SUN3D: A Database of Big Spaces Reconstructed using SfM and Object Labels

Jianxiong Xiao
Princeton University

Andrew Owens
MIT

Antonio Torralba
MIT

Abstract

Existing scene understanding datasets contain only a limited set of views of a place, and they lack representations of complete 3D spaces. In this paper, we introduce SUN3D, a large-scale RGB-D video database with camera pose and object labels, capturing the full 3D extent of many places. The tasks that go into constructing such a dataset are difficult in isolation – hand-labeling videos is painstaking, and structure from motion (SfM) is unreliable for large spaces. But if we combine them together, we make the dataset construction task much easier. First, we introduce an intuitive labeling tool that uses a partial reconstruction to propagate labels from one frame to another. Then we use the object labels to fix errors in the reconstruction. For this, we introduce a generalization of bundle adjustment that incorporates object-to-object correspondences. This algorithm works by constraining points for the same object from different frames to lie inside a fixed-size bounding box, parameterized by its rotation and translation. The SUN3D database, the source code for the generalized bundle adjustment, and the web-based 3D annotation tool are all available at http://sun3d.cs.princeton.edu.

1. Introduction

The popularity of the Microsoft Kinect and other depth-capturing devices has led to a renewed interest in 3D for recognition. Researchers have extended traditional object and scene recognition datasets to incorporate 3D. For example, [13] is an evolution of popular 2D object datasets such as Caltech 101 [5] to 3D objects captured by an RGB-D camera. The NYU Depth dataset [20] and others [11, 12, 2] go beyond objects by capturing RGB-D videos of scenes and labeling the objects within. However, these 3D datasets inherit many of the limitations of traditional 2D datasets: they contain a sample of views from the world, but the physical relationship *between* these views and the structure of the space containing them is mostly missing.

What we desire is a dataset that is *place-centric* rather than view-based, containing full 3D models of spaces (*e.g.* entire apartments) instead of a limited set of views (Fig. 1). Such a database would allow us to ask questions like: "what does this object look like from behind?" or "what can we expect the space to look like beyond the available field of view?" Such a database would be useful for learning complete 3D context models to be used for scene parsing (*e.g.* learning that there is always a bed in a bedroom); for obtaining an integrated understanding of a space instead of individual disconnected snapshots; and for reasoning about intuitive physics, functionality and human activity.

2D view-based SUN database 3D place-centric SUN3D database

Figure 1. **View-based vs. place-centric.** This example shows the difference between a view-based scene representation and a place-centric scene representation. SUN database [24] contains a view of a living room. SUN3D database contains an RGB-D video for the whole apartment and 3D models with camera poses.

With this goal in mind, we introduce *SUN3D*, a place-centric database. The items in our database are full 3D models with semantics: RGB-D images, camera poses, object segmentations, and point clouds registered into a global coordinate frame.

This database requires camera poses, but estimating them reliably for large space from an RGB-D video is a difficult problem. And despite recent progress in RGB-D structure-from-motion (SfM), existing automatic reconstruction methods are not reliable enough for our purposes. Additionally, we desire a semantic segmentation, but labeling every frame in a full video is a painstaking task – for this reason, existing RGB-D video databases [20] have semantic annotations only for a sparse subset of their frames.

To address this, we design our 3D reconstruction and object labeling tasks so that they mutually support one another (see Figure 2). Our approach is based on the idea that if the

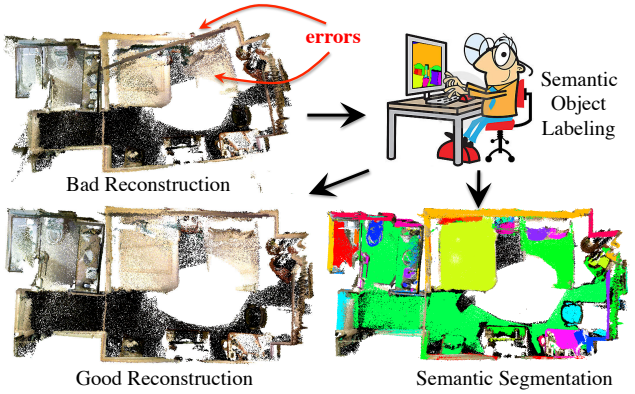


Figure 2. **Main idea.** Semantic object labeling as a way to correct pose estimation errors.

3D reconstruction were perfect, then object labeling would be easy – one would merely need to label an object in one frame, and the reconstruction could be used to propagate these annotations to the rest of the images. On the other hand, if objects were annotated in every frame, then reconstruction would improve dramatically since consistencies between frames could be used as constraints in optimization. By combining the two tasks, we (a) produce better 3D reconstructions, and (b) provide an object annotation tool that makes it easy to label long RGB-D videos.

To produce better reconstructions, we incorporate object labels into our structure-from-motion algorithm and solve jointly for object locations and camera poses. The resulting algorithm is based on standard bundle adjustment, and the addition of object labels helps to avoid errors due to drift and loop-closing failures, establishing "long-range" connections between frames that may be very far apart in a video but that nevertheless contain the same object instances.

Additionally, we use the 3D reconstruction to help with object annotation, creating a tool that speeds up the process of labeling a long video. A user labels an object in one frame, and the partially completed reconstruction is used to propagate object labels to other frames.

1.1. Related work

There are several 3D datasets that capture full videos [20, 21, 12, 2, 7] rather than snapshots [11]. Especially noteworthy is the NYU Depth dataset [20], which contains scans of a large number of scenes. Despite the large quantity of scans, it is still very much a view-based dataset. As an experiment, we reconstructed a large sample of the sequences in the NYU dataset using our structure-from-motion pipeline (Section 3) and found the coverage of most spaces to be incomplete. Figure 3 compares the size of the spaces scanned in several datasets. We can see that a typical scan covers only a small portion of each space, *e.g.* only the corner of a room.

Figure 3. **Space coverage.** Top view of the reconstruction results to illustrate the typical 3D space covered by four different datasets at the same physical scale (meters). (a) RGB-D Object dataset [13]: multiple views of a small object; (b) Berkeley 3-D Object [11]: one view of a scene; (c) NYU Depth [20]: one corner of a room; (d) Ours: a full apartment including bedroom, living room, kitchen, and bathroom.

In this paper, our goal is to develop tools that will allow us to build an annotated database of full 3D places. Although there are several easy-to-use tools for annotating 2D images and videos [18, 26, 22], they would be suboptimal to use for RGB-D sequences because they do not exploit the 3D structure of the problem in label propagation. In this paper we introduce a structure-from-motion pipeline that makes use of RGB, depth, and semantic annotations to improve the robustness of the 3D reconstructions. The tool uses 3D structure to propagate object labels to unlabeled frames and uses object labels to improve the 3D reconstruction. [25, 7] also use 3D information to help labeling, but the labeling happens in 3D space, which requires accurate camera poses to begin with and also 3D knowledge that is only available to experienced users.

Besides the dataset aspect of our work, there are several relevant works. [23] proposed a "place-centric" representation for scene understanding, but they only focus on 2D recognition and the place-centric representation built in [23] is not a complete 3D representation of the full extent of the space. While there are also several recent works [3, 6, 25, 19, 8] that combine recognition and 3D reconstruction, they do not use human annotations: instead, they jointly recognize objects and solve for 3D models. Our semantic labeling is much more reliable because it comes from user annotations and is essentially ground truth. In this sense, our work is related to match-moving software, where a user manually specifies correspondences; however this process is unintuitive, and may be intractable for long sequences.

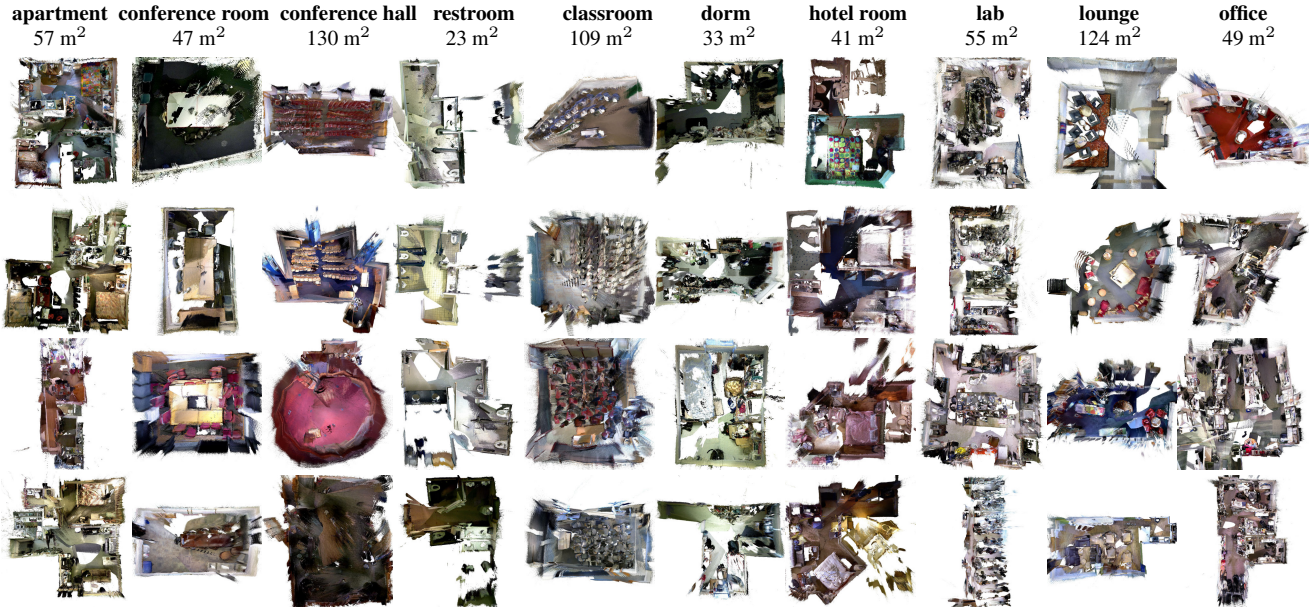


Figure 4. **SUN3D database.** Each column contains examples from a place category. The numbers are the median coverage areas.

	NYU Depth V2	SUN3D
raw video	yes	yes
coverage area	part of a room	whole room or multi-rooms
typical length	hundreds of frames	tens of thousands frames
camera poses	no	yes
object label	sparse frames	whole video
instance label	within one frame	within the whole video
depth improvement	cross-bilateral filtering	multi-frame TSDF filling

Table 1. **Differences** between NYU Depth [20] and SUN3D.

There are several systems that use RGB-D cameras to produce 3D models of spaces, *e.g.* [10, 4, 15]. These systems are oriented toward real-time reconstruction, and they allow the user to interact with the tool to correct errors as they appear. However, scanning a big space remains challenging as reconstruction errors are frequent and the user needs to understand how to correct errors and rescan the place, a process that is especially challenging for long-range errors such as loop-closing failures.

The distinction between view-based and place-centric representations has been studied in human vision. Notably, neuroscience studies (*e.g.* [17]) found that the parahippocampal place area (PPA), a brain area known to represent scenes and spatial layout properties, has a view-specific representation. In contrast, the retrosplenial cortex (RSC), a brain area that underpins episodic memory, navigation, route learning, imagination and planning, exhibited place-centric view-invariant representation. The complementary roles of these two scene-sensitive regions conceptually match the complementary functions of view-based databases (*e.g.* SUN [24]) and place-centric databases (*e.g.* SUN3D).

Device Capturing Pose Immediate Visual Feedback

Figure 5. **Data capturing setup.** A operator carries an RGB-D sensor mounted on a laptop, mimicking natural exploration.

1.2. Overview

We describe our process of capturing the SUN3D database in Section 2. To obtain the camera pose, we propose a system that allows a human to help with pose estimation for an RGB-D video. The first step of this system is to build an initial reconstruction with an RGB-D SfM (Section 3). This reconstruction usually has errors, but it is still useful for the object annotation task (Section 4). Our system then uses these object annotations as constraints in a generalized bundle adjustment procedure (Section 5) to refine the camera pose.

2. SUN3D: A Place-centric 3D Database

We introduce a dataset of full 3D spaces, scanned with an RGB-D sensor. Our database offers RGB-D frames with semantic object segmentations and camera pose. These pieces can be put together in interesting ways. For example, they can be used to obtain: (a) A point cloud for the whole space (Figure 4); (b) 3D object models (Figure 14); (c) All of the viewpoints of an object, and their poses relative to that object; (d) A map of a room, showing all of the objects and their semantic labels from a bird's-eye view.

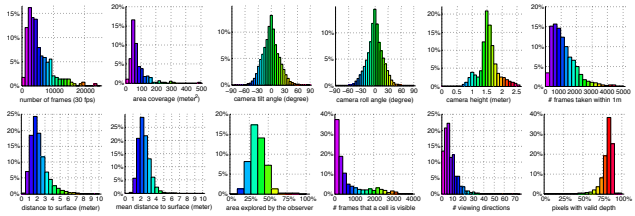
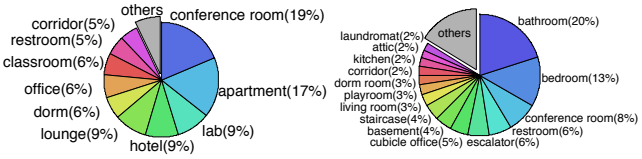


Figure 6. Geometric statistics of SUN3D.



Place category distribution. View category distribution.

Figure 7. Semantic statistics of SUN3D.

Capturing setup For capturing, we mount an ASUS Xtion PRO LIVE sensor to a laptop (Figure 5). To make a database that closely matches the human visual experience, we want the RGB-D video to be taken at a height and viewing angle similar to that of a human. Therefore, as shown in Figure 5, the operator carries the laptop with the Xtion sensor on his or her shoulder with a viewing angle that is mostly horizontal but tilted slightly toward the ground. We use OpenNI to record the video for both RGB and depth at 640×480 resolution, and at 30 frames per second. We use the default factory sensor calibration for the registration between the depth and image. We scan only indoor spaces, because our depth cameras do not work under direct sunlight.

Capturing procedure Each operator is told to mimic human exploration of the space while capturing. They are told to walk through the entire space, thoroughly scanning each room, including the floor and walls and every object. Guidelines include walking slowly, keeping the sensor upright, avoiding textureless shots of walls and floors, and walking carefully between rooms to avoid reconstructions with disconnected components.

Dataset analysis We have 415 sequences captured for 254 different spaces, in 41 different buildings. Operators capture some places for multiple times, at different times of day when possible. Geographically, the places scanned are mainly distributed across North America, Europe and Asia. Statistics are shown in Figure 7.

3. Data-driven bruce-force SfM

We now describe our automatic SfM algorithm, which we use to obtain initial camera pose estimates. These estimates will then be used as part of the object annotation tool. Our SfM algorithm is based on traditional bundle adjustment, but we take advantage of the content of our input videos to close loops more effectively. Standard SfM often fails to close loops because view-invariance feature matching fails. We take a "data-driven" bruce-force approach to SfM. We set a very conservative threshold for our key-point descriptor matching, and therefore our loop closure will have very high precision with low recall. But we have purposely designed our capturing procedure so that the user makes many passes over the same scene. In some scans, many of the viewpoints essentially appear twice or more, and the distances between key-point descriptors are so small that the key-point matching works. Since so many of the same views appear multiple times, we can match all the pairs of near-identical frames during loop closure. This idea is a natural extension of data-driven approach from other vision tasks, such as scene completion [9]. The following describes the details of our 2D+3D automatic SfM system.

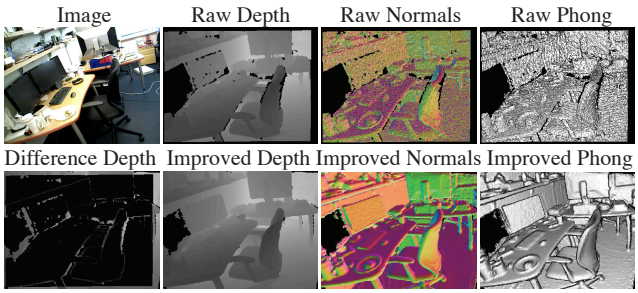


Figure 8. **Multi-view depth improvement.** A TSDF is accumulated using nearby frames to provide more stable depth estimation.

Registering neighboring frames We match each consecutive pair of frames and compute a relative transformation between them. We begin by matching key-points using SIFT and remove poor matches using the ratio test [14]. Within the set of SIFT key-points, we choose the ones with valid depth values, and use a 3-point-algorithm inside a RANSAC loop to find the relative transformation between pairs of frames.

Loop closure To detect loops, we use a Bag of Words model to compute a feature vector for each frame. For each video, we uses k-means to train a codebook for the bag of words model. For a given frame, we compute SIFT features and then compute a visual word histogram, weighted by their inverse frequency in the video (in the standard tf-idf manner). We then compute the dot product between all pairs of feature vectors to obtain the score matrix for possible loop closure pairs. With the score matrix, we use Gaussian smoothing, non-maximum suppression, and then dilation to pick the list of possible pairs. For each pair, we run the pairwise frame-to-frame registration discussed above. If there are more than 25 SIFT key-point correspondences found in the matching, we merge the feature tracks. Since this is a conservative threshold, our loop closure usually has very high precision with low recall, which is desirable in our case, since match errors are difficult to deal with in bundle

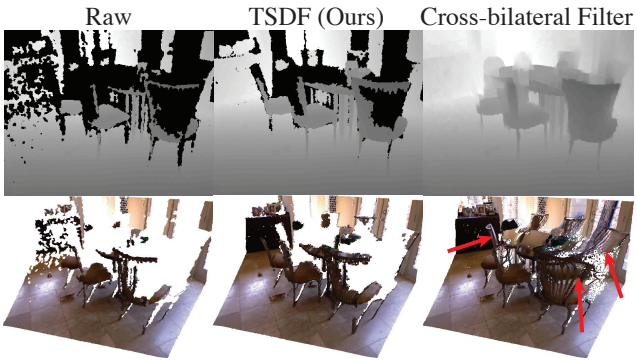


Figure 9. **Comparision of depth improvement algorithms.** The cross-bilateral filtering introduces large artifacts, *e.g.* it smooths over occlusion boundaries, as the 3D point cloud shows.

adjustment.

Joint 2D+3D bundle adjustment We obtain an initial pose estimate by multiplying the relative transformations together in succession. Then, we use the time ordering of frames and the inlier SIFT correspondences to link keypoint tracks for bundle adjustment. Note that the SIFT feature tracks are linked across multiple frames when they share the same location at each frame, and this allows us to have longer feature tracks. We use a joint 2D and 3D objective function for our bundle adjustment as follows:

$$\min \sum_{c} \sum_{p \in \mathrm{V}(c)} \left(\left\| \tilde{\mathbf{x}}_p^c - \mathbf{K} \left[\mathbf{R}_c | \mathbf{t}_c \right] \mathbf{X}_p \right\|^2 + \lambda \left\| \tilde{\mathbf{X}}_p^c - \left[\mathbf{R}_c | \mathbf{t}_c \right] \mathbf{X}_p \right\|^2 \right)$$

where $\mathbf{K}$ is a fixed intrinsics matrix read from device middleware, $\mathbf{R}_c$ and $\mathbf{t}_c$ are the rotation matrix and camera center for the camera corresponding to c-th frame, $\mathbf{X}_p$ is the 3D location of a 3D point visible from the c-th camera (*i.e.* $p \in \mathrm{v}(c)$), and $\tilde{\mathbf{x}}_p^c$ and $\tilde{\mathbf{X}}_p^c$ are the observed 2D pixel location and 3D location in the camera coordinate system respectively.

Depth map improvement The raw depth maps are usually noisy, with many holes. While improving the depth map is useful in itself, it also is helpful when we reconstruct the object polygon during object annotation (described in the next section). To fill in the holes, [20] uses cross-bilateral filtering, which produces a visually pleasing depth map, but it introduces many artifacts (Figure 9). Instead (Figure 8), we improve the depth map using a Truncated Signed Distance Function (TSDF) [15] to voxelize the space, accumulating the depth map from nearby frames (*e.g.* 40 closest frames) using the camera poses obtained above. By using only frames that are nearby in time, the local camera poses are usually easy to obtain reliably. Finally, we use ray casting to get a reliable depth map for each frame.

4. Multi-view object annotation

After the automatic SfM procedure, we have a reconstruction, but that reconstruction will often contain errors.

Figure 10. **Online user interface.** It provides the user a polygon-based tool to outline objects. The tool reconstructs the 3D polygon outlines for an object in the browser, and propagates the results in real time to other frames based on their initial camera poses.

However, many of these errors are *long-range* in nature, the result of small errors accumulating over time, and for any given subsequence the reconstruction is usually quite accurate. We take advantage of this fact to create a LabelMe-style [18] object-annotation interface. And these labels are then used to improve the reconstruction in Section 5.

Users are shown a video player with object annotations superimposed on the image frame, and they can advance the video using a regular video control bar, as shown in Figure 10. They then label objects by clicking on control points along the object's boundary. Upon completion, a popup dialog will appear asking for the object's name. The user can choose an existing name if it is the same instance being labelled in a previous frame, or create a new name if the object appears for the first time.

Whenever a user labels or corrects a frame, the object annotation will be propagated automatically to other frames, so it will be unnecessary to label them if the propagation is correct. The task is finished when the user is satisfied with the annotations in all frames.

Interaction at each frame When the user scrolls to an unlabeled frame, the 3D-based label propagation algorithm will try to propagate labels from frames that have already been labeled (which we call *keyframes*) to this new frame. Now, they can *correct* all mistakes if there are any, or just continue to other frames. If the user decides to *correct* the frame, then they are required to fully correct all of the mislabelings (*i.e.* one object mistakenly labeled as another). Such errors can be due to problems with camera pose or from approximations made by our propagation algorithm. When the user finishes and continues to navigate, the frame automatically becomes a keyframe, and its annotations will be propagated to other frames. Otherwise, if the user chooses to "ignore" the frame, then it means that they did not want

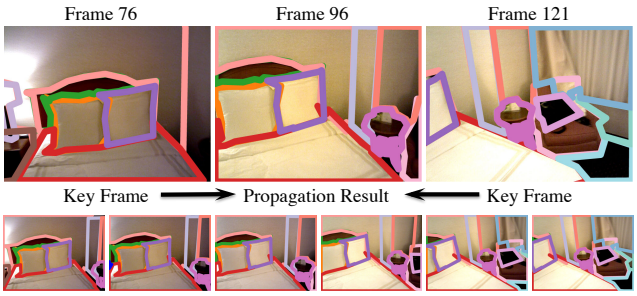


Figure 11. **3D label propagation.** Annotation of each frame is automatically populated from nearby key frames.

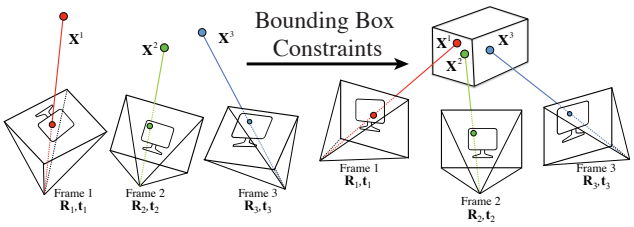


Figure 12. **Generalized bundle adjustment.** The object-to-object correspondence constraint essentially pulls a set of points belonging to the same object so that they fit into one 3D bounding box for that object. Together with constraints from other objects and key-points, the camera pose can be estimated more reliably.

to provide any feedback, and the algorithm does nothing.

Polygon reconstruction For a given manually annotated object polygon, we robustly estimate the 3D locations of the control points – these 3D control points are later used to define a new polygon. For each control point, we collect all of the 3D points that fall within 25 pixels of its incident edge (the one connecting with the previous control point) and fit a plane using RANSAC. If there are too few points to do this, *e.g.* if the object is small or depth is missing around the boundary, we fit a plane to all of the object's points. The control point's 3D location is then obtained by intersecting the corresponding camera ray with the plane. This simple scheme can be achieved in real time and implemented in JavaScript running on a standard web browser.

Annotation propagation For an unlabeled frame, we retrieve the closest two keyframes based on frame number. For each of these keyframes, we reproject the 3D object polygons into the current frame using the estimated camera poses, and check for visibility by comparing with the RGB-D depth value in the projected area. Multiple polygons are merged together by the union of the polygons from multiple frames. With this simple propagation scheme, we observe that the results are usually stable when the camera poses are correct, as we can see in the example shown in Figure 11.

Conflict list The major source of propagation error comes from camera pose errors. The user can correct the errors produced in one frame, but it is tedious to correct the errors of every nearby frame as well. Therefore, we maintain a *conflict list* between pairs of frames: when a user corrects a major mistake in a frame, the algorithm checks to see which keyframes the wrong label is propagated from, and places them into the conflict list with the current frame. All nearby frames will exclude frames from the conflict list during propagation. This mechanism significantly reduces the effect of camera pose error.

Instance naming For naming the object, we ask the user to name an object first for its object category, followed by a short description of where it is located, so that they can specify if the same object appears twice in another frame (*e.g.* "chair: next to the laptop"). For object categories with many instances close together or that the user cannot differentiate, *e.g.* hundreds of indistinguishable chairs in a classroom, the user will name them with "object category: *", and we do not use them in the subsequent generalized bundle adjustment.

Discussion The annotation task is intuitive because it requires no knowledge of 3D and geometry, so it can be used easily by general users. Most of the labeling time is spent on increasing the coverage of a space, rather than correcting errors. This is because the RGB-D depth map and local camera pose estimation are usually reliable, and the conflict list effectively deletes bad source keyframes when the camera poses are inconsistent. We have also considered alternative methods, such as drawing bounding boxes in 3D, but all these approaches require the user to work in 3D with potentially wrong camera poses, which requires significant training and geometric knowledge.

5. Generalized bundle adjustment

The object segmentations obtained in the previous section can be used to correct camera pose errors. For example, if the same object instance appears twice in different locations, it signals that the camera pose estimation is incorrect. We desire a way to allow the user to fix these errors.

One way to do this would be to use the annotations to match features between pairs of frames that view the same objects, and to register them as in Section 3. In our experience, this is not usually effective, since most matchable pairs have already been detected during the automatic loop-closing step. Another alternative would be to ask users to supply correspondences by hand, as in match-moving software. However, this is unintuitive and may be intractable for long sequences.

Given that exact point-to-point correspondences are very hard to obtain either automatically or manually, we propose a novel approach to generalize standard bundle adjustment from point-to-point correspondences to one with object-to-object correspondences. We parameterize each object in-

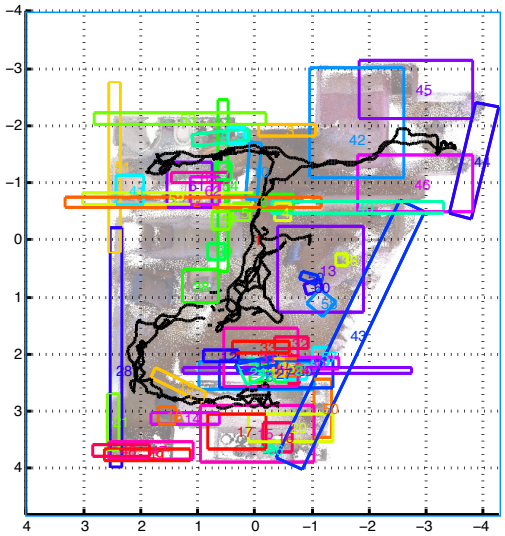


Figure 13. **Object constraints in generalized bundle adjustment.** Each box visualizes a constraint introduced by the user annotation.

stance (recall that annotators specify both the type of object and distinguish between instances) by its 3D location, rotation, and the size of its 3D bounding box. As shown in Figure 12, the 3D point cloud of an object from a view should lie inside the object's inferred 3D bounding box. Therefore, when the same object appears in a different location in the space, the bundle adjustment optimization will pull them together close enough to fit in the same 3D bounding box (since there is only one, limited-size bounding box per object instance). We call this new algorithm *generalized bundle adjustment*, as it generalizes an infinitely small 3D point to a 3D box with certain size and orientation.

More technically, for each object, the 6DOF location $\mathbf{t}_o$ and rotation $\mathbf{R}_o$ are unknown variables to be estimated by the bundle adjustment. The physical size $\mathbf{s}_o$ of the 3D bounding box is provided automatically by the system based on the object category[1]. Encoding the object-to-object correspondences in 3D, together with the original bundle adjustment constraints based on point-to-point tracks, our new objective function is

$$\min \sum_c \sum_{p \in \mathrm{V}(c)} \left(\left\| \tilde{\mathbf{x}}_p^c - \mathbf{K} \left[\mathbf{R}_c | \mathbf{t}_c \right] \mathbf{X}_p \right\|^2 + \lambda_0 \left\| \tilde{\mathbf{X}}_p^c - \left[\mathbf{R}_c | \mathbf{t}_c \right] \mathbf{X}_p \right\|^2 \right) + \lambda_1 \sum_o \sum_c \sum_{p \in \mathrm{L}(o,c)} \Psi(\left[\mathbf{R}_o | \mathbf{t}_o \right] \left[\mathbf{R}_c | \mathbf{t}_c \right]^{-1} \tilde{\mathbf{X}}_p^c, \mathbf{s}_o)^2,$$

where

$$\Psi(\mathbf{X}, \mathbf{s}) = \left\| \max \left(\mathbf{0}, \mathbf{X} - \frac{\mathbf{s}}{2}, -\mathbf{X} - \frac{\mathbf{s}}{2} \right) \right\|$$

is a loss function that has zero value inside a 3D cuboid with size $\mathbf{s}$, and goes linearly outside the cuboid. This means that given a 3D point $\tilde{\mathbf{X}}_p^c$ from the c-th camera being labelled as the o-th object (*i.e.* $p \in \mathrm{L}(o, c)$), we transform it from the local camera coordinate system to the world coordinate system using $\left[\mathbf{R}_c | \mathbf{t}_c \right]^{-1}$, and transform it from the world coordinate system to the object coordinate system using $\left[\mathbf{R}_o | \mathbf{t}_o \right]$; we see how far this point is from the canonical 3D cuboid centered at the origin with size $\mathbf{s}$. This Ψ function is basically an extension of the quadratic loss function typically used for point-to-point correspondences.

[1]We manually construct a table of object size upper bounds for a list of common object categories.

Figure 14. **Object gallery.** Segmented object point cloud merged from different viewpoints.

Special semantics We use the same framework to place further restrictions on objects that have special semantics. For walls and floors, we make the bounding box constraint into a planarity constraint by making the box very thin in one dimension. We further constrain floors so that their bounding boxes are at $y = 0$, and we force walls to be orthogonal to the x-z plane by only allowing rotations around the y axis. We constrain other axis-aligned objects, such as beds and cabinets, in the same way as well. These constraints result in a more accurate, rectified reconstruction.

Object context We note that our general framework can also be used to model contextual object relationships. For example, if one object is parallel with the other object, we can add hard constraints to let the two objects share the same rotation variables. If an object is attached on the other object, not only they are parallel, the bounding box for the smaller object (*e.g.* a painting) should lie inside the bounding box for the bigger object (*e.g.* a wall). Orthogonality (*e.g.* walls meet at right angles) and other angle constraints can be encoded as a relationship between the rotation matrices of two objects. While we do not currently use such constraints, they could easily be added to the optimization under our framework to provide additional regularization.

Optimization We use the Levenberg-Marquardt algorithm to optimize our objective function with automatic differentiation in Ceres solver [1]. In our implementation, we use an angle-axis representation for general 3D rotations,

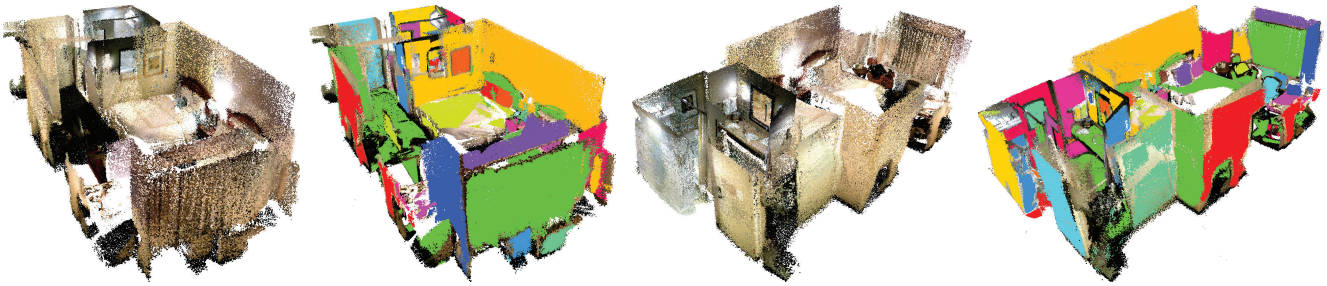


Figure 15. **Annotation and reconstruction correction result.** The 3D point cloud is colored based on their semantic object categories.

except for axis-aligned bounding boxes, where it suffices to use a single angle parameter.

Discussion Although the generalized bundle adjustment is used here with manual annotation, it could potentially be extended to work with automatic object detection as well [3, 6]. Just as the standard bundle adjustment requires most of the point-to-point correspondences to be correct, the generalized bundle adjustment also requires high-quality object-to-object correspondences. And it would be interesting as future work to apply existing methods for dealing with outliers to this new domain (*e.g.* outlier removal and robust loss functions for Ψ, such as cauchy, arctan etc.).

6. Conclusion

We introduce SUN3D, a RGB-D video database of big spaces for place-centric scene understanding. We have proposed a 3D reconstruction and labeling tool: it incorporates semantic labels to obtain an accurate 3D reconstruction, and uses the 3D reconstruction to make an efficient annotation tool. We propose a novel generalized bundle adjustment algorithm to incorporate object-to-object correspondences as constraints. We believe that many new algorithms and applications are enabled by our SUN3D database (*e.g.* [16]). All source code, labeling tool, and data are publicly available to facilitate further research.

Acknowledgments We thank Erika Lee, Tianfan Xue, Deqing Sun for help in data collection. We thank the area chairs for valuable feedback. This work was partially funded by a NDSEG fellowship to A.O and ONR MURI N000141010933 to A.T.

References

[1] S. Agarwal and K. Mierle. *Ceres Solver: Tutorial & Reference*. Google Inc. 7

[2] A. Aydemir, R. Göransson, and P. Jensfelt. *Kinect@Home*, 2012. 1, 2

[3] S. Y. Bao and S. Savarese. Semantic structure from motion. In *CVPR*, 2011. 2, 8

[4] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *ICRA*, 2012. 3

[5] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *CVIU*, 2007. 1

[6] N. Fioraio and L. Di Stefano. SLAM++: Simultaneous localisation and mapping at the level of objects. In *CVPR*, 2013. 2, 8

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2

[8] C. Hane, C. Zach, A. Cohen, R. Angst, and M. Pollefeys. Joint 3d scene reconstruction and class segmentation. In *CVPR*, 2013. 2

[9] J. Hays and A. A. Efros. Scene completion using millions of photographs. *SIGGRAPH*, 2007. 4

[10] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3d modeling of indoor environments, 2010. 3

[11] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3-d object dataset: Putting the kinect to work. In *ICCV Workshop*, 2011. 1, 2

[12] H. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*, 2011. 1, 2

[13] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *ICRA*, 2011. 1, 2

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 4

[15] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 3, 5

[16] A. Owens, J. Xiao, A. Torralba, and W. T. Freeman. Shape anchors for data-driven multi-view reconstruction. In *ICCV*, 2013. 8

[17] S. Park and M. M. Chun. Different roles of the parahippocampal place area (ppa) and retrosplenial cortex (rsc) in panoramic scene perception. *NeuroImage*, 47(4):1747–1756, 2009. 3

[18] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: A database and web-based tool for image annotation. *IJCV*, 2008. 2, 5

[19] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. Joint detection, tracking and mapping by semantic bundle adjustment. In *CVPR*, 2013. 2

[20] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1, 2, 3, 5

[21] S. Song and J. Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. In *ICCV*, 2013. 2

[22] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 2012. 2

[23] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba. Recognizing scene viewpoint using panoramic place representation. In *CVPR*, 2012. 2

[24] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 1, 3

[25] J. Xiao and L. Quan. Multiple view semantic segmentation for street view images. In *ICCV*, 2009. 2

[26] J. Yuen, B. C. Russell, C. Liu, and A. Torralba. Labelme video: Building a video database with human annotations. In *ICCV*, 2009. 2