Supplementary Material: Automatically Annotating Indoor Images with CAD Models via RGB-D Scans

Stefan Ainetter(1), Sinisa Stekovic(1), Friedrich Fraundorfer(1), Vincent Lepetit(2,1)
(1)Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria
(2)LIGM, École des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France
{stefan.ainetter, sinisa.stekovic, fraundorfer}@icg.tugraz.at, vincent.lepetit@enpc.fr

1. Additional Information about Scene Preprocessing

As described in the main paper, we use a 3D oriented bounding box as initialization for position, scale and orientation of the target object, whereas 3D semantic instance segmentation is used as geometrical information to replace the target object in the scene with the CAD models. We use either the 3D bounding box information or the instance segmentation provided by a given dataset, which allows us to calculate an approximation for the other missing part.

1.1. Extraction of 3D Bounding Box from 3D Instance Segmentation

Given the 3D instance segmentation of a target object in an indoor scene, we want to calculate the 3D oriented bounding box which fits the object well, as initialization for position, scale and orientation. We use Trimesh [1] to directly extract an initial oriented bounding box from the 3D points of the target object. This initial box is already oriented, however, due to the fact that 3D pointclouds are often incomplete, it might not be well aligned with the target object.

Box alignment. We use the domain knowledge that most furniture is aligned to the scene floor to rectify the initial box so that the up-axis of the object points in the same direction as the up-axis of the indoor scene. This is done by calculating the cosine similarity between the basis vectors of the initial box and the basis vector of the up-axis of the scene. After finding the basis vector most similar to the up-axis of the scene, we adjust the corners of the initial bounding box accordingly, so that both up-vectors point in the same direction.

After adapting the orientation of the box, we need to adapt the scale. One problem when extracting oriented bounding boxes from instance segmentation is that boxes are often too small, as for example legs of chairs and tables are often missing. To avoid this, we automatically extend all boxes for specific furniture classes (table, sofa, chair, bed) to the floor. Figure 1 shows an example how we generate a 3D oriented bounding box from object instance segmentation. Note that after calculating the bounding box, we know that center, scale and up-vector of the box is correct, however, we do not know if the other basis vectors match the canonical pose of the ShapeNet CAD models. Therefore, we generate four box proposals for each target object by rotating the box around the up-axis by $[0^\circ, 90^\circ, 180^\circ, 270^\circ]$. We can then rely on our CAD retrieval method to select the CAD model with correct orientation according to our objective function. Also note that the extracted bounding box does not have to be very accurate, as we refine the pose of each CAD model after retrieval.

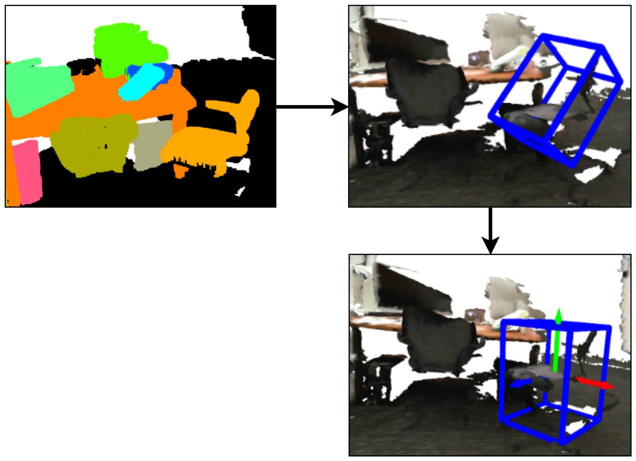
Figure 1. Example for approximating the 3D oriented bounding box from the 3D object instance segmentation. Given the instance segmentation of the orange chair, we use [1] to extract an initial 3D bounding box. Then, we align the up-axis of object and scene and extend the initial box to the floor if needed.

1.2. Extraction of 3D Instance Segmentation from 3D Bounding Box

We directly assign all 3D points inside a given 3D box to the instance segmentation mask of this corresponding ob-

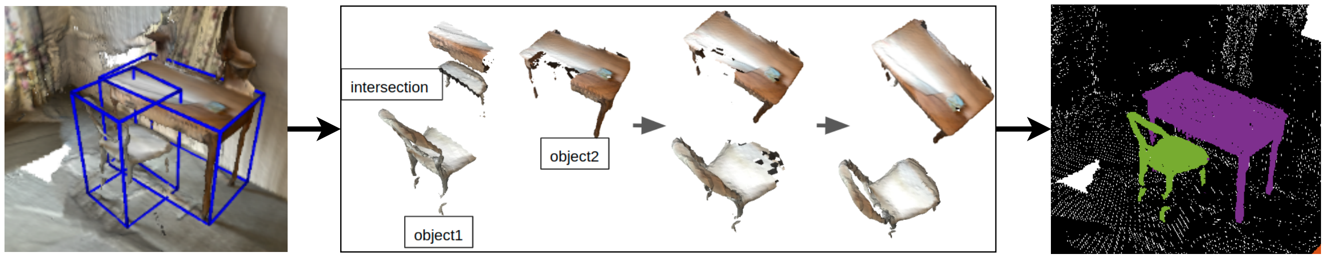

Figure 2. Example for approximating the 3D object instance segmentation from the 3D bounding box. Points which are part of only one bounding box are assigned to the corresponding object. If the bounding boxes of multiple objects overlap, we iteratively assign each point in the intersection to the corresponding object according to the minimal L1 distance between 3D points of intersection and 3D points of the objects.

ject. If boxes of multiple objects overlap, we have to assign the points which are in the intersection of these bounding boxes to the correct object. This is done by first calculating which points are located in the intersection of these boxes. For each point in the intersection pointcloud, we search the nearest neighbour 3D point in the first and second object, using the L1 norm. The point in the intersection box with the lowest distance, either to a point in the first or second object, will be added to this corresponding object pointcloud and removed from the intersection pointcloud. We do this iteratively until all points in the intersection pointcloud are assigned to the corresponding object. Figure 2 shows an example of our method for approximating the 3D object instance segmentation mask for overlapping objects.

2. Additional Visualizations of our Fine-grain Comparison

Figures 3, 4 show additional visualizations from our fine-grain comparison of results with high deviation to Scan2CAD. We show the RGB-D scan, the 3D overlay of the Scan2CAD object (in green) and the 3D overlay of the CAD model retrieved with our method (in blue), and below, the 2D reprojections of the CAD models. We leave it to the reader to assess the quality of these examples.

3. Visualizations of Additional Annotations for ScanNet

Figures 5, 6 show results for full scene CAD retrieval. In each row, the RGB-D scan is on the left side, Scan2CAD annotations are in the middle (in green), and our results are on the right, whereas red CAD models are objects not annotated in Scan2CAD.

4. Additional Visualizations for ARKitScenes Dataset

Figure 7 shows additional visualizations of results from our method for the ARKitScenes dataset, which provides no ground-truth for CAD retrieval.

5. Information about ShapeNet Database for CAD Model Retrieval

Table 1 provides an overview about which classes from the Shapenet database are present in ScanNet, Scan2CAD and ARKitScenes datasets. Additionally, the number of CAD models used for CAD model retrieval is stated for each class.

References

[1] Dawson-Haggerty et al. trimesh.

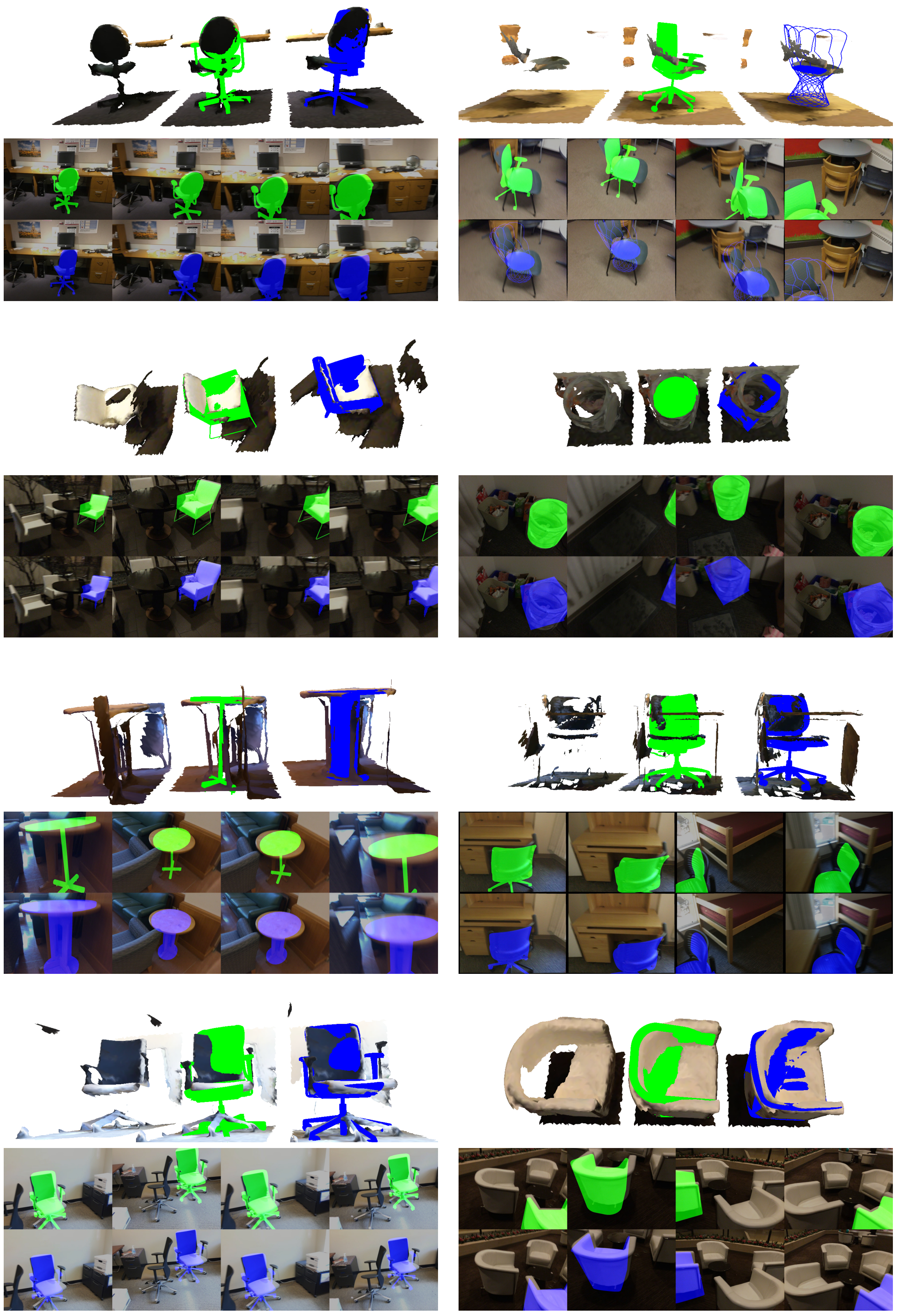

Figure 3. Additional visualizations from our fine-grain visual comparison. For each example, we show the RGB-D scan, the 3D overlay of the Scan2CAD object (in green) and the 3D overlay of the CAD model retrieved with our method (in blue), and below, the 2d reprojections of the CAD models.

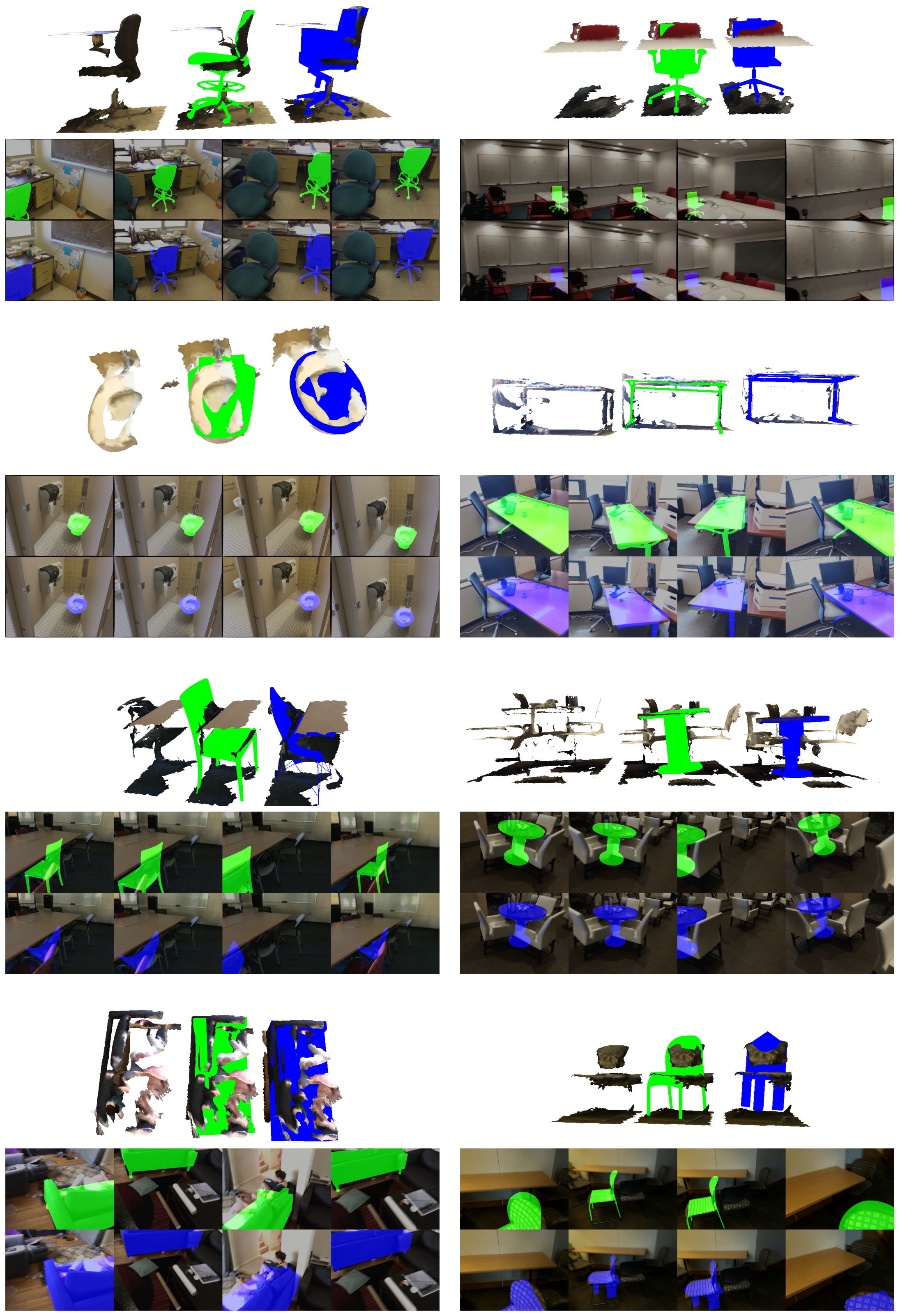
Figure 4. Additional visualizations from our fine-grain visual comparison. For each example, we show the RGB-D scan, the 3D overlay of the Scan2CAD object (in green) and the 3D overlay of the CAD model retrieved with our method (in blue), and below, the 2d reprojections of the CAD models.

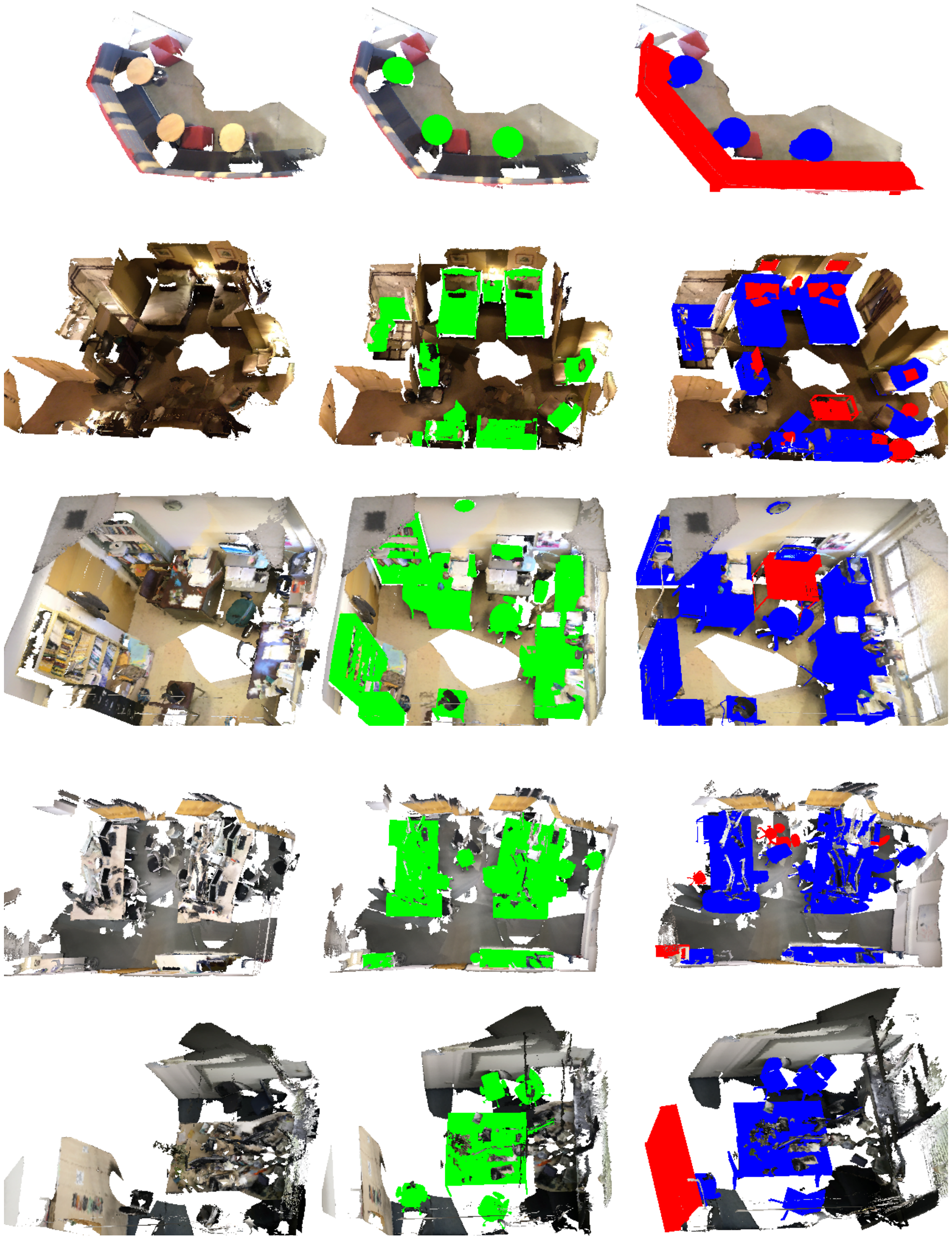
Figure 5. Visualizations showing additional annotations for ScanNet. Left: RGB-D scan. Middle: Scan2CAD annotations. Right: Our results, where red CAD models are for objects not annotated in Scan2CAD.

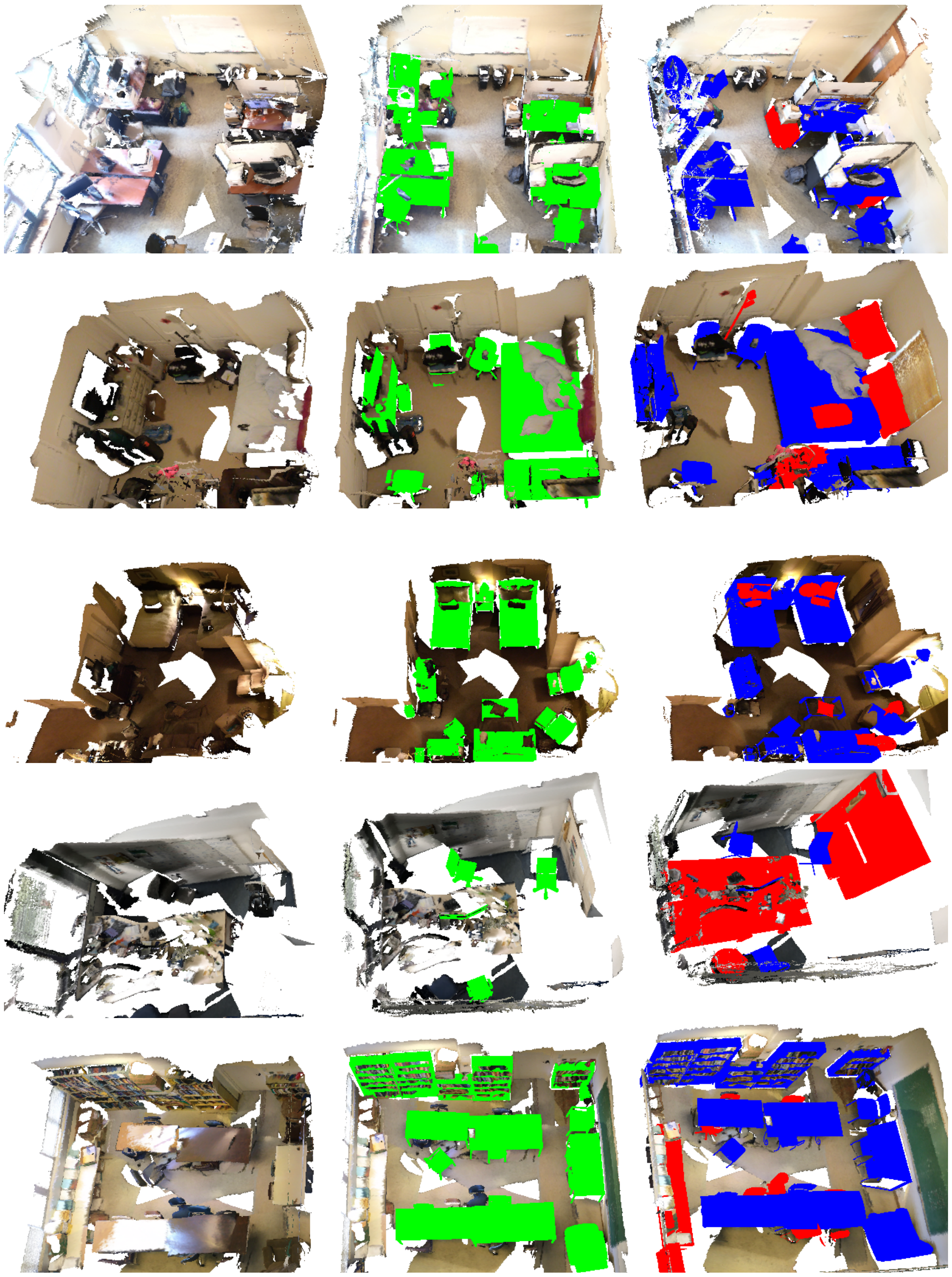

Figure 6. Visualizations showing additional annotations for ScanNet. Left: RGB-D scan. Middle: Scan2CAD annotations. Right: Our results, where red CAD models are for objects not annotated in Scan2CAD.

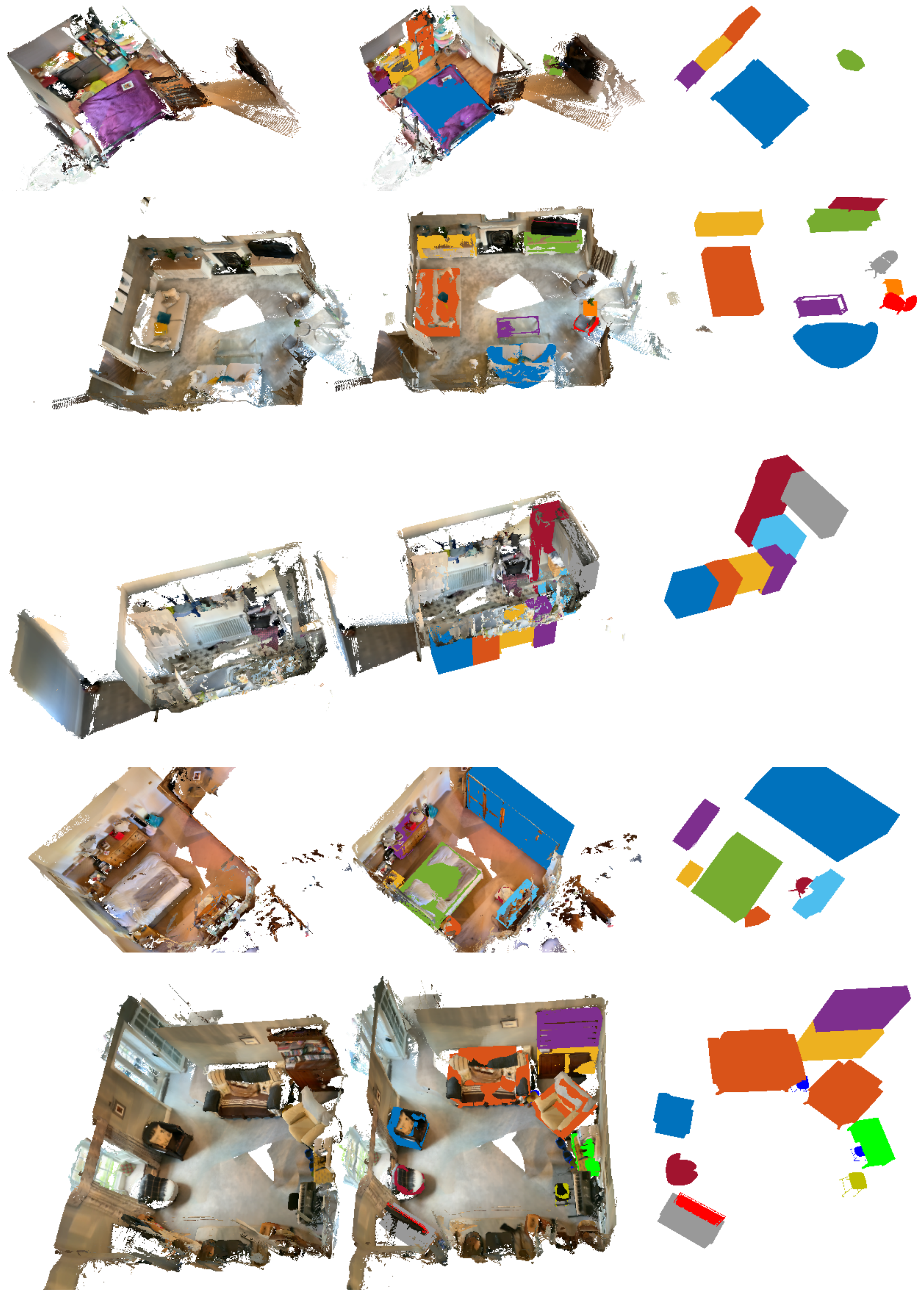
Figure 7. Results of our method for the ARKitScenes dataset. Left: RGB-D scan. Middle: Our results fused with the RGB-D Scan. Right: CAD model retrievals only.

Class	ScanNet	Scan2CAD	ARKitScenes	# CAD Models
Table	✓	✓	✓	8437
Chair	✓	✓	✓	6779
Sofa	✓	✓	✓	3174
Lamp	✓	✓		2319
Bench		✓		1814
Cabinet	✓	✓	✓	1572
Desk	✓			1227
Display	✓	✓	✓	1094
Bathtub	✓	✓	✓	857
Guitar		✓		798
Faucet		✓		745
Clock		✓		652
Flowerpot		✓		603
Dresser	✓			483
Laptop		✓		461
Bookshelf	✓	✓	✓	453
Trash Bin		✓		344
File Cabinet		✓		299
Piano		✓		240
Bed	✓	✓	✓	234
Stove		✓	✓	219
Bowl		✓		187
Washer		✓	✓	170
Printer		✓		167
Microwaves		✓	✓	153
Basket		✓		114
Pillow	✓			97
Dishwasher		✓	✓	94
Bag	✓	✓		84
Counter	✓			71
Keyboard		✓		66
Night Stand	✓			42
Sink	✓		✓	32
Toilet	✓		✓	29
Refrigerator	✓		✓	18

Table 1. List of classes from the ShapeNet database used for our experiments.