for smoothing, we can observe that the scene labels and rendering results become smoother. This suggests that it pays more attention on the overall visibility of the surrounding area rather than the visibility of just a single point. However, excessive smoothing (*i.e.*, smoothing 32 times for a $64^3$ grid) can make the visibility between scenes too similar to distinguish, resulting in artifacts. Specifically, our setup assumes that the main difference between scenes lies in

the foreground, while the background remains consistent. Smoothing can be seen as averaging with the surrounding areas, hence excessive smoothing can weaken the differences between foreground objects, causing the calculated visibility field to become similar across all scenes and affecting the scene fusion result.

In practice, we perform $I = 4$ iterations of smoothing for all experiments.

## F MORE ABLATIONS AND DISCUSSIONS

**Shadow removal** The proposed visibility-aware rendering can achieve shadow removal by selecting shadow-free area. We assume that this is because areas with shadows typically have lower visibility (due to occlusion by objects), and therefore they can be replaced by parts of other scenes without shadows. This example demonstrates that our method is effective in removing shadows from the rendered scenes while preserving the parts that are illuminated by the light.