ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes

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

www.scan-net.org

Abstract

A key requirement for leveraging supervised deep learning methods is the availability of large, labeled datasets. Unfortunately, in the context of RGB-D scene understanding, very little data is available – current datasets cover a small range of scene views and have limited semantic annotations. To address this issue, we introduce ScanNet, an RGB-D video dataset containing 2.5M views in 1513 scenes annotated with 3D camera poses, surface reconstructions, and semantic segmentations. To collect this data, we designed an easy-to-use and scalable RGB-D capture system that includes automated surface reconstruction and crowdsourced semantic annotation.We show that using this data helps achieve state-of-the-art performance on several 3D scene understanding tasks, including 3D object classification, semantic voxel labeling, and CAD model retrieval. The dataset is freely available at http://www.scan-net.org.

1. Introduction

Since the introduction of commodity RGB-D sensors, such as the Microsoft Kinect, the field of 3D geometry capture has gained significant attention and opened up a wide range of new applications. Although there has been significant effort on 3D reconstruction algorithms, general 3D scene understanding with RGB-D data has only very recently started to become popular. Research along semantic understanding is also heavily facilitated by the rapid progress of modern machine learning methods, such as neural models. One key to successfully applying theses approaches is the availability of large, labeled datasets. While much effort has been made on 2D datasets [17, 42, 45], where images can be downloaded from the web and directly annotated, the situation for 3D data is more challenging. Thus, many of the current RGB-D datasets [71, 89, 74, 30] are orders of magnitude smaller than their 2D counterparts. Typically, 3D deep learning methods use synthetic data to mitigate this lack of real-world data [88, 6].

Figure 1. Example reconstructed spaces in ScanNet annotated with instance-level object category labels through our crowdsourced annotation framework.

One of the reasons that current 3D datasets are small is because their capture requires much more effort, and efficiently providing (dense) annotations in 3D is non-trivial. Thus, existing work on 3D datasets often fall back to polygon or bounding box annotations on 2.5D RGB-D images [71, 89, 74], rather than directly annotating in 3D. In the latter case, labels are added manually by expert users (typically by the paper authors) [30, 68] which limits their overall size and scalability.

In this paper, we introduce *ScanNet*, a dataset of richly-annotated RGB-D scans of real-world environments containing 2.5M RGB-D images in 1513 scans acquired in 707 distinct spaces. The sheer magnitude of this dataset is larger than any other [55, 78, 89, 72, 3, 68, 30]. However, what makes it particularly valuable for research in scene understanding is its annotation with estimated calibration parameters, camera poses, 3D surface reconstructions, textured meshes, dense object-level semantic segmentations, and aligned CAD models (see Fig. 2). The semantic segmentations are more than an order of magnitude larger than any previous RGB-D dataset.

Dataset	Size	Labels	Annotation Tool	Reconstruction	CAD Models
NYU v2 [55]	464 scans	1449 frames	2D LabelMe-style [66]	none	some [25]
TUM [78]	47 scans	none	-	aligned poses (Vicon)	no
SUN 3D [89]	415 scans	8 scans	2D polygons	aligned poses [89]	no
SUN RGB-D [72]	10k frames	10k frames	2D polygons + bounding boxes	aligned poses [89]	no
BuildingParser [3]	265 rooms	265 rooms	CloudCompare [24]	point cloud	no
PiGraphs [68]	26 scans	26 scans	dense 3D, by the authors [68]	dense 3D [59]	no
SceneNN [30]	100 scans	100 scans	dense 3D, by the authors [57]	dense 3D [9]	no
ScanNet (ours)	**1513 scans 2.5M frames**	**1513 scans**	**dense 3D, crowd-sourced MTurk labels also proj. to 2D frames**	**dense 3D [12]**	**yes**

Table 1. Overview of RGB-D datasets for 3D reconstruction and semantic scene understanding. Note that in addition to the 1513 scans in ScanNet, we also provided dense 3D reconstruction and annotations on all NYU v2 sequences.

In the collection of this dataset, we have considered two main research questions: 1) how can we design a framework that allows many people to collect and annotate large amounts of RGB-D data, and 2) can we use the rich annotations and data quantity provided in ScanNet to learn better 3D models for scene understanding?

To investigate the first question, we built a capture pipeline to help novices acquire semantically-labeled 3D models of scenes. A person uses an app on an iPad mounted with a depth camera to acquire RGB-D video, and then we processes the data off-line and return a complete semantically-labeled 3D reconstruction of the scene. The challenges in developing such a framework are numerous, including how to perform 3D surface reconstruction robustly in a scalable pipeline and how to crowdsource semantic labeling. The paper discusses our study of these issues and documents our experience with scaling up RGB-D scan collection (20 people) and annotation (500 crowd workers).

To investigate the second question, we trained 3D deep networks with the data provided by ScanNet and tested their performance on several scene understanding tasks, including 3D object classification, semantic voxel labeling, and CAD model retrieval. For the semantic voxel labeling task, we introduce a new volumetric CNN architecture.

Overall, the contributions of this paper are:

- A large 3D dataset containing 1513 RGB-D scans of over 707 unique indoor environments with estimated camera parameters, surface reconstructions, textured meshes, semantic segmentations. We also provide CAD model placements for a subset of the scans.
- A design for efficient 3D data capture and annotation suitable for novice users.
- New RGB-D benchmarks and improved results for state-of-the art machine learning methods on 3D object classification, semantic voxel labeling, and CAD model retrieval.
- A complete open source acquisition and annotation framework for dense RGB-D reconstructions.

2. Previous Work

A large number of RGB-D datasets have been captured and made publicly available for training and benchmarking [53, 32, 48, 62, 76, 80, 71, 4, 55, 78, 15, 52, 1, 65, 29, 49, 21, 46, 41, 89, 77, 58, 69, 90, 34, 16, 33, 54, 38, 28, 67, 50, 43, 92, 72, 9, 31, 82, 68, 30, 3, 10, 75, 2].[1] These datasets have been used to train models for many 3D scene understanding tasks, including semantic segmentation [64, 55, 26, 83], 3D object detection [70, 44, 27, 73, 74], 3D object classification [88, 51, 63], and others [91, 22, 23].

Most RGB-D datasets contain scans of individual objects. For example, the Redwood dataset [10] contains over 10,000 scans of objects annotated with class labels, 1,781 of which are reconstructed with KinectFusion [56]. Since the objects are scanned in isolation without scene context, the dataset's focus is mainly on evaluating surface reconstruction quality rather than semantic understanding of complete scenes.

One of the earliest and most popular datasets for RGB-D scene understanding is NYU v2 [71]. It is composed of 464 short RGB-D sequences, from which 1449 frames have been annotated with 2D polygons denoting semantic segmentations, as in LabelMe [66]. SUN RGB-D [72] follows up on this work by collecting 10,335 RGB-D frames annotated with polygons in 2D and bounding boxes in 3D. These datasets have scene diversity comparable to ours, but include only a limited range of viewpoints, and do not provide complete 3D surface reconstructions, dense 3D semantic segmentations, or a large set of CAD model alignments.

One of the first RGB-D datasets focused on long RGB-D sequences in indoor environments is SUN3D. It contains a set of 415 Kinect v1 sequences of 254 unique spaces. Although some objects were annotated manually with 2D polygons, and 8 scans have estimated camera poses based

[1] A comprehensive and detailed overview of publicly-accessible RGB-D datasets is given by [20] at http://www0.cs.ucl.ac.uk/staff/M.Firman/RGBDdatasets/, which is updated on a regular basis.

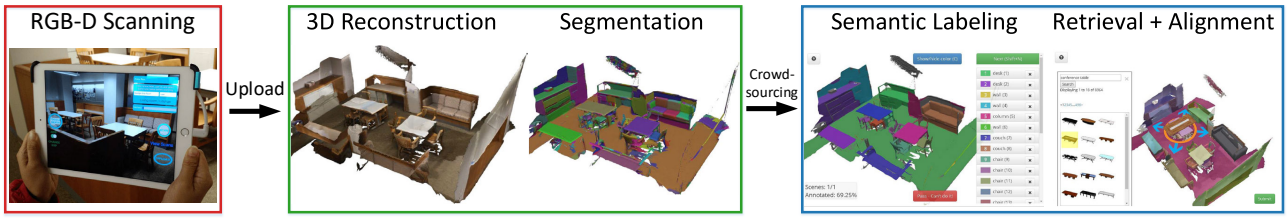

Figure 2. Overview of our RGB-D reconstruction and semantic annotation framework. **Left:** a novice user uses a handheld RGB-D device with our scanning interface to scan an environment. **Mid:** RGB-D sequences are uploaded to a processing server which produces 3D surface mesh reconstructions and their surface segmentations. **Right:** Semantic annotation tasks are issued for crowdsourcing to obtain instance-level object category annotations and 3D CAD model alignments to the reconstruction.

on user input, the bulk of the dataset does not include camera poses, 3D reconstructions, or semantic annotations.

Recently, Armeni et al. [3, 2] introduced an indoor dataset containing 3D meshes for 265 rooms captured with a custom Matterport camera and manually labeled with semantic annotations. The dataset is high-quality, but the capture pipeline is based on expensive and less portable hardware. Furthermore, only a fused point cloud is provided as output. Due to the lack of raw color and depth data, its applicability to research on reconstruction and scene understanding from raw RGB-D input is limited.

The datasets most similar to ours are SceneNN [30] and PiGraphs [68], which are composed of 100 and 26 densely reconstructed and labeled scenes respectively. The annotations are done directly in 3D [57, 68]. However, both scanning and labeling are performed only by expert users (i.e. the authors), limiting the scalability of the system and the size of the dataset. In contrast, we design our RGB-D acquisition framework specifically for ease-of-use by untrained users and for scalable processing through crowdsourcing. This allows us to acquire a significantly larger dataset with more annotations (currently, 1513 sequences are reconstructed and labeled).

3. Dataset Acquisition Framework

In this section, we focus on the design of the framework used to acquire the ScanNet dataset (Fig. 2). We discuss design trade-offs in building the framework and relay findings on which methods were found to work best for large-scale RGB-D data collection and processing.

Our main goal driving the design of our framework was to allow untrained users to capture semantically labeled surfaces of indoor scenes with commodity hardware. This design goal has broad implications on the choices made at every step of the process. The RGB-D scanning system must be trivial to use, the data processing must be robust and automatic, the semantic annotations must be crowdsourced, and the flow of data through the system must be handled by a tracking server. We discuss our approach to each of these issues in the following subsections.

3.1. RGB-D Scanning

Hardware. There is a spectrum of choices for RGB-D sensor hardware. Our requirement for deployment to large groups of inexperienced users necessitates a portable and low-cost RGB-D sensor setup. We use the Structure sensor [60], a commodity RGB-D sensor with design similar to the Microsoft Kinect v1. We attach this sensor to a handheld device such as an iPhone or iPad (see Fig. 2 left) — results in this paper were collected using iPad Air2 devices. The iPad RGB camera data is temporally synchronized with the depth sensor via hardware, thus providing synchronized depth and color capture at 30 Hz. Depth frames are captured at a resolution of 640×480 and color at 1296×968 pixels. We enable auto-white balance and auto-exposure by default. The sensor unit is calibrated once by the user to ensure accurate, registered color and depth data, as described in the following section.

Calibration. Our use of commodity RGB-D sensors necessitates unwarping of depth data and alignment of depth and color data. Prior work has dealt with camera calibration in controlled lab conditions where more accurate equipment such as LIDAR scanners can be used to obtain undistortion data for commodity RGB-D sensors (e.g., Wang et al. [84]). However, for our procedure to be practical for novice users it needs to be based only on the portable commodity sensor setup. Therefore, the user only needs to print a checkerboard pattern on paper, place it on a large and flat surface, and capture an RGB-D sequence viewing the flat surface from close by to far away. This sequence, as well as a set of infrared and color frame pairs viewing the checkerboard from a variety of views, are uploaded by the user as input to the calibration. Our system then runs a calibration procedure based on [81, 14] to obtain intrinsic parameters for both depth and color sensors, and an extrinsic transformation of depth to color. This process computes a depth undistortion lookup table $f(x, y, d)$ which provides a depth correction for each image x, y coordinate at measured depth d. We find that this calibration procedure is easy for users and results in improved data and consequently enhanced reconstruction quality (see Fig. 3).

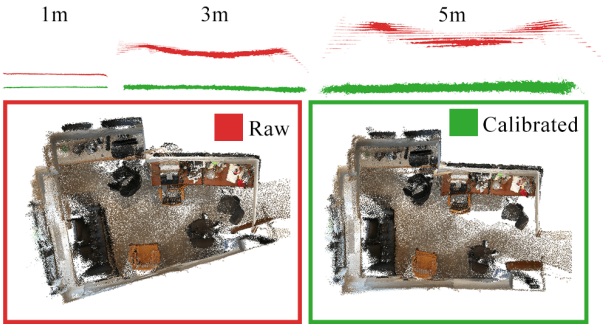

Figure 3. Comparison of calibration results. In the top row, we show results of calibration on a flat wall. As the distance increases the distortion becomes quite severe, motivating the need for depth distortion calibration. In the bottom row, we show results of frame-to-frame tracking on raw and calibrated data.

User Interface. To make the capture process simple for untrained users, we designed an iOS app with a simple live RGB-D video capture UI (see Fig. 2 left). The user provides a name and scene type for the current scan and proceeds to record a sequence. During scanning, a log-scale RGB feature detector point metric is shown as a "featurefulness" bar to provide a rough measure of tracking robustness and reconstruction quality in different regions being scanned. This feature was critical for providing intuition to users who are not familiar with the constraints and limitations of 3D reconstruction algorithms.

Storage. We store scans as compressed RGB-D data on the device flash memory so that a stable internet connection is not required during scanning. The user can upload scans to the processing server when convenient by pressing an "upload" button. Our sensor units used 128 GB iPad Air2 devices which allowed for several hours of recorded RGB-D video. In practice, the bottleneck was battery life rather than storage space. Depth is recorded as 16-bit unsigned short values and stored using standard zLib compression. RGB data is encoded with the H.264 codec with a high bitrate of 15 Mbps to prevent encoding artifacts. In addition to the RGB-D frames, we also record Inertial Measurement Unit (IMU) data, including acceleration, and angular velocities, using the Apple SDK. Timestamps are recorded for IMU, color, and depth images.

3.2. Surface Reconstruction

Once data has been uploaded from the iPad to our server, the first processing step is to estimate a densely-reconstructed 3D surface mesh and 6-DoF camera poses for all RGB-D frames. To conform with the goal for an automated and scalable framework, we choose methods that favor robustness and processing speed such that uploaded recordings can be processed at near real-time rates with little supervision.

Dense Reconstruction. We use volumetric fusion [11] to perform the dense reconstruction, since this approach is widely used in the context of commodity RGB-D data. There is a large variety of algorithms targeting this scenario [56, 85, 7, 59, 35, 86, 40, 9, 87, 36, 12]. We chose the BundleFusion system [12] as it was designed and evaluated for similar sensor setups as ours, and provides real-time speed while being reasonably robust given handheld RGB-D video data.

For each input scan, we first run BundleFusion [12] at a voxel resolution of $1\,\mathrm{cm}^3$. BundleFusion produces accurate pose alignments which we then use to perform volumetric integration through VoxelHashing [59] and extract a high resolution surface mesh using the Marching Cubes algorithm on the implicit TSDF ($4\,\mathrm{mm}^3$ voxels). The mesh is then automatically cleaned up with a set of filtering steps to merge close vertices, delete duplicate and isolated mesh parts, and finally to downsample the mesh to high, medium, and low resolution versions (each level reducing the number of faces by a factor of two).

Orientation. After the surface mesh is extracted, we automatically align it and all camera poses to a common coordinate frame with the z-axis set to the up vector, and the xy plane aligned with the floor plane. To perform this alignment, we first extract all planar regions of sufficient size, merge regions defined by the same plane, and sort them by normal (we use a normal threshold of 25° and a planar offset threshold of $5\,\mathrm{cm}$). We then determine a prior for the up vector by projecting the IMU gravity vectors for all frames into the coordinates of the first frame. This allows us to select the floor plane based on the scan bounding box and the normal that is most similar to the IMU up vector direction. Finally, we use a PCA on the mesh vertices to determine the rotation around the z-axis and translate the scan such that its bounds are within the positive octant of the coordinate system.

Validation. This reconstruction process is automatically triggered when a scan is uploaded to the processing server and runs unsupervised. In order to establish a clean snapshot to construct the ScanNet dataset reported in this paper, we automatically discard scan sequences that are short, have high residual reconstruction error, or have low percentage of aligned frames. We then manually check for and discard reconstructions with noticeable misalignments.

3.3. Semantic Annotation

After a reconstruction is produced by the processing server, annotation HITs (Human Intelligence Tasks) are issued on the Amazon Mechanical Turk crowdsourcing market. The two HITs that we crowdsource are: i) instance-

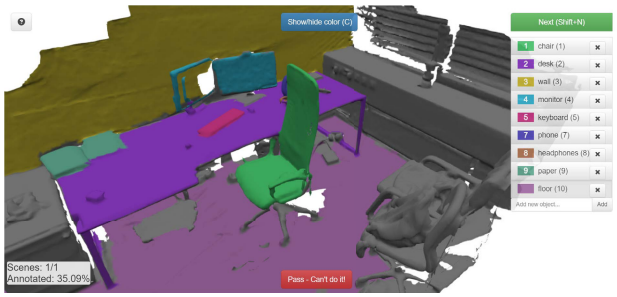

Figure 4. Our web-based crowdsourcing interface for annotating a scene with instance-level object category labels. The right panel lists object instances already annotated in the scene with matching painted colors. This annotation is in progress at around $\approx 35\%$ with gray regions indicating unannotated surfaces.

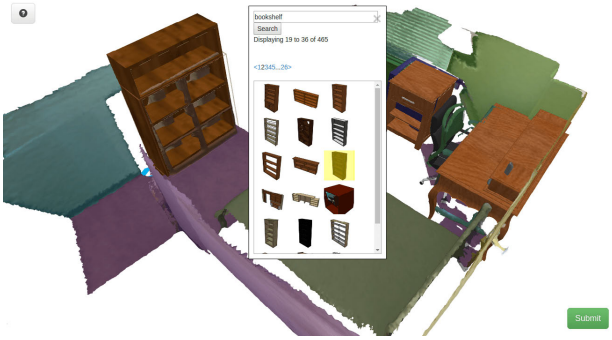

Figure 5. Crowdsourcing interface for aligning CAD models to objects in a reconstruction. Objects can be clicked to initiate an assisted search for CAD models (see list of bookshelves in middle). A suggested model is placed at the position of the clicked object, and the user then refines the position and orientation. A desk, chair, and nightstand have been already placed here.

level object category labeling of all surfaces in the reconstruction, and ii) 3D CAD model alignment to the reconstruction. These annotations are crowdsourced using web-based interfaces to again maintain the overall scalability of the framework.

Instance-level Semantic Labeling. Our first annotation step is to obtain a set of object instance-level labels directly on each reconstructed 3D surface mesh. This is in contrast to much prior work that uses 2D polygon annotations on RGB or RGB-D images, or 3D bounding box annotations.

We developed a WebGL interface that takes as input the low-resolution surface mesh of a given reconstruction and a conservative over-segmentation of the mesh using a normal-based graph cut method [19, 37]. The crowd worker then selects segments to annotate with instance-level object category labels (see Fig. 4). Each worker is required to annotate at least 25% of the surfaces in a reconstruction, and encouraged to annotate more than 50% before submission. Each scan is annotated by multiple workers (scans in ScanNet are annotated by 2.3 workers on average).

A key challenge in designing this interface is to allow for efficient annotation by workers who have no prior experience with the task, or 3D interfaces in general. Our interface uses a simple painting metaphor where clicking and dragging over surfaces paints segments with a given label and corresponding color. This functions similarly to 2D painting and allows for erasing and modifying existing regions.

Another design requirement is to allow users to provide freeform text labels to reduce the inherent bias and scalability issues of pre-selected label lists. At the same time, it is desirable to guide users for consistency and coverage of basic object types. To achieve this consistency and improve labeling efficiency, the interface provides autocomplete functionality over all labels previously provided by other workers that pass a frequency threshold (more than five annotations). Workers are always allowed to add arbitrary text labels to ensure coverage and allow expansion of the label set.

Several additional design details are important to ensure usability by novice workers. First, a simple distance check for connectedness is used to disallow labeling of disconnected surfaces with the same label. Earlier experiments without this constraint resulted in two undesirable behaviors: cheating by painting many surfaces with a few labels, and labeling of multiple object instances with the same label. Second, 3D camera control is challenging for novice users, and the 3D nature of the data is hard to illustrate from one view. Therefore, we first show a full turntable rotation of each reconstruction and then instruct workers to change the view using a rotating turntable metaphor. Without the turntable rotation animation, many workers only annotated from the initial view and never used camera controls despite the provided instructions.

CAD Model Retrieval and Alignment. In the second annotation task, a crowd worker was given a reconstruction already annotated with object instances and asked to place appropriate 3D CAD models to represent major objects in the scene. The challenge of this task lies in the selection of closely matching 3D models from a large database, and in precisely aligning each model to the 3D position of the corresponding object in the reconstruction.

We implemented an assisted object retrieval interface where clicking on a previously labeled (and thus painted) object in a reconstruction immediately searched for CAD models with the same category label in the ShapeNetCore [6] dataset, and placed one example model such that it overlaps with the oriented bounding box of the clicked object (see Fig. 5). The worker then used keyboard and mouse-based controls to adjust the alignment of the model, and was allowed to submit the task once at least three CAD

Statistic	SceneNN [30]	ScanNet
# of scans	100	1513
# of RGB-D frames	2,475,905	2,492,518
floor area (avg / sum m^2)	22.6 / 2,124	22.6 / 34,453
surface area (avg / sum m^2)	75.3 / 7,078	51.6 / 78,595
labeled objects (avg / sum)	15.8 / 1482	24.1 / 36,213

Table 2. Summary statistics for ScanNet compared to the most similar existing dataset (SceneNN [30]). ScanNet has an order of magnitude more scans, with 3D surface mesh reconstructions covering more than ten times the floor and surface area, and with more than 36,000 annotated object instances.

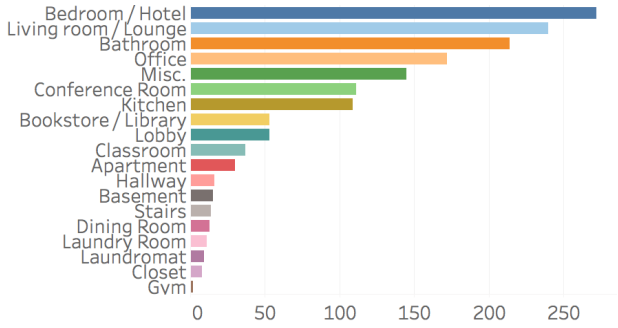

Figure 6. Distribution of the scans in ScanNet organized by type.

models were placed.

Using this interface, we collected sets of CAD models aligned to each ScanNet reconstruction. Preliminary results indicate that despite the challenging nature of this task, workers select semantically appropriate CAD models to match objects in the reconstructions. The main limitation of this interface is due to the mismatch between the corpus of available CAD models and the objects observed in the ScanNet scans. Despite the diversity of the ShapeNet CAD model dataset (55K objects), it is still hard to find exact instance-level matches for chairs, desks and more rare object categories. A promising way to alleviate this limitation is to algorithmically suggest candidate retrieved and aligned CAD models such that workers can perform an easier verification and adjustment task.

4. ScanNet Dataset

In this section, we summarize the data we collected using our framework to establish the ScanNet dataset. This dataset is a snapshot of available data which was achieved in roughly one month of data acquisition by twenty users at locations in several countries. It has annotations by more than 500 crowd workers on the Mechanical Turk platform. Since the presented framework runs in an unsupervised fashion and people are continuously collecting data, this dataset continues to grow organically. Here, we report some statistics for an initial snapshot of 1513 scans, which are summarized in Table 2.

Fig. 6 plots the distribution of scanned scenes over different types of real-world spaces. ScanNet contains a variety of spaces such as offices, apartments, and bathrooms. The dataset contains a diverse set of spaces ranging from small (e.g., bathrooms, closets, utility rooms) to large (e.g., apartments, classrooms, and libraries). Each of the scans has been annotated with instance-level semantic category labels through our crowdsourcing task. In total, we deployed 3,391 annotation tasks to annotate all 1513 scans.

All the text labels used by crowd workers to annotate object instances are mapped to the object category sets of NYU v2 [55], ModelNet [88], ShapeNet [6], and corresponding WordNet [18] synsets. This mapping is made more robust by a preprocess that collapses the initial text labels through synonym and misspelling detection. Each labeled object instance has a corresponding oriented bounding box computed as part of the annotation process. We use these bounding boxes to detect outlier annotations for manual verification.

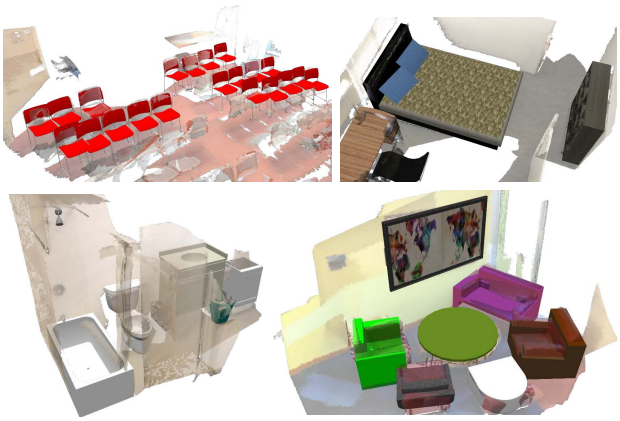

Figure 7. ShapeNetCore [6] CAD models retrieved and placed on ScanNet scans by crowd workers (scan mesh is transparent and CAD models are opaque). From top left clockwise: a classroom, bedroom, bathroom, and lounge scan.

In addition to reconstructing and annotating the 1513 ScanNet scans, we have processed all the NYU v2 RGB-D sequences with our framework. The result is a set of dense reconstructions of the NYU v2 spaces with instance-level object annotations in 3D that are complementary in nature to the existing image-based annotations.

We also deployed the CAD model alignment crowdsourcing task to collect a total of 107 virtual scene interpretations consisting of aligned ShapeNet models placed on a subset of 52 ScanNet scans by 106 workers. See Fig. 7 for several examples visualized by overlaying the CAD models on the 3D surface mesh reconstruction. There were a total of 681 CAD model instances (of 296 unique models) retrieved and placed on the reconstructions, with an average of 6.4 CAD model instances per annotated scan.

		Scans		Instances	
		#Train	#Test	#Train	#Test
Object Classification	ScanNet	1205	312	9305	2606
	NYU	452	80	3260	613
	SceneNN	70	12	377	66
Semantic Voxel Labeling	ScanNet	1205	312	80554	21300

Table 3. Train/Test split for object classification and dense voxel prediction tasks. Note that the number of instances does not include the rotation augmentation.

For more detailed statistics on this first ScanNet dataset snapshot, please see the appendix.

5. Tasks and Benchmarks

In this section, we describe the three tasks that we developed as benchmarks for demonstrating the value of ScanNet data.

Train/Test split statistics. Table 3 shows the test and training splits of ScanNet in the context of the object classification and dense voxel prediction benchmarks. Note that our data is significantly larger than any existing comparable dataset. We use these tasks to demonstrate that ScanNet enables the use of deep learning methods for 3D scene understanding tasks with supervised training, and compare performance to that using data from other existing datasets.

5.1. 3D Object Classification

With the availability of large-scale synthetic 3D datasets such as [88, 6] and recent advances in 3D deep learning, research has developed approaches to classify objects using only geometric data with volumetric deep nets [88, 79, 50, 13, 63]. All of these methods train on purely synthetic data and focus on isolated objects. Although they show limited evaluation on real-world data, a larger evaluation on realistic scanning data is largely missing. When training data is synthetic and test is performed on real data, there is also a significant discrepancy of test performance, as data characteristics, such as noise and occlusions patterns, are inherently different.

With ScanNet, we close this gap as we have captured a sufficiently large amount of 3D data to use real-world RGB-D input for *both* training and test sets. For this task, we use the bounding boxes around annotated objects in ScanNet, and isolate the contained geometry. As a result, we obtain local volumes around each object instance for which we know the annotated category. The goal of the task is to classify the object represented by a set of scanned points within a given bounding box. For this benchmark, we use 17 categories, for which we use $9,677$ instances in the training set, and $2,606$ instances in the test set.

Network and training. For object classification, we follow the network architecture of the 3D Network-in-Network of [63], without the multi-orientation pooling step. In order to classify partial data, we add a second channel to the 30^3 occupancy grid input, indicating known and unknown regions (with 1 and 0, respectively) according to the camera scanning trajectory. As in Qi et al. [63], we use an SGD solver with learning rate 0.01 and momentum 0.9, decaying the learning rate by half every 20 epochs, and training the model for 200 epochs. We augment training samples with 12 instances of different rotations (including both elevation and tilt), resulting in a total training set of $111,660$ samples.

Benchmark performance. As a baseline evaluation, we run the 3D CNN approach of Qi et al. [63]. Table 4 shows the performance of 3D shape classification with different train and test sets. The first two columns show results on synthetic test data from ShapeNet [6] including both complete and partial data. Naturally, training with the corresponding synthetic counterparts of ShapeNet provides the best performance, as data characteristics are shared. However, the more interesting case is testing on real-world data (last two columns on the right); here, we show results on the test sets of SceneNN [30] and ScanNet. First, we see that training on synthetic data allows only for limited knowledge transfer (first two rows). Second, although the relatively small SceneNN dataset is able to learn within its own dataset to a reasonable degree, it does not generalize to the larger variety of environments found in ScanNet. On the other hand, training on ScanNet translates well to testing on SceneNN; as a result, the test results on SceneNN are significantly improved by using the training data from ScanNet. Interestingly enough, these results can be slightly improved when mixing training data of ScanNet with partial scans of ShapeNet (last row).

	Synthetic Test Sets		Real Test Sets	
Training Set	ShapeNet	ShapeNet Partial	SceneNN	ScanNet
ShapeNet	**92.5**	37.6	68.2	39.5
ShapeNet Partial	88.5	**92.1**	72.7	45.7
SceneNN	19.9	27.7	69.8	48.2
NYU	26.2	26.6	72.7	53.2
ScanNet	21.4	31.0	78.8	74.9
ScanNet +ShapeNet Par.	79.7	89.8	**81.2**	**76.6**

Table 4. 3D object classification benchmark performance. Percentages give the classification accuracy over all models in each test set (average instance accuracy).

5.2. Semantic Voxel Labeling

A common task on RGB data is semantic segmentation (i.e. labeling pixels with semantic classes) [47]. With our data, we can extend this task to 3D, where the goal is to

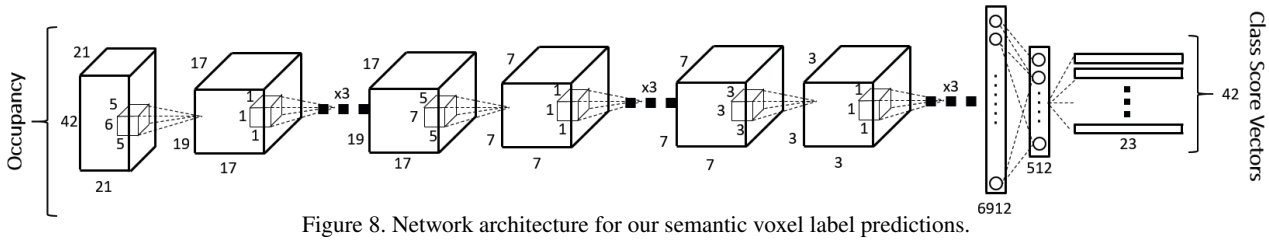

Figure 8. Network architecture for our semantic voxel label predictions.

predict the semantic object label on a per-voxel basis. This task of predicting a semantic class for each visible 3D voxel has been addressed by some prior work, but using hand-crafted features to predict a small number of classes [39, 83], or focusing on outdoor environments [8, 5].

Data Generation. We first voxelize a scene and obtain a dense voxel grid with a voxel size of 2cm^3, where every object stores the underlying object class annotation (empty space and unlabeled surface points have their own respective class). We now extract subvolumes of the scene volume, where each of these volumes has a dimension of $21 \times 21 \times 42$ and a spatial extent of $1\text{m} \times 1\text{m} \times 2\text{m}$; i.e., as voxel size of approx. 4.8cm^3. These sample volumes are always aligned with the xy-ground plane, and they only differ in their xy-coordinates (the subvolumes are centered around xy samples). For ground truth data generation, voxel labels are propagated from the scene voxelization to these sample volumes. The samples are chosen that at least 2% of the voxels are occupied (i.e., on the surface), and among the voxels on the surface at least 80% are required to have valid annotations; samples not meeting these criteria are discarded. Across ScanNet, we generate $80,554$ subvolume examples for training which are augmented by 8 rotations each (i.e., we have $644,432$ training samples) from 1205 training scenes. In addition, we extract $21,300$ sample volumes for the test data set which are also augmented by 8 rotations each (i.e., a total of $170,400$ test samples) from 312 test scenes (test and training scenes are different). We have 20 object class labels plus 1 class for free space.

Network and training. For the semantic voxel labeling task, we propose a network which predicts class labels for a column of voxels in a scene according to the occupancy characteristics of the voxels' neighborhood. In order to infer labels for an entire scene, we use the network to predict a label for every voxel column at test time (i.e., every xy position that has voxels on the surface). The network takes as input a $21 \times 21 \times 42$ volume and uses a series of fully convolutional layers to simultaneously predict class scores for the center column of 42 voxels (see Fig. 8). We use ReLU and batch normalization for all layers (except the last) in the network. We augment training samples with 8 instances of different rotations. Since we randomly sample real-world scenes for the training and test data, the training data is unbalanced (particularly with regard to floors and walls), so we weight the cross entropy measure with the inverse log of the histogram of the train data.

We use an SGD solver with learning rate 0.005 and momentum 0.9, decaying the learning rate by half every 20 epochs, and training the model for 100 epochs.

Quantitative Results. The goal of this task is to predict semantic labels for all visible surface voxels in a given 3D scene; i.e., every voxel on a visible surface receives one of the 20 object class labels. We use NYU2 labels, and list voxel classification results on ScanNet in Table 5. Sample scans with voxel label predictions are visualized in Fig. 9. We achieve an voxel classification accuracy of 78.9%, which is based purely on the geometric input (no color is used).

Note that we do not explicitly enforce consistency predictions between neighboring voxel columns when the *test volume* is slid across the xy plane. This could be enforced with a volumetric CRF [61] which is used in [83]; however, our goal in this task to focus exclusively on the per-voxel classification accuracy.

5.3. 3D Object Retrieval

Another important task is retrieval of similar CAD models given (potentially partial) RGB-D scans. To this end, one wants to learn a shape embedding where a feature descriptor defines geometric similarity between shapes. The core idea is to train a network on a shape classification task where a shape embedding can be learned as *byproduct* of the classification task. For instance, Wu et al. [88] and Qi et al. [63] use this technique to perform shape retrieval queries within the ShapeNet database.

With ScanNet, we have established category-level correspondences between real-world objects and ShapeNet models. This allows us to train on a classification problem where both real and synthetic data are mixed inside of each category using real and synthetic data within shared class labels. Thus, we can learn an embedding between real and synthetic data in order to perform model retrieval for RGB-D scans. To this end, we use the volumetric shape classification network by Qi et al. [63], we use the same training

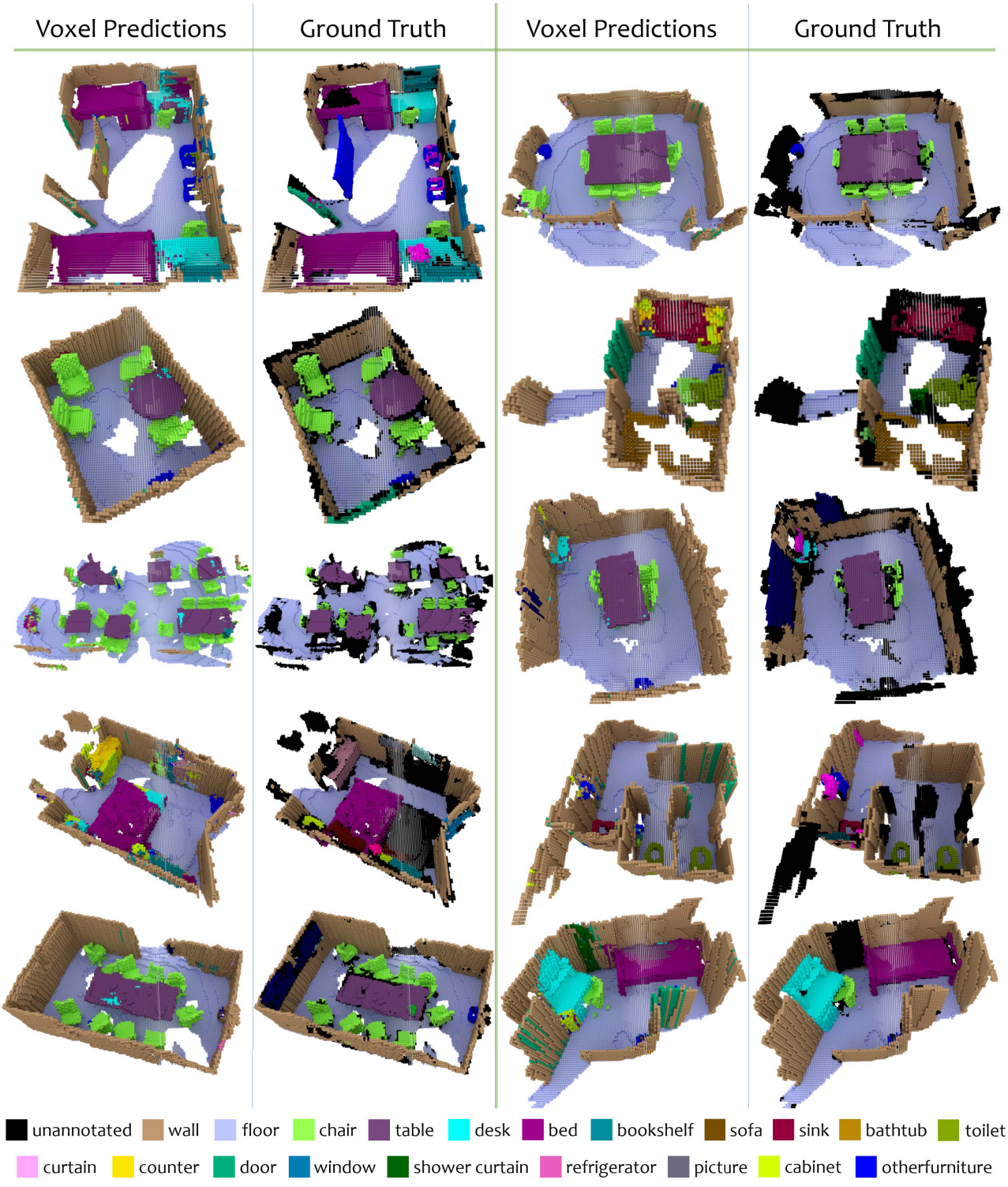

Figure 9. Semantic voxel labeling of 3D scans in ScanNet using our 3D CNN architecture. Voxel colors indicate predicted or ground truth category.

Class	% of Test Samples	Accuracy
Wall	40.1%	90%
Floor	27.7%	97%
Chair	4.9%	73%
Sofa	2.9%	71%
Table	2.5%	58%
Door	3.0%	25%
Cabinet	3.0%	51%
Bed	2.5%	59%
Desk	1.8%	40%
Toilet	0.7%	74%
Sink	0.6%	59%
Window	0.8%	16%
Picture	0.2%	7%
Bookshelf	2.8%	55%
Curtain	1.0%	13%
Shower Curtain	0.2%	63%
Counter	0.7%	9%
Refrigerator	0.5%	35%
Bathtub	0.9%	84%
OtherFurniture	3.0%	21%
Total	-	78.9%

Table 5. Semantic voxel label prediction accuracy.

	Retrieval from ShapeNet	
Train	Top 1 NN	Top 3 NNs
ShapeNet	10.4%	8.0%
ScanNet	12.7%	11.7%
ShapeNet + ScanNet	**77.5%**	**77.0%**

Table 6. 3D model retrieval benchmark performance. Nearest neighbor models are retrieved for ScanNet objects from the ShapeNetCore database. Percentages indicate average instance accuracy of retrieved model to query region.

procedure as in Sec. 5.1. Nearest neighbors are retrieved based on the ℓ_2 distance between the extracted feature descriptors, and measured against the ground truth provided by the CAD model retrieval task. In Table 6, we show object retrieval results using objects from ScanNet to query for nearest neighbor models from ShapeNetCore. Note that training on ShapeNet and ScanNet independently results in poor retrieval performance, as neither are able to bridge the gap between the differing characteristics of synthetic and real-world data. Training on both ShapeNet and ScanNet together is able to find an embedding of shape similarities between both data modalities, resulting in much higher retrieval accuracy.

6. Conclusion

This paper introduces ScanNet: a large-scale RGB-D dataset of 1513 scans with surface reconstructions, instance-level object category annotations, and 3D CAD model placements. To make the collection of this data possible, we designed a scalable RGB-D acquisition and semantic annotation framework that we provide for the benefit of the community. We demonstrated that the richly-annotated scan data collected so far in ScanNet is useful in achieving state-of-the-art performance on several 3D scene understanding tasks; we hope that ScanNet will inspire future work on many other tasks.

Acknowledgments

This project is funded by Google Tango, Intel, NSF (IIS-1251217 and VEC 1539014/1539099), and a Stanford Graduate fellowship. We also thank Occipital for donating structure sensors and Nvidia for hardware donations, as well as support by the Max-Planck Center for Visual Computing and the Stanford CURIS program. Further, we thank Toan Vuong, Joseph Chang, and Helen Jiang for help on the mobile scanning app and the scanning process, and Hope Casey-Allen and Duc Nugyen for early prototypes of the annotation interfaces. Last but not least, we would like to thank all people for help with scanning and getting us access to scanning spaces.

References

[1] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze. A global hypotheses verification method for 3D object recognition. In *European Conference on Computer Vision*, pages 511–524. Springer, 2012. 2

[2] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 2, 3

[3] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3D semantic parsing of large-scale indoor spaces. CVPR, 2016. 1, 2, 3

[4] I. B. Barbosa, M. Cristani, A. Del Bue, L. Bazzani, and V. Murino. Re-identification with RGB-D sensors. In *European Conference on Computer Vision*, pages 433–442. Springer, 2012. 2

[5] M. Blaha, C. Vogel, A. Richard, J. D. Wegner, T. Pock, and K. Schindler. Large-scale semantic 3d reconstruction: an adaptive multi-resolution model for multi-class volumetric labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3176–3184, 2016. 8

[6] 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. 1, 5, 6, 7, 21

[7] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(4):113, 2013. 4

[8] I. Cherabier, C. Häne, M. R. Oswald, and M. Pollefeys. Multi-label semantic 3d reconstruction using voxel blocks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 601–610. IEEE, 2016. 8

[9] 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. 2, 4, 16

[10] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016. 2, 16

[11] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996. 4

[12] 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. 2, 4, 20

[13] A. Dai, C. R. Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. *arXiv preprint arXiv:1612.00101*, 2016. 7

[14] M. Di Cicco, L. Iocchi, and G. Grisetti. Non-parametric calibration for depth sensors. *Robotics and Autonomous Systems*, 74:309–317, 2015. 3, 20

[15] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691–1696. IEEE, 2012. 2

[16] N. Erdogmus and S. Marcel. Spoofing in 2D face recognition with 3D masks and anti-spoofing with Kinect. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–6. IEEE, 2013. 2

[17] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 1

[18] C. Fellbaum. *WordNet*. Wiley Online Library, 1998. 6, 13, 21

[19] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. 5

[20] M. Firman. RGBD datasets: Past, present and future. In *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*, 2016. 2

[21] S. Fothergill, H. Mentis, P. Kohli, and S. Nowozin. Instructing people for training gestural interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1737–1746. ACM, 2012. 2

[22] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3392–3399, 2013. 2

[23] D. F. Fouhey, A. Gupta, and M. Hebert. Unfolding an indoor origami world. In *European Conference on Computer Vision*, pages 687–702. Springer, 2014. 2

[24] D. Girardeau-Montaut. CloudCompare3D point cloud and mesh processing software. *OpenSource Project*, 2011. 2

[25] R. Guo and D. Hoiem. Support surface prediction in indoor scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2144–2151, 2013. 2

[26] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 564–571, 2013. 2

[27] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014. 2

[28] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531. IEEE, 2014. 2

[29] V. Hedau, D. Hoiem, and D. Forsyth. Recovering free space of indoor scenes from a single image. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2807–2814. IEEE, 2012. 2

[30] 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, 2, 3, 6, 7, 16, 17

[31] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. VolumeDeform: Real-time volumetric non-rigid reconstruction. *arXiv preprint arXiv:1603.08161*, 2016. 2

[32] C. Ionescu, F. Li, and C. Sminchisescu. Latent structured models for human pose estimation. In *2011 International Conference on Computer Vision*, pages 2220–2227. IEEE, 2011. 2

[33] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2014. 2

[34] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3D object dataset: Putting the Kinect to work. In *Consumer Depth Cameras for Computer Vision*, pages 141–165. Springer, 2013. 2

[35] O. Kahler, V. Adrian Prisacariu, C. Yuheng Ren, X. Sun, P. Torr, and D. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1241–1250, 2015. 4

[36] O. Kähler, V. A. Prisacariu, and D. W. Murray. Real-time large-scale dense 3D reconstruction with loop closure. In *European Conference on Computer Vision*, pages 500–516. Springer, 2016. 4

[37] A. Karpathy, S. Miller, and L. Fei-Fei. Object discovery in 3D scenes via shape analysis. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2088–2095. IEEE, 2013. 5

[38] M. Kepski and B. Kwolek. Fall detection using ceiling-mounted 3D depth camera. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, volume 2, pages 640–647. IEEE, 2014. 2

[39] B.-s. Kim, P. Kohli, and S. Savarese. 3d scene understanding by voxel-crf. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1425–1432, 2013. 8

[40] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao. Chisel: Real time large scale 3D reconstruction onboard a

mobile device using spatially hashed signed distance fields. In *Robotics: Science and Systems*, 2015. 4

[41] H. S. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from RGB-D videos. *The International Journal of Robotics Research*, 32(8):951–970, 2013. 2

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[43] Y. Li, A. Dai, L. Guibas, and M. Nießner. Database-assisted object retrieval for real-time 3D reconstruction. In *Computer Graphics Forum*, volume 34, pages 435–446. Wiley Online Library, 2015. 2

[44] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3D object detection with RGBD cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1417–1424, 2013. 2

[45] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 1

[46] L. Liu and L. Shao. Learning discriminative representations from RGB-D video data. In *IJCAI*, volume 1, page 3, 2013. 2

[47] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 7

[48] M. Luber, L. Spinello, and K. O. Arras. People tracking in RGB-D data with on-line boosted target models. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3844–3849. IEEE, 2011. 2

[49] J. Mason, B. Marthi, and R. Parr. Object disappearance for object discovery. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2836–2843. IEEE, 2012. 2

[50] O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornung, and R. Pajarola. Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum*, volume 33, pages 11–21. Wiley Online Library, 2014. 2, 7

[51] D. Maturana and S. Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 2

[52] S. Meister, S. Izadi, P. Kohli, M. Hämmerle, C. Rother, and D. Kondermann. When can we use KinectFusion for ground truth acquisition. In *Workshop on Color-Depth Camera Fusion in Robotics, IROS*, volume 2, 2012. 2

[53] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361, 2010. 2

[54] R. Min, N. Kose, and J.-L. Dugelay. KinectFaceDB: A Kinect database for face recognition. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(11):1534–1548, 2014. 2

[55] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 1, 2, 6, 16, 21

[56] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011. 2, 4

[57] D. T. Nguyen, B.-S. Hua, L.-F. Yu, and S.-K. Yeung. A robust 3D-2D interactive tool for scene segmentation and annotation. *arXiv preprint arXiv:1610.05883*, 2016. 2, 3

[58] B. Ni, G. Wang, and P. Moulin. RGBD-HuDaAct: A color-depth video database for human daily activity recognition. In *Consumer Depth Cameras for Computer Vision*, pages 193–208. Springer, 2013. 2

[59] 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. 2, 4, 16

[60] Occipital. Occipital: The structure sensor, 2016. 3

[61] K. Phillip and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Adv. Neural Inf. Process. Syst*, 2011. 8

[62] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart. Tracking a depth camera: Parameter exploration for fast ICP. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3824–3829. IEEE, 2011. 2

[63] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. *arXiv preprint arXiv:1604.03265*, 2016. 2, 7, 8

[64] X. Ren, L. Bo, and D. Fox. RGB-(D) scene labeling: Features and algorithms. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2759–2766. IEEE, 2012. 2

[65] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4791–4796. IEEE, 2012. 2

[66] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008. 2

[67] M. Savva, A. X. Chang, P. Hanrahan, M. Fisher, and M. Nießner. SceneGrok: Inferring action maps in 3D environments. *ACM Transactions on Graphics (TOG)*, 33(6):212, 2014. 2

[68] 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, 2, 3, 16, 17

[69] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pages 2930–2937, 2013. 2

[70] A. Shrivastava and A. Gupta. Building part-based object detectors via 3D geometry. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1745–1752, 2013. 2

[71] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition*, 2011. 1, 2

[72] S. Song, S. P. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576, 2015. 1, 2

[73] S. Song and J. Xiao. Sliding shapes for 3D object detection in depth images. In *European Conference on Computer Vision*, pages 634–651. Springer, 2014. 2

[74] S. Song and J. Xiao. Deep sliding shapes for amodal 3D object detection in RGB-D images. *arXiv preprint arXiv:1511.02300*, 2015. 1, 2

[75] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *arXiv preprint arXiv:1611.08974*, 2016. 2

[76] L. Spinello and K. O. Arras. People detection in RGB-D data. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3838–3843. IEEE, 2011. 2

[77] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738. ACM, 2013. 2

[78] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012. 1, 2

[79] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proc. ICCV*, 2015. 7

[80] J. Sung, C. Ponce, B. Selman, and A. Saxena. Human activity detection from RGBD images. *plan, activity, and intent recognition*, 64, 2011. 2

[81] A. Teichman, S. Miller, and S. Thrun. Unsupervised intrinsic calibration of depth sensors via SLAM. In *Robotics: Science and Systems*, volume 248, 2013. 3, 20

[82] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin. Learning to navigate the energy landscape. *arXiv preprint arXiv:1603.05772*, 2016. 2

[83] J. Valentin, V. Vineet, M.-M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. Torr. SemanticPaint: Interactive 3D labeling and learning at your fingertips. *ACM Transactions on Graphics (TOG)*, 34(5):154, 2015. 2, 8

[84] H. Wang, J. Wang, and W. Liang. Online reconstruction of indoor scenes from RGB-D streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3271–3279, 2016. 3

[85] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended KinectFusion. 2012. 4

[86] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM without a pose graph. *Proc. Robotics: Science and Systems, Rome, Italy*, 2015. 4

[87] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, page 0278364916669237, 2016. 4

[88] 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. 1, 2, 6, 7, 8, 21

[89] J. Xiao, A. Owens, and A. Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1625–1632. IEEE, 2013. 1, 2

[90] B. Zeisl, K. Koser, and M. Pollefeys. Automatic registration of RGB-D scans via salient directions. In *Proceedings of the IEEE international conference on computer vision*, pages 2808–2815, 2013. 2

[91] J. Zhang, C. Kan, A. G. Schwing, and R. Urtasun. Estimating the 3D layout of indoor scenes and its clutter from depth sensors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1273–1280, 2013. 2

[92] M. Zollhöfer, A. Dai, M. Innmann, C. Wu, M. Stamminger, C. Theobalt, and M. Nießner. Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics (TOG)*, 34(4):96, 2015. 2

A. 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.

A.1. Example Annotated Reconstructions

Fig. 10 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 [18] 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. 10.

Fig. 11 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).

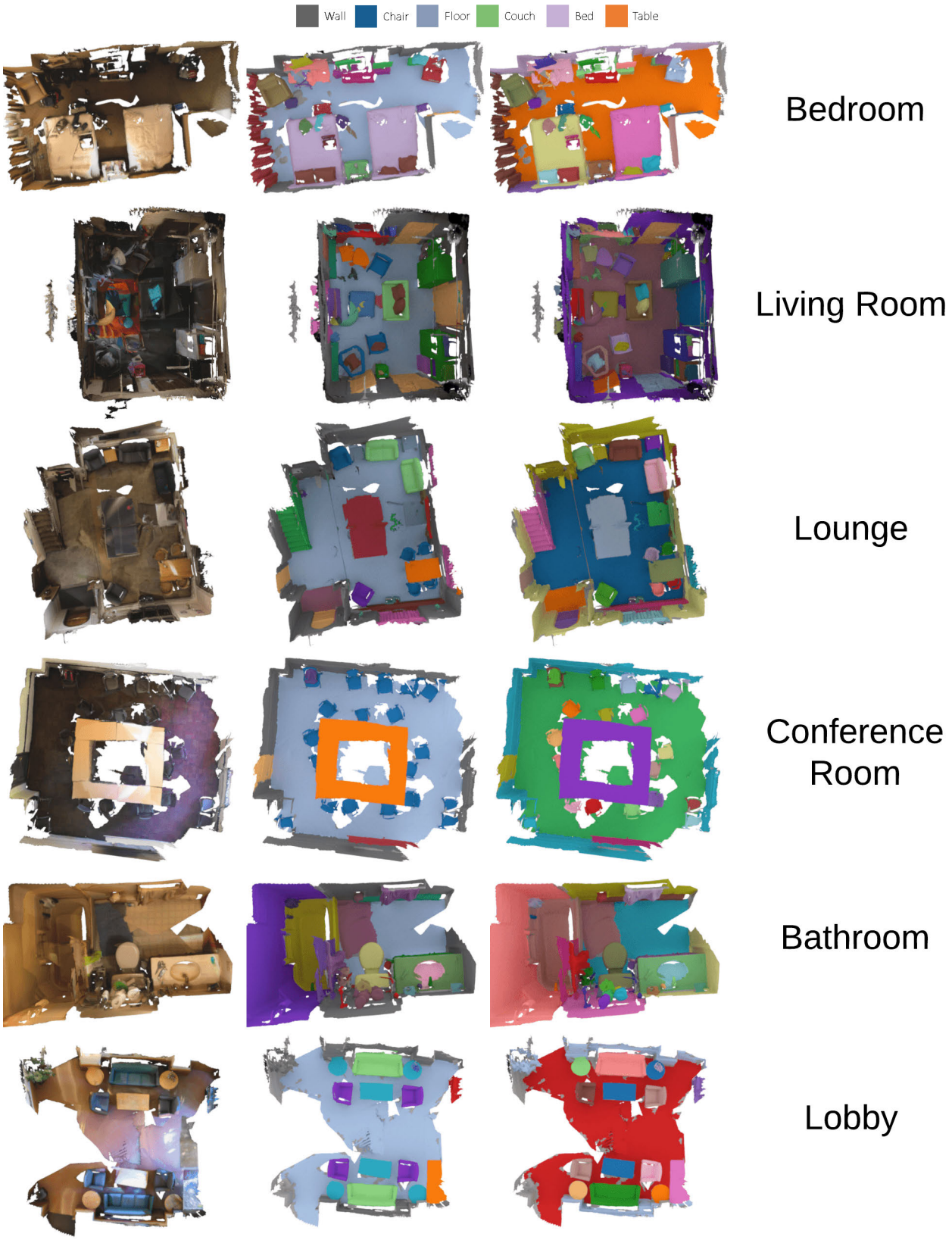

Figure 10. Example annotated scans in ScanNet. **Left:** reconstructed surface mesh with original colors. **Middle:** color indicates category label consistently across all scans. **Right:** each object instance shown with a different randomly assigned color.

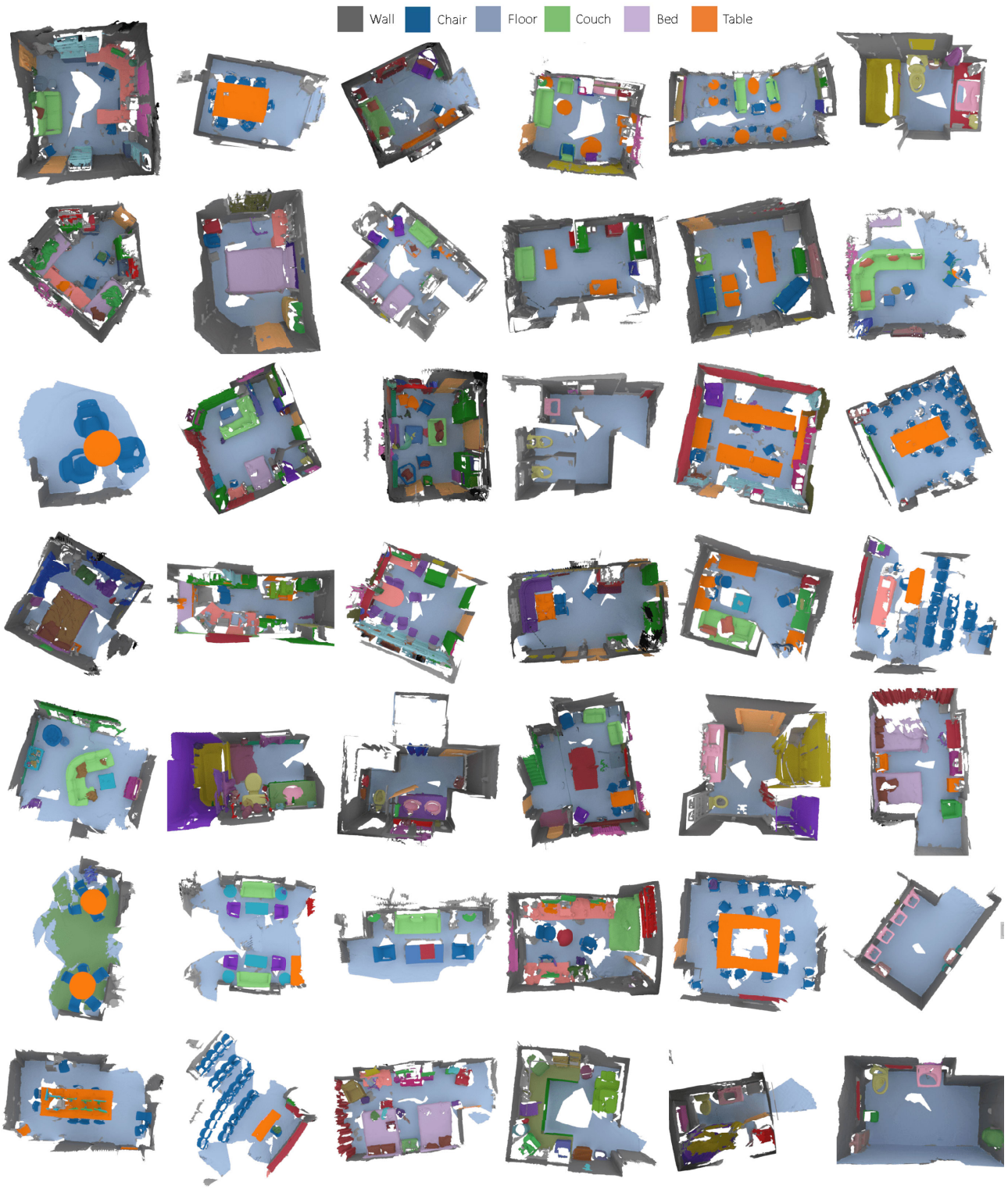

Figure 11. A variety of example annotated scans in ScanNet. Colors indicate category consistently across all scans.

ScanNet		SceneNN [30]	
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 7. Total counts of annotated object instances of the 50 largest categories in ScanNet (left), and in SceneNN [30] (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.

A.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 $22.3\,\mathrm{min}$.

A.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 [55]), or they focus on annotating decontextualized objects and not scenes (e.g., Choi et al. [10]). The two datasets that do annotate densely reconstructed RGB-D spaces at the scene level are the SceneNN dataset by Hua et al. [30] and the smaller PiGraphs dataset by Savva et al. [68].

SceneNN consists of 94 RGB-D scans captured using Asus Xtion Pro devices and reconstructed with the method of Choi et al. [9]. 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 [68] dataset consists of 26 RGB-D scans captured with Kinect v1 devices and reconstructed with the VoxelHashing approach of Nießner et al. [59]. 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. 12 plots histograms of the total number of labeled object instances and the total numbers of unique semantic 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 categories in Fig. 13. This mapping is leveraged during

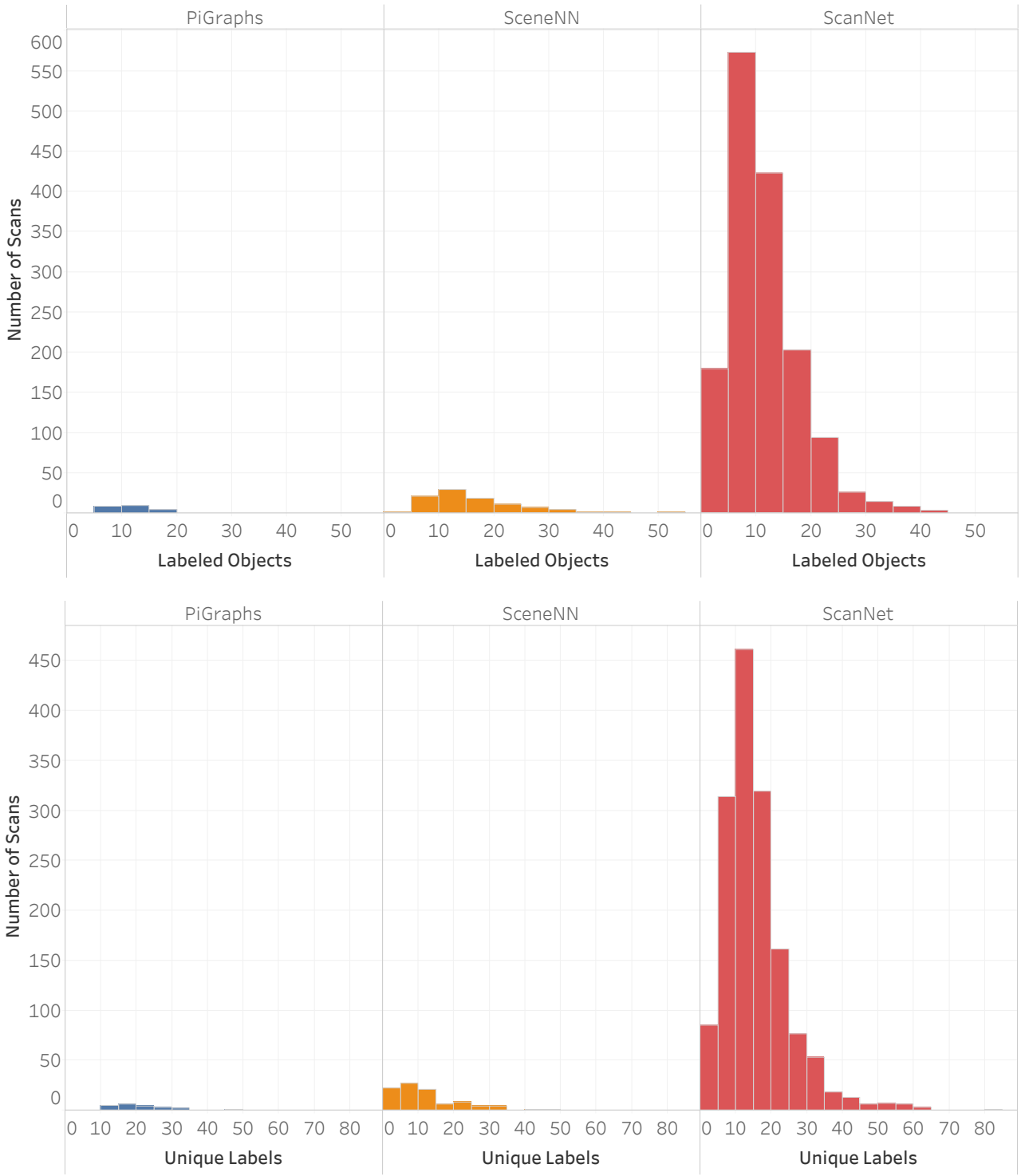

Figure 12. Histograms of the total number of objects labeled per scan (**top**) and total number of unique labels per scan (**bottom**) in the PiGraphs [68], SceneNN [30] and our dataset (ScanNet). The histograms show that ScanNet has many annotated objects over a larger number of scans, ranging in complexity with regards to the total number of objets per scan.

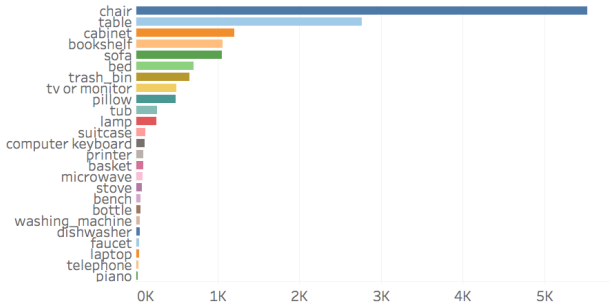

Figure 13. Top 25 most frequent annotation labels in ScanNet scans mapped to ShapeNetCore classes. ScanNet has thousands of 3D reconstructed instances of common objects such as chairs, tables, and cabinets.

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.

A.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. 14 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 sequences are more complete surface reconstructions of the

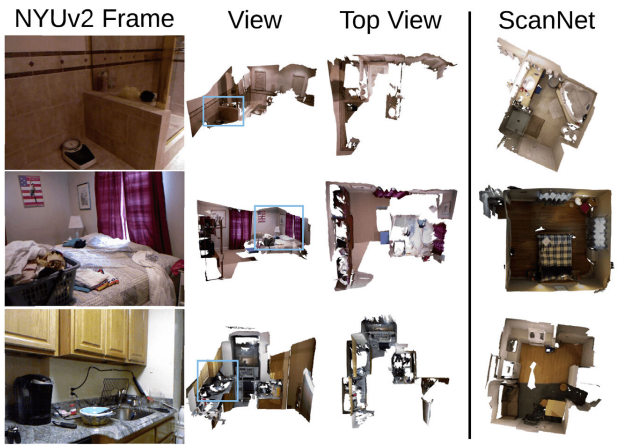

Figure 15. Comparison of reconstructed Bathroom (**top**), Bedroom (**middle**), and Kitchen (**bottom**) from NYUv2 RGB-D frames (**left**), and a comparable reconstruction from ScanNet (**right**). For each NYU scene, we show an example color frame, the rough corresponding region of the view in the 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.

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. 15 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. 14.

B. 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 acquisition, 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 Appendix B.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 Appendix B.1). We also estab-

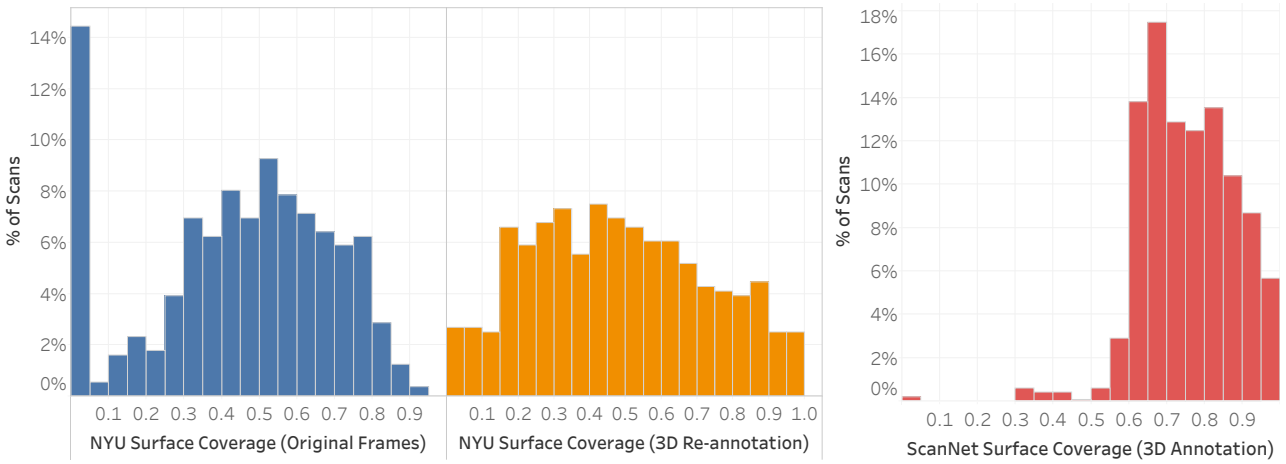

Figure 14. Histograms of the percentage of total reconstruction surface area per scan that is semantically labeled for: NYU v2 reconstructions using projection of RGB-D annotated frames (**left**), for NYU v2 reconstructions using our 3D annotation interface (**middle**), and for ScanNet reconstructions similarly annotated with our interface (**right**).

lished a simple calibration process that novice users could carry out (see Appendix B.2), and offloaded RGB-D reconstruction to the cloud (see Appendix B.3). Finally, we developed web-based UIs for crowdsourcing semantic annotation tasks as described in Appendix B.4, and for managing the collected data as described in Appendix B.6.

B.1. RGB-D Acquisition UI

Fig. 16 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.

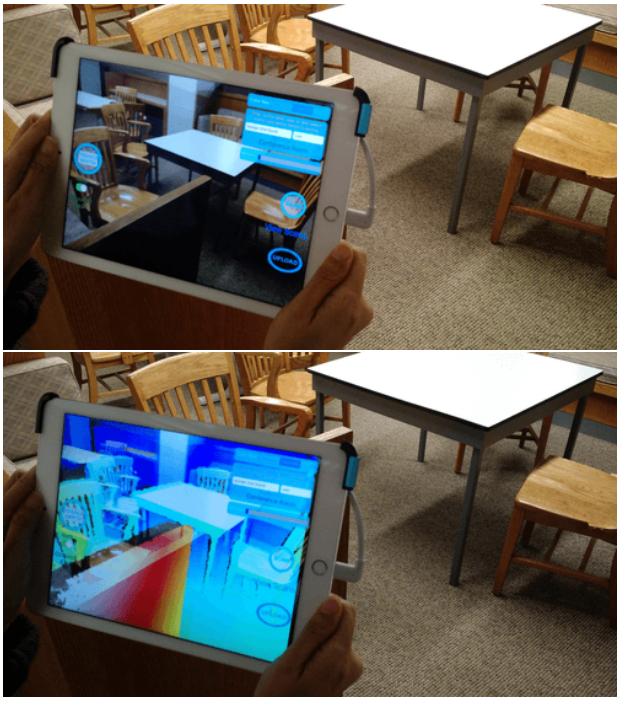

Figure 16. Our RGB-D recording app on an iPad Air2 with attached Structure sensor (showing color stream at the top and depth stream at the bottom). The app allows novice users to record RGB-D videos and upload to a server for reconstruction and annotation.

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 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 $30\,\mathrm{ms}$.

B.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.

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. [81]. 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 [14]. 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. [81]. We found that undistorting depth images obtained by a Structure sensor leads to significantly improved tracking.

B.3. Surface Reconstruction

Given a calibrated RGB-D sequence as input, a fused 3D surface reconstruction is obtained using the BundleFusion framework [12], as described in the main paper. The reconstruction is then cleaned by merging vertices within $1\,\mathrm{mm}$ 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 $146\,\mathrm{MB}$ to $5.82\,\mathrm{MB}$ 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.

B.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. 17) 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. 18 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.

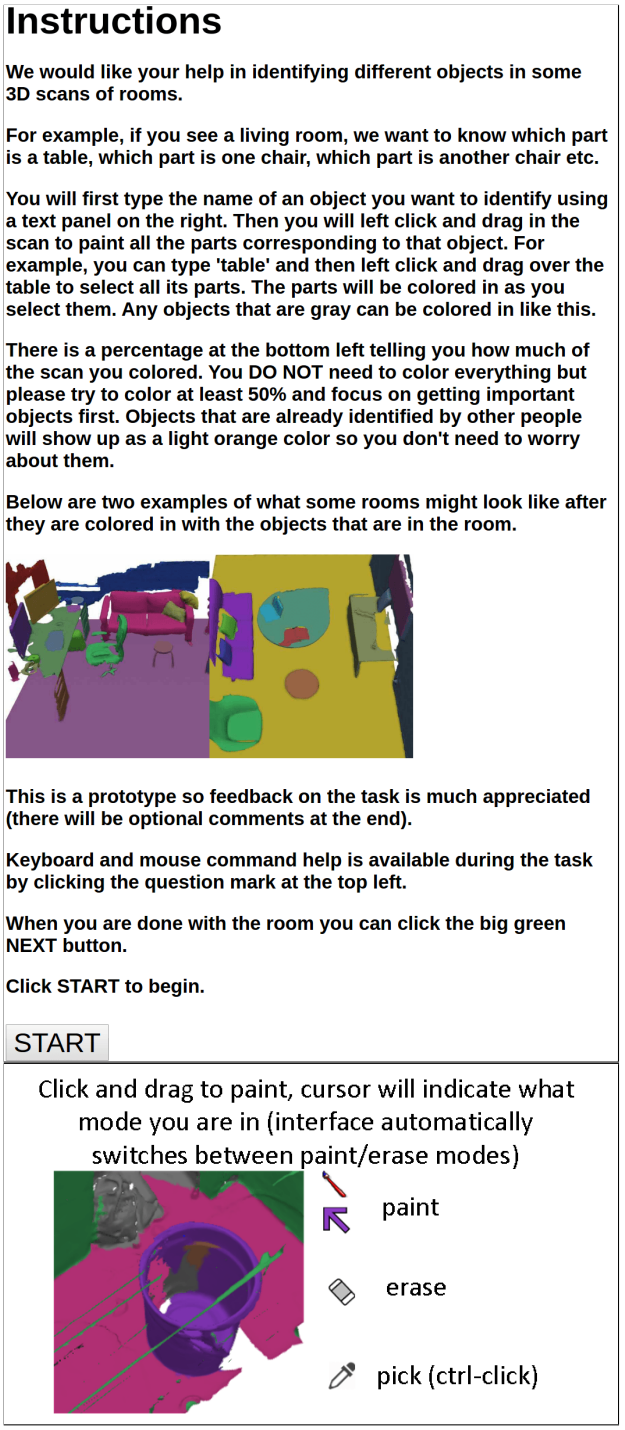

Figure 17. Instructions provided to crowd workers for our semantic annotation task. **Top:** instructions before the beginning of the task. **Bottom:** interface instructions during annotation.

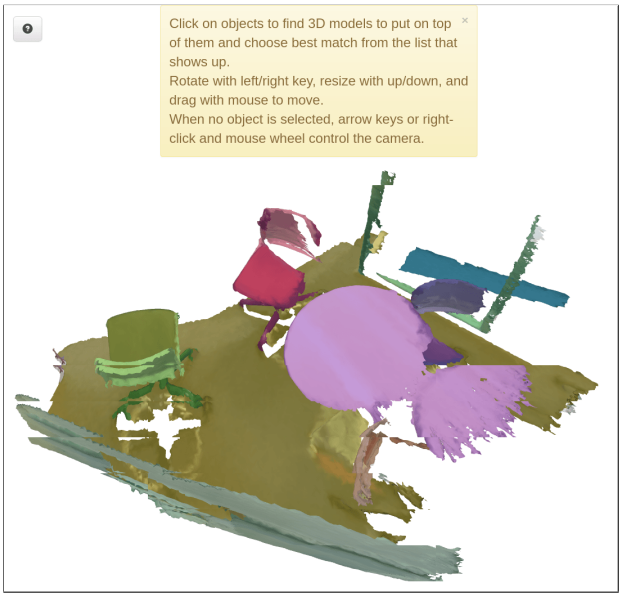

Figure 18. Instructions provided to crowd workers for our CAD model alignment task. The worker clicks on colored objects to retrieve and place CAD models.

B.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 [18] synsets when possible, and to other common label sets that are commonly used for RGB-D and 3D CAD data (NYUv2 [55], ModelNet [88], and ShapeNetCore [6]).

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

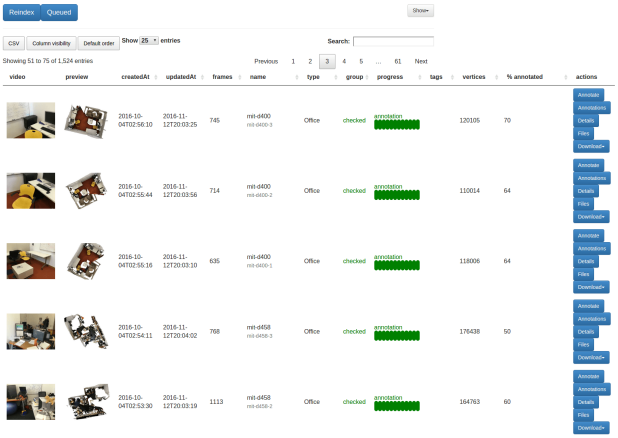

Figure 19. Our web-based data management UI for ScanNet scan data.

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.

B.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. 19). When a user is finished scanning and presses the upload button on an iPad device, their scan data is automatically 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 are visible and usable for sorting and filtering of the scan database.