1. Trained Object Models

This section provides additional visualizations for the trained object models. Training loss for the chair category is visualized in Fig. 1, which shows the loss is decreasing and stabilizes around 40,000 epochs.

Fig. 2 visualizes the rendering results for some chairs in the training set. It shows that the scale of the primitive-based representation varies proportionally with the high-resolution representation.

Fig. 3 visualizes the rendering results for sofas in the training set. There is a lack of shape variation since the majority of sofas have similar structure. Nevertheless, the ellipsoid for the angle sofa is still different with that of other sofas.

Fig. 4 visualizes the rendering results for tables in the training set. Similar to sofas, the variation is limited due to similar table shapes. Nonetheless, the ellipsoid for the rounded table is different from the rest.

Fig. 5 visualizes the rendering results for trashbins in the training set. It could be observed that the ellipsoid shape varies based on the object shape, for instance, the ellipsoid is elongated for a tall trashbin.

Fig. 6 visualizes the rendering results for displays in the
training set. The ellipsoid is rounded for the thicker display and is very thin for the rest.

Fig. 7 visualizes the rendering results for cabinets in the training set. The ellipsoid varies according to the different cabinet shapes.

2. More Qualitative Results on ScanNet

This section presents more qualitative results on ScanNet [2]. Fig. 8 shows a reconstruction with table, trashbins, and cabinet. The cabinet and trashbins are reconstructed well, as can be seen from the resulting meshes which resemble the original object shapes. However, the table is poorly reconstructed, since the shape is quite different and the pose is inaccurate. This is because the available observation in the scene for the table is very limited, as can be seen in the segmented mesh, which is insufficient for optimization.

A ScanNet scene with bookshelves and tables are shown in Fig. 9, to demonstrate the usefulness of the coarse and fine level residuals. The figure illustrates that the initialized object pose and shape are different from the actual scene, since the two bookshelves in the center are not parallel and are too small compared to the observation. In contrast, the bookshelves become larger after applying the fine level residual, which is more consistent with the observations. The reconstructions are further improved with both the coarse and fine level residuals, where the bookshelves become parallel. Moreover, the bottom bookshelf and the top right table also become thinner, which agrees more with the observation. This example clearly shows the effectiveness of the proposed bi-level model for joint object pose and shape optimization.
3. Pose Estimation Metric

This section presents the metric used to evaluate the object pose, which follows Scan2CAD [1]. We introduce the details on how to decompose a pose \( T \in \text{SIM}(3) \) into rotation \( q \), translation \( p \) and scale \( s \) and the error functions for each element separately. For rotation and scale, \( R_s = \text{PTP}^T \):

\[
\begin{align*}
    s_1 &= \|R_se_1\|_2, \quad s_2 = \|R_se_2\|_2, \quad s_3 = \|R_se_3\|_2, \\
    R_{e1} &= \frac{R_se_1}{s_1}, \quad R_{e2} = \frac{R_se_2}{s_2}, \quad R_{e3} = \frac{R_se_3}{s_3}.
\end{align*}
\]

(1)

Suppose \( R = \{m_{ij}\}, i, j \in [1, 2, 3] \), we transform it to quaternion \( q \) by

\[
q_0 = \frac{\sqrt{1 + R_{e1}}}{2}, \quad q_1 = \frac{m_{23} - m_{32}}{4s_0}, \quad q_2 = \frac{m_{31} - m_{13}}{4s_0}, \quad q_3 = \frac{m_{12} - m_{21}}{4s_0}.
\]

(2)

Suppose the prediction and groundtruth are \( q_{pred}, q_{gt} \), we compute the difference by

\[
e_{\text{SO}(3)}(q, \hat{q}) := 2 \arccos(|q_{gt}^Tq_{pred}|).
\]

(3)

Translation is \( p = T[1 : 3, 4] \), and we compare the difference between prediction and groundtruth by

\[
\|p_{pred} - p_{gt}\|_2.
\]

(4)

For scale percentage error, we compute it by

\[
100 \times \frac{1}{3} \sum_{i=1}^{3} \bar{s}_i - 1,
\]

(5)

where \( \bar{s}_i = \frac{s_{pred}}{s_{gt}} \) for each of \( s_1, s_2, s_3 \) recovered from the \( \text{SIM}(3) \) matrix.

4. Timing

<table>
<thead>
<tr>
<th></th>
<th>Init</th>
<th>Latent Code Opt</th>
<th>SIM(3) Opt</th>
<th>SDF Decoding</th>
<th>Meshing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sec)</td>
<td>0.04</td>
<td>0.13</td>
<td>0.58</td>
<td>1.38</td>
<td>2.34</td>
</tr>
</tbody>
</table>

Timing for one instance is provided in Table 1. Init is the pose initialization in (14) for 100 views. Latent Code Opt and SIM(3) Opt are a single SGD step with respect to \( \delta z \) and \( T \) respectively using 10000 points as batch size. SDF Decoding and Meshing are optional steps that generate SDF predictions over 256\(^3\) points and apply Marching Cubes to generate a mesh. Our approach does not currently operate in real-time but it is more efficient than existing work. We will investigate how to accelerate the current slow python \( \text{SIM}(3) \) optimization.

References
