ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes Supplemental Material

Angela Dai[1] Angel X. Chang[2] Manolis Savva[2] Maciej Halber[2] Thomas Funkhouser[2] Matthias Nießner[1,3]
[1]Stanford University [2]Princeton University [3]Technical University of Munich

In this supplemental document, we show additional statistics and comparisons for ScanNet. We compare to similar datasets (Sec. 1), provide additional technical details on evaluation tasks and benchmarks (Sec. 2), and provide further detail on the design of our RGB-D reconstruction and annotation framework (Sec. 3).

1. Dataset Statistics and Comparisons

In this section, we provide thorough statistics on the construction and composition of ScanNet dataset, and also compare it to the most similar datasets from prior work.

1.1. Example Annotated Reconstructions

Fig. 1 shows six example annotated reconstructions for a variety of spaces. For each reconstruction, the surface mesh with colors is shown, as well as a visualization with category labels for each object collected using our crowdsourced annotation interface. Category labels are consistent between spaces and are mapped to WordNet [6] synsets. In addition to the category label, separate object instance labels are also available to indicate multiple instances of a given category, such as distinct chairs around a conference table in the fourth row of Fig. 1.

Fig. 2 shows a larger set of reconstructed spaces in ScanNet to illustrate the variety of spaces that are part of the dataset. The scans range from small spaces with just a few objects (e.g., toilets), to large areas with dozens of objects (e.g., classrooms and studio apartments).

1.2. Dataset Construction Statistics

The construction of ScanNet was carried out with the RGB-D acquisition and annotation framework described in the main paper. In order to provide an intuition of the scalability of our framework, we report timing statistics for both the reconstruction and annotation steps. The median reconstruction processing time (including data conversion, dense voxel fusion, surface mesh extraction, alignment, cleanup, and preview thumbnail image rendering) is $11.3\,\mathrm{min}$ for each scene. A few outliers exist with significantly higher processing times (on the order of hours), due to unplanned processing server downtime during our data collection (mainly software updates), resulting in a higher mean reconstruction time of $14.9\,\mathrm{min}$.

After reconstruction is complete, each scan is annotated by several crowd workers on Amazon Mechanical Turk (2.3 workers on average per scan). The median annotation time per crowd worker is $12.0\,\mathrm{min}$ (mean time is $17.3\,\mathrm{min}$, again due to a few outlier workers who take significantly longer). Aggregating the time taken across workers for annotating each of the 1513 scans in ScanNet, the median time per scan is $16.8\,\mathrm{min}$, and the mean time per scan is $22.3\,\mathrm{min}$.

1.3. Dataset Composition Statistics

The construction of the ScanNet dataset is motivated by the lack of large, annotated, densely reconstructed RGB-D dataset of 3D scenes that are publicly available in the academic community. Existing RGB-D datasets either have full scene-level annotations only for a subset of RGB-D frames (e.g., NYU v2 depth [8]), or they focus on annotating decontextualized objects and not scenes (e.g., Choi et al. [3]). The two datasets that do annotate densely reconstructed RGB-D spaces at the scene level are the SceneNN dataset by Hua et al. [7] and the smaller PiGraphs dataset by Savva et al. [10].

SceneNN consists of 94 RGB-D scans captured using Asus Xtion Pro devices and reconstructed with the method of Choi et al. [2]. The resulting densely-fused surface meshes are fully segmented at the level of meaningful objects. However, only a small set of segments are annotated with semantic labels. On the other hand, the PiGraphs [10] dataset consists of 26 RGB-D scans captured with Kinect v1 devices and reconstructed with the VoxelHashing approach of Nießner et al. [9]. This dataset has more complete and clean semantic labels, including object parts and object instances. However, it contains very few scenes and is limited in the variety of environments, consisting mostly of offices and conference rooms. To illustrate the large gap in quantity of annotated semantic labels between these two datasets and ScanNet, Fig. 3 plots histograms of the total number of labeled object instances and the total numbers of unique se-

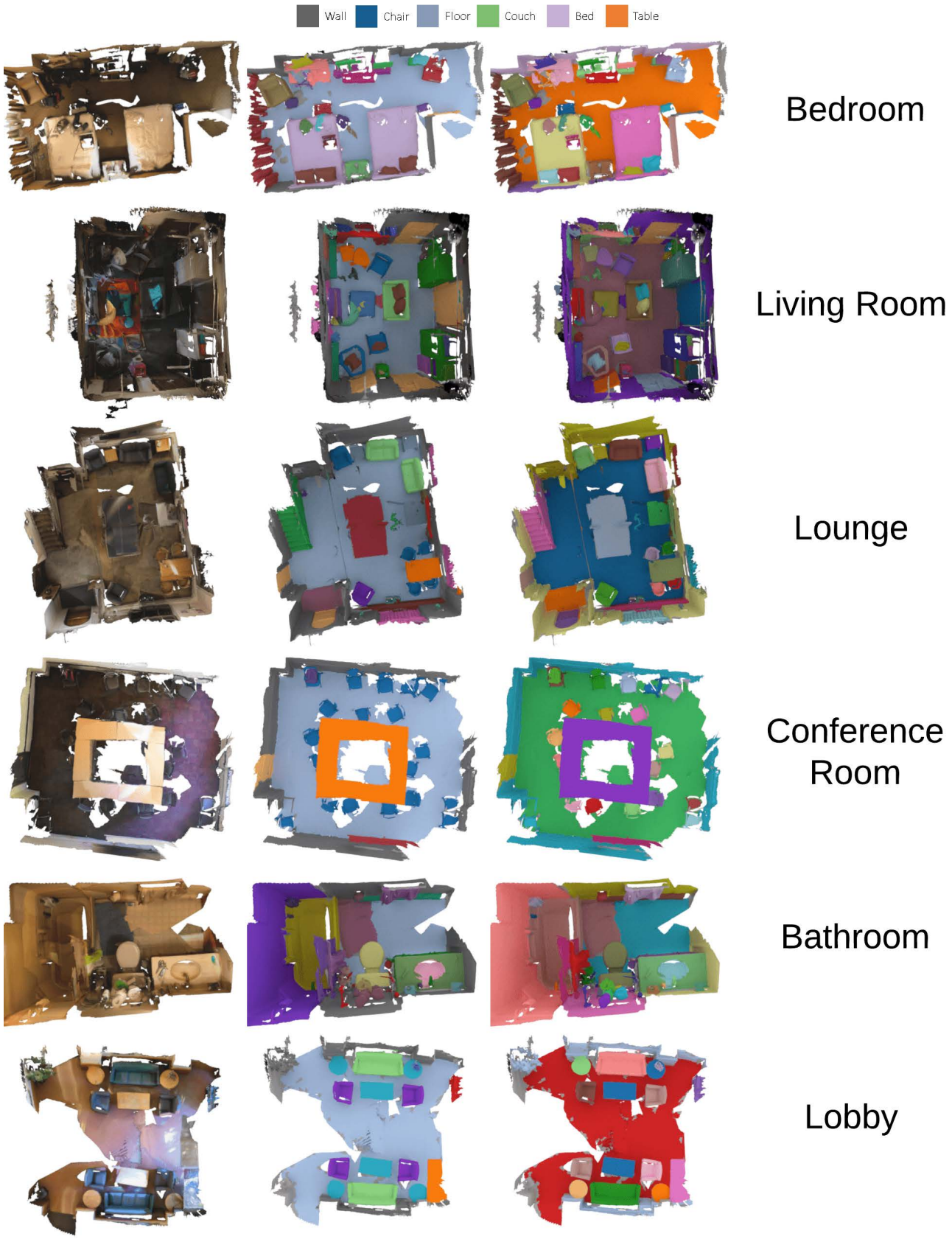

Figure 1. Example annotated scans in ScanNet. **Left:** reconstructed surface mesh with original colors. **Middle:** color indicates category label consistently across all scans. **Right:** each object instance shown with a different randomly assigned color.

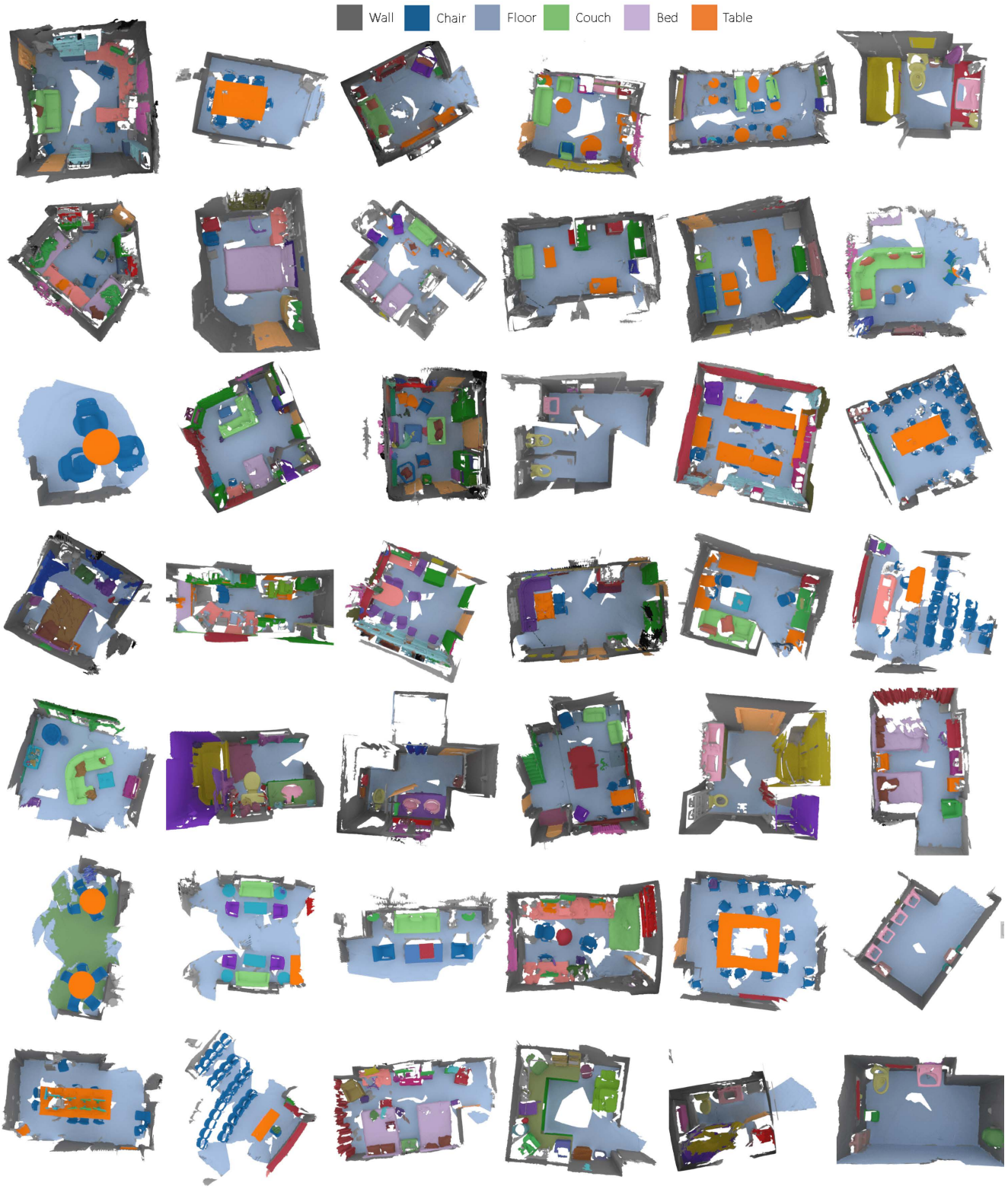

Figure 2. A variety of example annotated scans in ScanNet. Colors indicate category consistently across all scans.

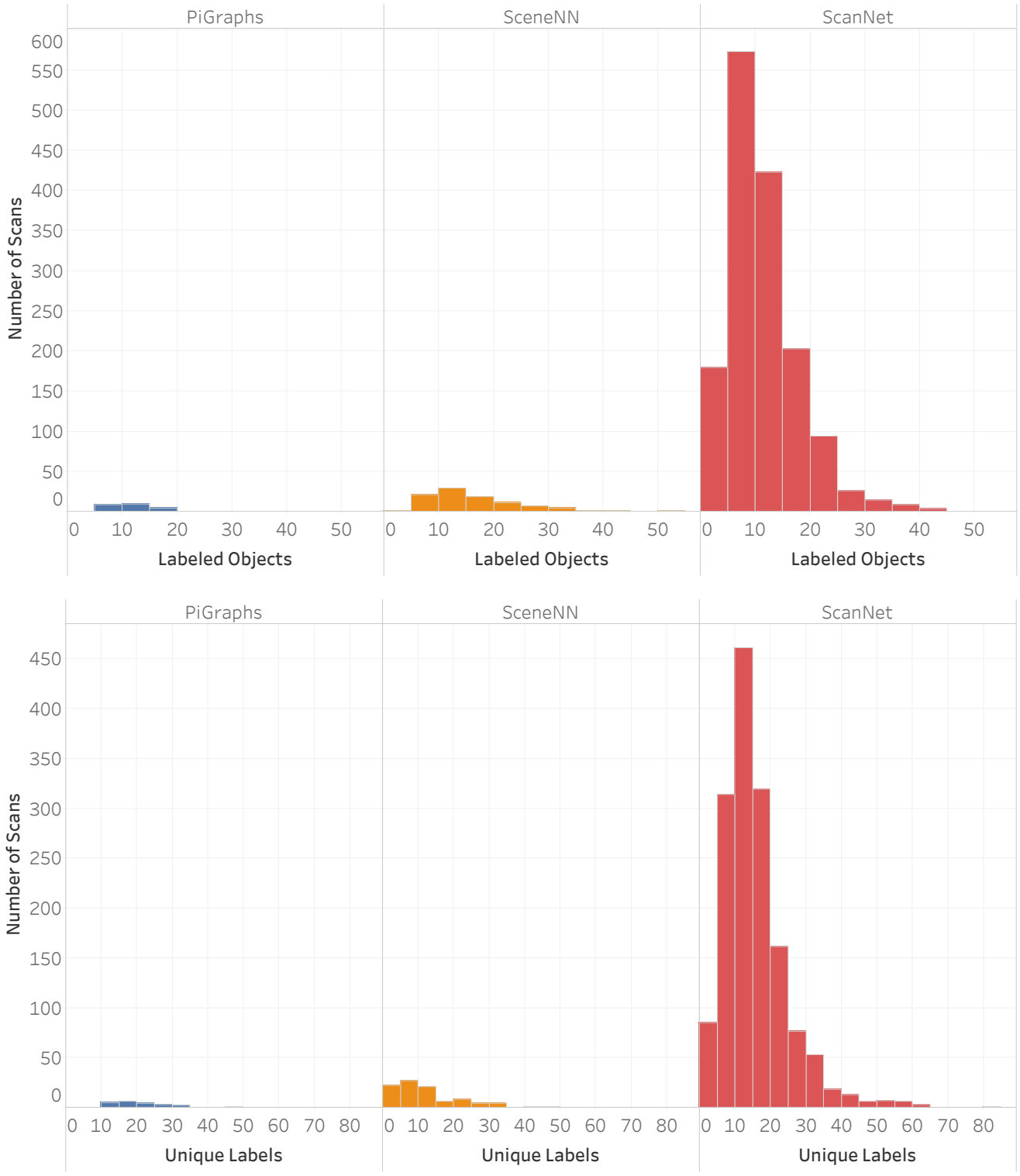

Figure 3. Histograms of the total number of objects labeled per scan (**top**) and total number of unique labels per scan (**bottom**) in the PiGraphs [10], SceneNN [7] and our dataset (ScanNet). The histograms show that ScanNet has many annotated objects over a larger number of scans, ranging in complexity with regards to the total number of objets per scan.

ScanNet		SceneNN [7]	
Category	Count	Category	Count
wall	6226	chair	194
chair	4279	table	53
floor	3212	floor	44
table	2223	seat	41
door	1181	desk	39
couch	1048	monitor	31
cabinet	937	sofa	25
desk	733	cabinet	25
shelf	732	door	24
bed	699	box	23
office chair	669	keyboard	23
trashcan	561	trash bin	21
pillow	490	wall	20
sink	470	pillow	19
window	398	fridge	18
toilet	397	stand	18
picture	351	bag	17
bookshelf	328	bed	16
monitor	308	window	14
curtain	280	sink	13
computer	274	printer	12
armchair	264	computer	12
bathtub	253	chair01	12
coffee table	239	desk1	11
box	231	monitor01	10
dining chair	230	shelves	10
refrigerator	226	shelf	10
book	221	chair1	10
lamp	218	chair02	10
towel	216	fan	9
kitchen cabinet	203	basket	9
drawer	202	desk2	9
tv	187	laptop	9
nightstand	182	trashbin	9
counter	179	kettle	9
dresser	177	microwave	9
clothes	164	monitor1	8
countertop	163	stove	8
stool	130	chair2	8
plant	130	bike	7
cushion	116	blanket	7
ceiling	114	drawer	7
bedframe	111	lamp	7
keyboard	107	wall02	7
end table	105	wall01	7
toilet paper	104	wall04	7
bag	104	backpack	7
backpack	100	cup	7
blanket	94	chair3	7
dining table	94	whiteboard	7

Table 1. Total counts of annotated object instances of the 50 largest categories in ScanNet (left), and in SceneNN [7] (right), the most similar annotated RGB-D reconstruction dataset. ScanNet contains far more annotated object instances, and the annotated labels are processed for consistency to remove duplicates such as "chair01" in SceneNN.

mantic labels for each scan.

In order to demonstrate how our category labels map to other data, we plot the distribution of annotated object labels corresponded to the ShapeNetCore 3D CAD model

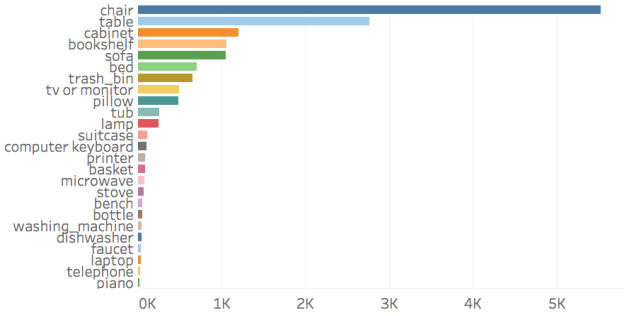

Figure 4. Top 25 most frequent annotation labels in ScanNet scans mapped to ShapeNetCore classes. ScanNet has thousands of 3D reconstructed instances of common objects such as chairs, tables, and cabinets.

categories in Fig. 4. This mapping is leveraged during our CAD model alignment and retrieval task to automatically suggest instances of CAD models from ShapeNet that match the label of a given object category in the reconstruction.

We can also obtain 2D annotations on the input RGB-D sequences by projecting our 3D annotations into each frame using the corresponding camera pose. This way, we obtain an average of 76% annotation coverage of all pixels per scene by using the previously obtained 3D annotations.

1.4. NYUv2 Reconstruction and Comparison

Here, we discuss how ScanNet relates to NYUv2, one of the most popular RGB-D dataset with annotations. In order to compare the data in ScanNet with the data in NYUv2, we reconstructed and annotated all the RGB-D sequences in NYUv2 using our framework. (Note that for 9 sequences of the NYUv2 dataset, our framework did not obtain valid camera poses for $> 50\%$ of the frames, so we did not compute reconstructions and annotations for these sequences.) Moreover, we created a set of surface mesh semantic annotations for the NYUv2 reconstructions by projecting every pixel of the annotated RGB-D frames with valid depth and label into world space using our computed camera poses, and assigning the corresponding object label to the closest surface mesh vertices (within $0.04cm$, using a kd-tree lookup).

We then compare the total surface area of the reconstructed meshes that was annotated using projection from the annotated NYUv2 frames, and using our annotation pipeline. Fig. 5 plots the percentage of reconstructed surfaces in NYUv2 that were annotated with each approach, as well as the percentage distribution for the ScanNet reconstructions for comparison. Note that we exclude the 9 sequences for which we do not have enough valid camera poses.

A noticeable difference between the RGB-D sequences in NYUv2 and those in ScanNet is that overall, the ScanNet

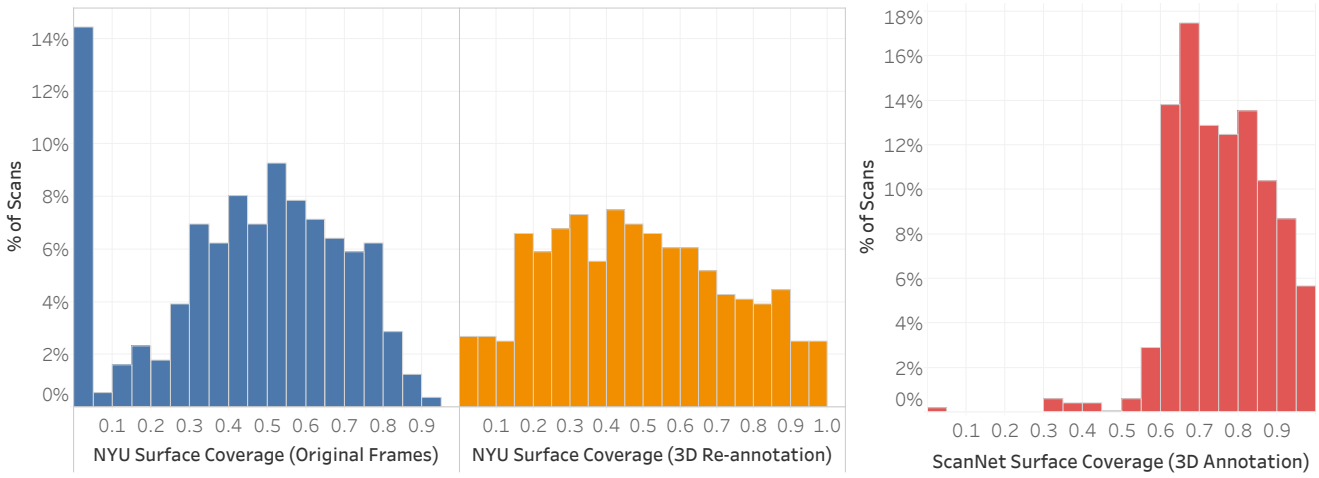

Figure 5. Histograms of the percentage of total reconstruction surface area per scan that is semantically labeled for: NYU v2 reconstructions using projection of RGB-D annotated frames (**left**), for NYU v2 reconstructions using our 3D annotation interface (**middle**), and for ScanNet reconstructions similarly annotated with our interface (**right**).

sequences are more complete surface reconstructions of the real-world spaces. Most importantly, the NYUv2 original frames in general do not cover a sufficient number of viewpoints of the space to ensure full reconstruction of semantically meaningful complete objects. Fig. 6 shows a comparison of several reconstructed scenes from NYUv2 RGB-D sequences vs comparable reconstructions from ScanNet. As shown in the top-down views, the NYU reconstructions are much more sparse than the ScanNet reconstructions. This disparity makes a more direct comparison with ScanNet reconstructions hard to quantify. However, we can conclude that projecting the annotated NYUv2 RGB-D frames to reconstructions is not sufficient to semantically annotate the spaces, as is clear from the far lower surface coverage distribution for NYUv2 in Fig. 5.

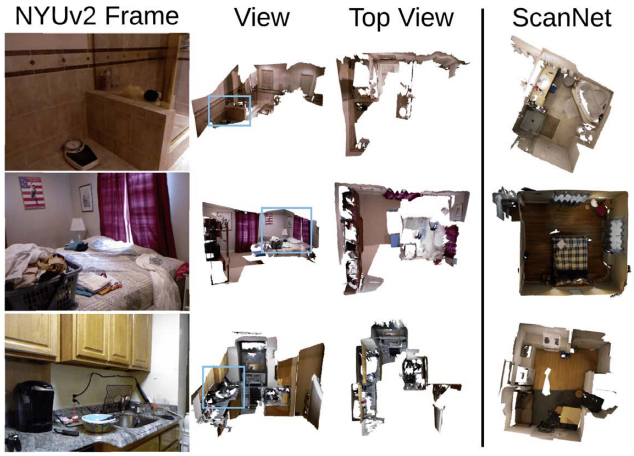

Figure 6. Comparison of reconstructed Bathroom (**top**), Bedroom (**middle**), and Kitchen (**bottom**) from NYUv2 RGB-D frames (**left**), and a comparable reconstruction from ScanNet (**right**). For each NYU scene, we show an example color frame, the rough corresponding region of the view in the reconstructed scene (light blue box), and a top down view of the reconstruction. While NYUv2 reconstruction look complete from some viewpoints, much of the scene is left uncovered (see top down views). In constrast, ScanNet reconstruction have a much more complete coverage of the space and allow for denser annotation.

2. Tasks

Here we provide more details about the 3D scene understanding tasks and benchmarks discussed in the main paper.

2.1. Semantic Voxel Labeling

For the semantic voxel labeling task, we propose a network which predicts class labels for each column of a voxelized scene. As shown in Fig. 7, our network takes as input a $2 \times 31 \times 31 \times 62$ volume and uses a series of fully convolutional layers to simultaneously predict class scores for the center column of 62 voxels. We leverage information from the voxel neighborhood of both occupied space (voxels on the surface) and known space (voxels in front of a surface according to the camera trajectory) to describe the input partial data from a scan.

At test time, we slide the network through a scan through a voxelized scan along the xy-plane, and each column is predicted independently. Fig. 8 visualizes several ScanNet test scans with voxel label predictions, alongside the ground truth annotations from our crowdsourced labeling task.

3. Dataset Acquisition Framework

This section provides more details for specific steps in our RGB-D data acquisition framework which was described in the main paper. To enable scalable dataset ac-

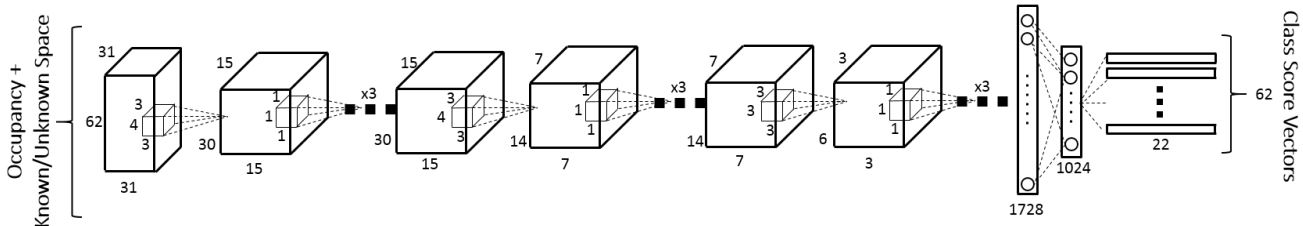

Figure 7. Deep Neural Network architecture for our semantic voxel label prediction task. The network is mainly composed of 3D convolutions that process the geometry of a scene using a 3D voxel grid representation.

quisition, we designed our data acquisition framework for 1) ease of use during capture, 2) robust reconstruction, 3) rapid crowdsourcing, 4) visibility into the collected data and its metadata. For 1) we developed an iPad app (see Sec. 3.1) with an easy-to-use interface, reasonable scanning presets, and minimalistic user controls. To ensure good reconstruction with minimal user interaction during scanning, we tested different exposure time settings and enabled auto white balancing (see Sec. 3.1). We also established a simple calibration process that novice users could carry out (see Sec. 3.2), and offloaded RGB-D reconstruction to the cloud (see Sec. 3.3). Finally, we developed web-based UIs for crowdsourcing semantic annotation tasks as described in Sec. 3.4, and for managing the collected data as described in Sec. 3.6.

3.1. RGB-D Acquisition UI

Fig. 9 shows our RGB-D recording app on the iPad. We designed an iPad app with a simple camera-based UI and a minimalistic set of controls. Before scanning, the user enters a user name, a scene name, and selects the type of room being scanned. The user then presses a single button to start and stop a scan recording. The interface can be toggled between visualizing the color stream and the depth stream overlaid on the color.

We found that the most challenging part of scanning for novice users was acquiring an intuition as to what regions during scanning are likely to result in poor tracking and failed reconstruction. To alleviate this, we added a "progress bar"-style visualization during active scanning which indicates the featurefulness of the region being scanned. The bar ranges from full green, indicating high feature count, to near-empty black, indicating low feature count and high likelihood of tracking loss. This UI element was helpful for quickly familiarizing users with the scanning process. After scanning, the user can view a list of scans on the device and select to upload the scan data to a processing server. During upload, a progress bar is shown and scanning is disabled. Upon completion of the upload, the checksums of scan data on the server are verified against local data and the scans are automatically deleted to provide more memory for scanning.

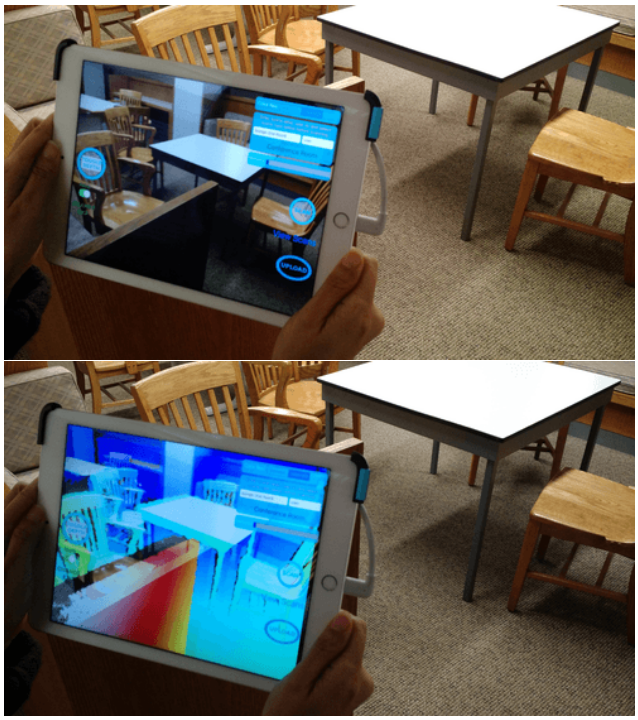

Figure 9. Our RGB-D recording app on an iPad Air2 with attached Structure sensor (showing color stream at the top and depth stream at the bottom). The app allows novice users to record RGB-D videos and upload to a server for reconstruction and annotation.

Auto white balancing and Exposure Settings Another challenge towards performing reconstruction in uncontrolled scenarios is the wide variety of illumination conditions. Since our scanning app was designed for novice users, we opted to provide a reasonable set of presets and allow for manual override only when deemed necessary. By default, we enabled continuous automatic whitepoint balancing as implemented by the iOS SDK. We also enabled dynamic exposure selection again as implemented by the iOS SDK, but instructed users that they could manually adjust exposure if necessary to make overly dark locations brighter, or overly bright locations darker. The exposure setting can have a significant impact on the amount of motion blur during scanning. However, we found that inexperienced users preferred to rely on dynamic exposure, and

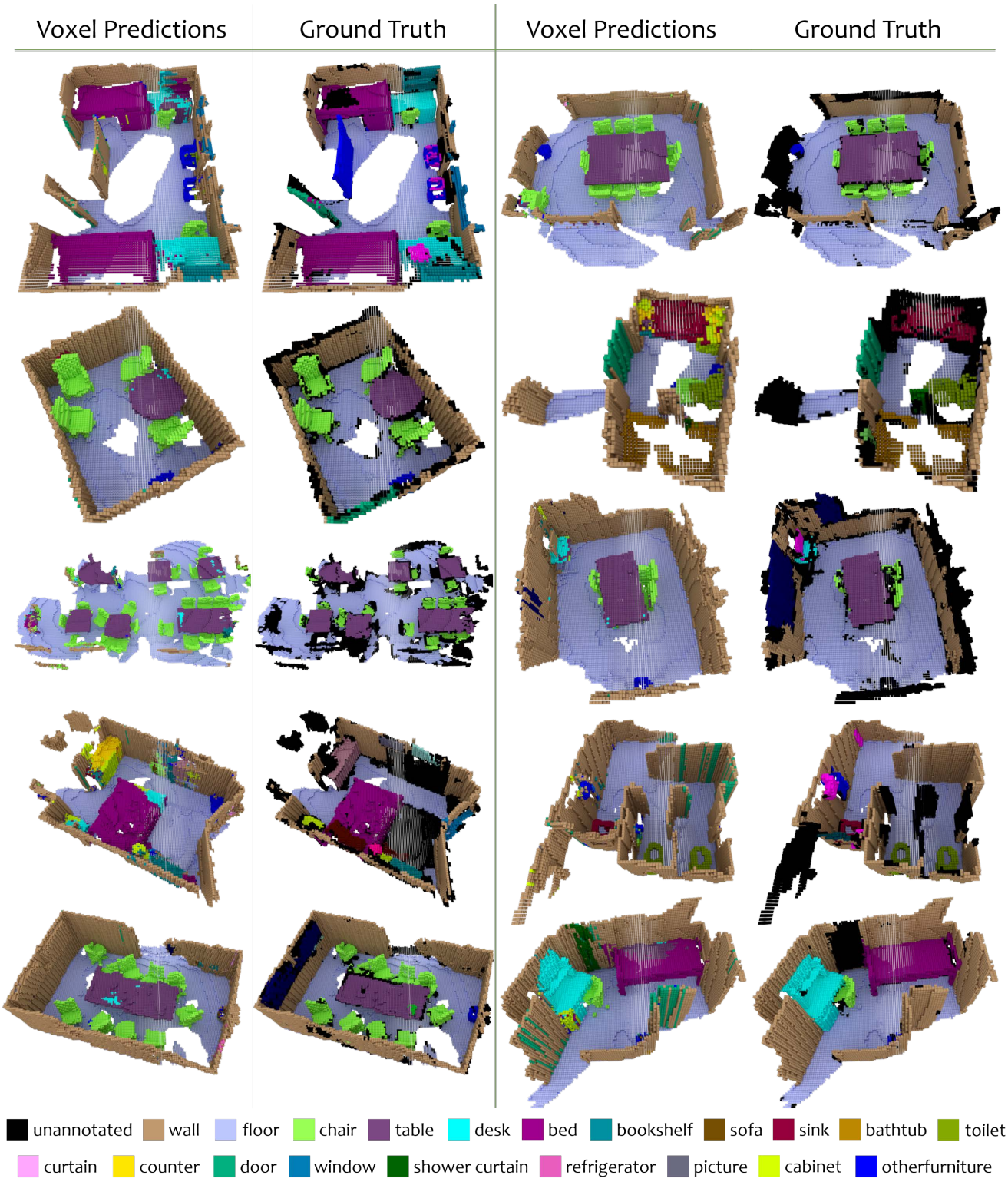

Figure 8. Semantic voxel labeling of 3D scans in ScanNet using our 3D CNN architecture. Voxel colors indicate predicted or ground truth category.

typically moved relatively slowly during scanning, making motion blur less of an issue. The average exposure time during scans with dynamic exposure was close to 30 ms.

3.2. Sensor Calibration

Sensor calibration is a critical, yet often overlooked part of RGB-D data acquisition. Our experiments showed that depth-to-color calibration is an important step in acquiring good 3D reconstructions from RGB-D sequences (see Fig. 10).

Depth To Color Calibration To align a depth image $\mathcal{D}$ to color image $\mathcal{C}$, we need to estimate intrinsic parameters of both sensors, the infrared camera $\mathbf{K}_\mathcal{D}$ and color camera $\mathbf{K}_\mathcal{C}$, as well as extrinsic transformation $\mathbf{T}_{\mathcal{D}\to\mathcal{C}}$. In our experiments we have found that using the set of intrinsic parameters of focal length, center of projection, and two barrel distortion coefficients models worked well for the used cameras. To obtain calibration parameters $\mathbf{K}_\mathcal{D}$ and $\mathbf{K}_\mathcal{C}$ we capture a series of color-infrared pairs showing an asymmetric checkerboard grid. We then estimate calibration parameters for each camera with Matlab's *CameraCalibrator* application. During this procedure we additionally obtain the world positions of calibration grid corners, and use them to estimate the transformation $\mathbf{T}_{\mathcal{D}\to\mathcal{C}}$.

Depth Distortion Calibration Previous work suggests that for consumer-level depth cameras there exists depth-dependent distortion that increases as camera moves away from the surface. Thus, we decided to augment our set of intrinsic parameters for depth cameras with a undistortion lookup table, as first suggested in Teichman et al. [11]. This look up table is a function $f(x, y, d)$, of spatial coordinates x, y and observed depth d, returning a multiplication factor m used to obtain undistorted depth $d' = md$. The table is computed from training pairs of observed and ground truth depths d and d_t. However, unlike Teichman's unsupervised approach, which produces training pairs using carefully taken 'calibration sequences', we decided to design a supervised approach similar to that of Di Cicco [5]. However, we found that at large distances the depth distortion becomes so severe that approaches based on fitting planes to depth data are bound to fail. Thus to obtain training pairs $\{d, d_t\}$, we capture a color-depth video sequence of a large flat wall with a calibration target at the center, as the user moves away and towards the wall. To ensure successfull calibration process user needs to ensure that the viewed wall is the only observed surface and that it covers the entire field of view. With the captured color-depth sequence and previously estimated $\mathbf{K}_\mathcal{D}$, $\mathbf{K}_\mathcal{C}$, $\mathbf{T}_{\mathcal{D}\to\mathcal{C}}$ we can recover the the world positions of the calibration grid corners, effectively obtaining the ground truth plane locations for each of the captured depth images. For each pixel x, y with depth d, we then shoot a ray through x, y to intersect with the related plane. d_t can be recovered from the point of intersection. The rest of our undistortion pipeline follows closely the that of Teichman et al. [11]. We found that undistorting depth images obtained by a Structure sensor leads to significantly improved tracking.

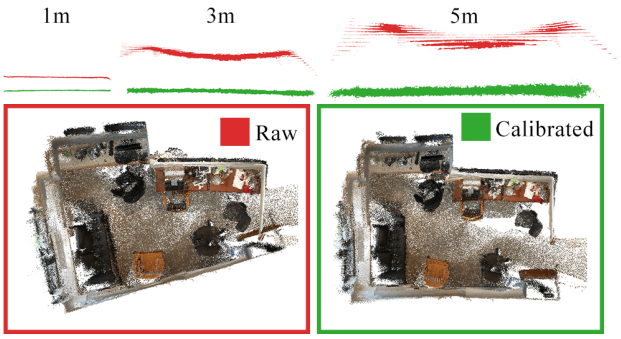

Figure 10. Comparison of calibration results. In the top row, we show results of calibration on a flat wall. As the distance increases the distortion becomes quite severe, motivating the need for depth distortion calibration. In the bottom row, we show results of frame-to-frame tracking on raw and calibrated data.

3.3. Surface Reconstruction

Given a calibrated RGB-D sequence as input, a fused 3D surface reconstruction is obtained using the BundleFusion framework [4], as described in the main paper. The reconstruction is then cleaned by merging vertices within 1 mm of each other, and removing connected components with fewer than 7500 triangles. Following this cleanup step, two quadric edge collapse decimation steps are performed to produce lower triangle count versions of each surface mesh. Each decimation halves the number of triangles in the surface mesh, reducing the size of the original meshes from an average of 146 MB to 5.82 MB for the low resolution mesh. The mesh decimation step is important for reducing data transfer requirements and improving loading times during the crowdsourced annotation using our web-based UI.

3.4. Crowdsourced Annotation UI

We deployed our semantic annotation task to crowd workers on the Amazon Mechanical Turk platform. Each annotation task began with an introduction (see Fig. 11) providing a basic overview of the task. The worker was then shown a reconstruction and asked to paint all object instances with a color and corresponding label. The worker was required to annotate at least 25% of the surface area of the reconstruction, and encouraged to cover at least 50%. Once the worker was done, they could submit by pressing a button. Workers were compensated with $0.50 for each annotation task performed.

The CAD model retrieval and alignment task began with a view of an already semantically annotated reconstruction and asked workers to click on objects to retrieve and place appropriate CAD models. Fig. 12 shows the initial instructions for an example reconstruction with several chairs. Workers for this task were required to place at least three objects before submitting. Once the worker was done, they were compensated with $1.00 for each completed task.

3.5. Label cleaning and propagation

Labeling is performed on the surface mesh reconstruction, with several workers labeling each scan. To ensure that labels are consistent across workers, we use standard NLP techniques to clean up the labels. First, we use a manually curated list of good labels and their synonyms to compute a map to a single canonical label for each set, also including common misspellings by a small edit distance threshold of the given label. Labels with less than 5 counts are deemed unreliable and ignored in all statistics. Labels with more than 20 counts are manually examined and added to the list of good labels or collapsed as a synonym of a good label. The list of these frequent collapsed labels is also mapped to WordNet [6] synsets when possible, and to other common label sets that are commonly used for RGB-D and 3D CAD data (NYUv2 [8], ModelNet [12], and ShapeNetCore [1]).

Using the cleaned labels, we then compute an aggregated consensus labeling of each scene, since any individual crowdsourced annotation of a scene may not cover the entire scene, or may contain some errors. For each segment in the over-segmentation of a scene mesh, we first take the majority vote label. This groups together instances of the same class of objects, so we also compute a labeling purely based on geometric overlap; that is, we greedily take the unions of annotations which have $\geq 50\%$ overlap of segments. We then take the maximal intersections between these two labelings to obtain the final consensus.

After we have obtained the aggregated consensus semantic annotation for a scene, we then propagate these labels to the high-resolution mesh as well as to the 2D frames of the input RGB-D sequence. To propagate the labels to the high resolution mesh, we compute a kd-tree over the mesh vertices of the labeled coarse mesh, and we label each vertex of the high resolution mesh according to a nearest neighbor lookup in the kd-tree. We project the 3D semantic annotations to the input 2D frames by rendering the labeled mesh from the camera poses of each frame, and follow this with a joint dilation filter with the original RGB image and joint erosion filter with the original RGB image.

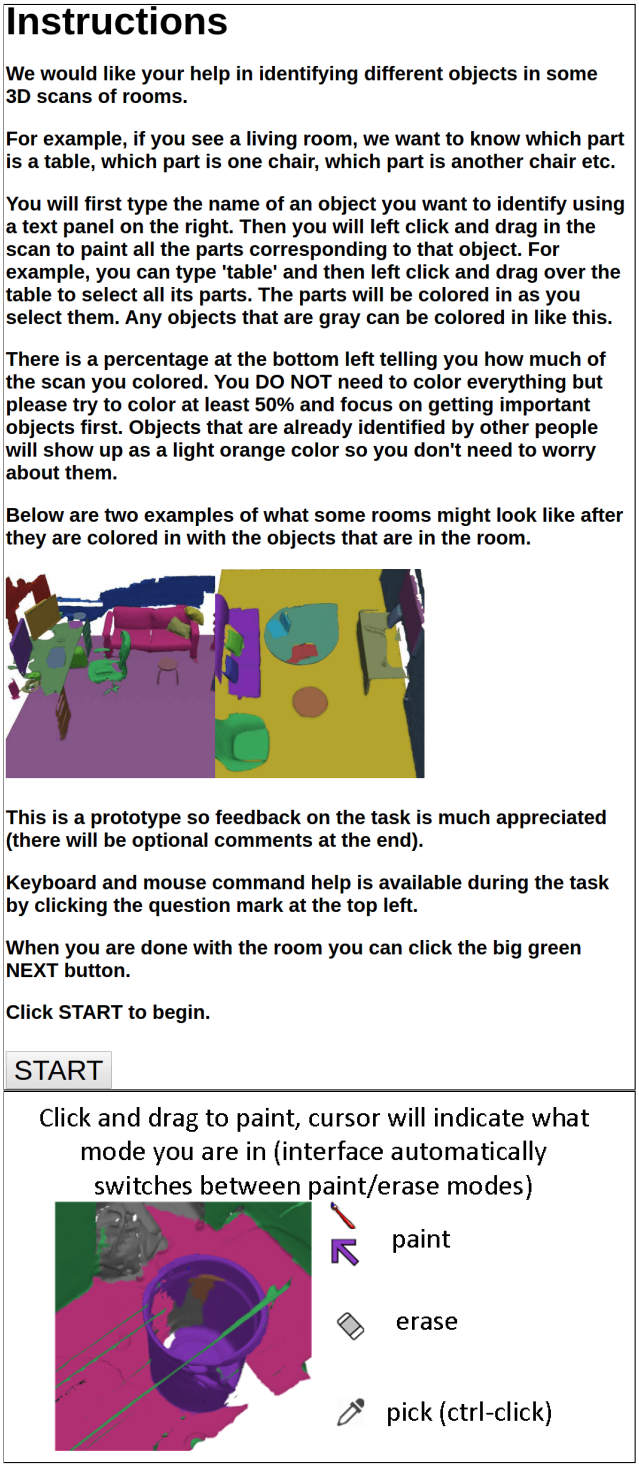

Figure 11. Instructions provided to crowd workers for our semantic annotation task. **Top:** instructions before the beginning of the task. **Bottom:** interface instructions during annotation.

3.6. Management UI

To enable scalability of our RGB-D acquisition and annotation, and continual transparency into the progress of scans throughout our framework, we created a web-based management UI to track and organize all data (see Fig. 14). When a user is finished scanning and presses the upload button on an iPad device, their scan data is automatically

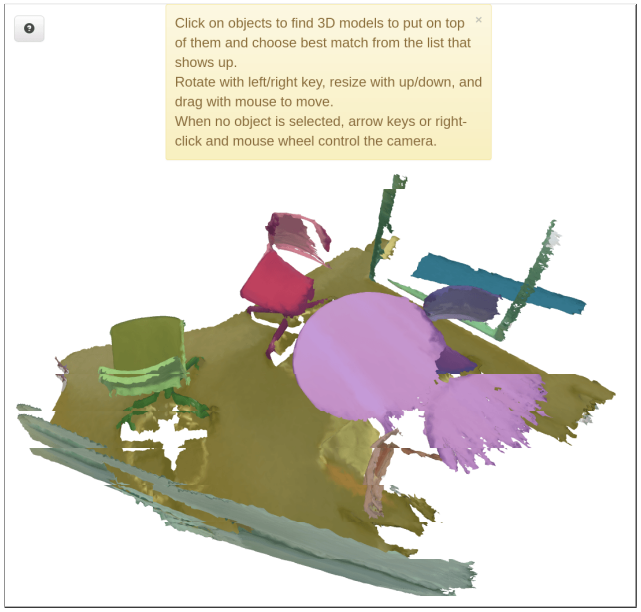

Figure 12. Instructions provided to crowd workers for our CAD model alignment task. The worker clicks on colored objects to retrieve and place CAD models.

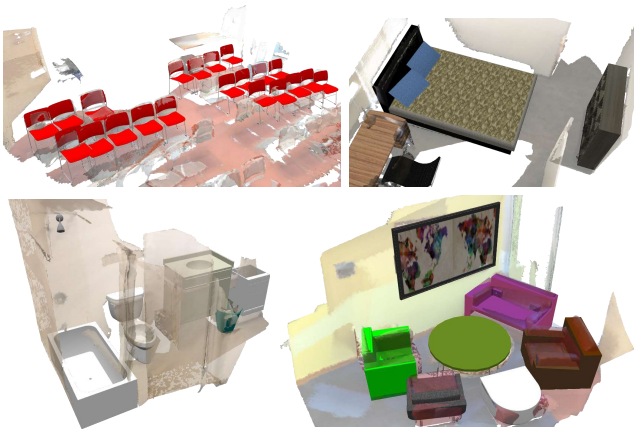

Figure 13. ShapeNetCore [1] CAD models retrieved and placed on ScanNet scans by crowd workers (scan mesh is transparent and CAD models are opaque). From top left clockwise: a classroom, bedroom, bathroom, and lounge scan.

uploaded to our processing server, placed into a reconstruction queue, and immediately made visible in the management UI. As the reconstruction proceeds through the various stages of data conversion, calibration, pose optimization and RGB-D fusion, alignment, cleanup, decimation, and segmentation, progress is visualized in the management UI. Thumbnail renderings of the generated surface reconstruction, and statistics such as total number of frames, reconstructed floor area etc. are automatically computed and can be used for filtering and sorting of the reconstructions. Similarly, during crowdsourced annotation, worker progress and aggregated annotated surface area statistics

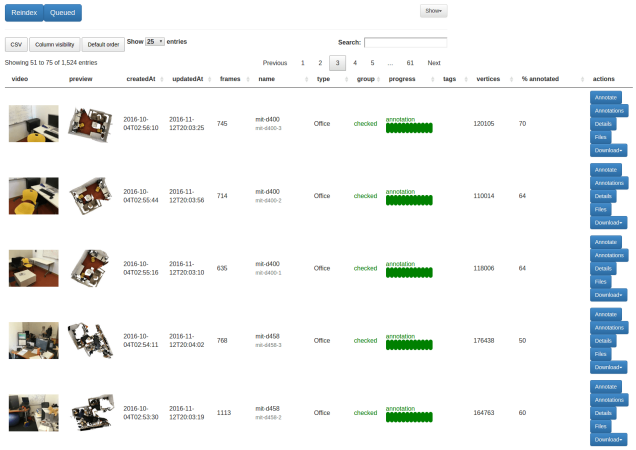

Figure 14. Our web-based data management UI for ScanNet scan data.

are visible and usable for sorting and filtering of the scan database.

References

[1] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 10, 11

[2] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565. IEEE, 2015. 1

[3] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016. 1

[4] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt. BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration. *arXiv preprint arXiv:1604.01093*, 2016. 9

[5] M. Di Cicco, L. Iocchi, and G. Grisetti. Non-parametric calibration for depth sensors. *Robotics and Autonomous Systems*, 74:309–317, 2015. 9

[6] C. Fellbaum. *WordNet*. Wiley Online Library, 1998. 1, 10

[7] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung. SceneNN: A scene meshes dataset with annotations. In *International Conference on 3D Vision (3DV)*, volume 1, 2016. 1, 4, 5

[8] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 1, 10

[9] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32(6):169, 2013. 1

[10] M. Savva, A. X. Chang, P. Hanrahan, M. Fisher, and M. Nießner. PiGraphs: Learning interaction snapshots from observations. *ACM Transactions on Graphics (TOG)*, 35(4), 2016. 1, 4

[11] A. Teichman, S. Miller, and S. Thrun. Unsupervised intrinsic calibration of depth sensors via SLAM. In *Robotics: Science and Systems*, volume 248, 2013. 9

[12] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 10