SceneNN: a Scene Meshes Dataset with aNNotations

Binh-Son Hua[1] Quang-Hieu Pham[1] Duc Thanh Nguyen[2] Minh-Khoi Tran[1]
Lap-Fai Yu[3] Sai-Kit Yeung[1]

[1]Singapore University of Technology and Design
[2]Deakin University [3]University of Massachusetts Boston

Abstract

Several RGB-D datasets have been publicized over the past few years for facilitating research in computer vision and robotics. However, the lack of comprehensive and fine-grained annotation in these RGB-D datasets has posed challenges to their widespread usage. In this paper, we introduce SceneNN, an RGB-D scene dataset consisting of 100 scenes. All scenes are reconstructed into triangle meshes and have per-vertex and per-pixel annotation. We further enriched the dataset with fine-grained information such as axis-aligned bounding boxes, oriented bounding boxes, and object poses. We used the dataset as a benchmark to evaluate the state-of-the-art methods on relevant research problems such as intrinsic decomposition and shape completion. Our dataset and annotation tools are available at http://www.scenenn.net.

1. Introduction

Literature has shown that daunting challenges in computer vision, *e.g.*, intrinsic decomposition, edge detection, object detection and recognition, to name a few, can be solved effectively by convolutional deep neural networks [18, 47, 37, 11]. However, in order to train deep networks, a huge amount of data is required. In 2D, building datasets of millions of images is no longer a problem, thanks to the development of social networks and photo-sharing sites. Recent progress in consumer-grade depth cameras has demonstrated the potential of creating datasets containing millions of RGB-D images, *e.g.*, the NYU dataset [27] for scene segmentation and the SUN-RGBD dataset [28] for scene understanding.

While photographs and RGB-D images can be captured easily, reconstructing a complete scene in 3D is much more challenging. Recent advances in 3D reconstruction techniques have facilitated this task significantly, but this task is still far from trivial considering that very often at least tens of range images must be aligned and stitched together to create a high-quality 3D scene. In addition, documentation on the entire pipeline for scene reconstruction and annotation are rather limited. When releasing a new scene dataset, it is particularly important to clearly document the process and techniques adopted for creating the dataset to sustain its continuous growth.

Our goal is to create a dataset of real-world 3D scenes with annotations. Figure 1 shows a few example scenes in our dataset. Figure 2 shows an overview of our processing pipeline. Our major contributions are:

- We reconstructed the triangle meshes of 100 indoor scenes, including offices, bedrooms, living rooms, kitchens and scenes with repetitive objects. We release the scene dataset together with the RGB-D videos from which the scenes were reconstructed.
- We annotated all objects in the scenes. The segmentation and annotations were done per vertex and per pixel in both 3D and 2D. We also enriched the annotations with fine-grained information, *e.g.*, axis-aligned bounding boxes, oriented bounding boxes, and object poses.
- We demonstrated the use of our scenes for several applications, including benchmarking state-of-the-art shape completion methods, relighting the scenes using reflectance obtained from intrinsic decomposition, synthesizing novel views that are not captured in the original RGB-D videos, and synthesizing CAD scenes using statistics extracted from our dataset.

2. Related Work

In this section, we mainly review the relevant datasets. Specific techniques applied in the reconstruction pipeline are discussed in the subsequent sections. Our dataset is

Figure 1. Example scenes of our dataset. The complexity of indoor scenes in the real world is depicted by the amount of objects and clutter. Our scenes come with both mesh and color texture data, as well as dense annotations in both 2D and 3D.

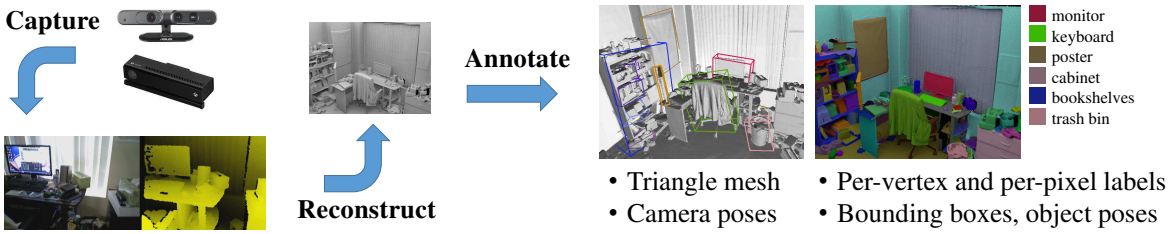


Figure 2. Overview of the processing pipeline and features of our dataset.

probably most similar to the SUN3D dataset created by Xiao et al. [36]. Their dataset contains over 250 scenes and is one of the biggest RGB-D datasets for 3D reconstruction to date. However, only a few scenes are fully annotated. In addition, their camera poses are not accurate, resulting in several misaligned surfaces (*e.g.*, providence_station, mit_w85k2, mit_w85_4). As their scenes are represented using point cloud, visibility check during rendering can be inaccurate. The annotations are done on color images via a web-based annotation tool similar to the LabelMe system [24]. The annotation results are then propagated to 3D scenes. In contrast, our scenes are reconstructed and represented as *triangle mesh*, and annotations are done efficiently in 3D. Triangle mesh allows accurate visibility check, and supports high-quality novel view rendering and relighting. We include relightable scenes in our dataset which, to the best of our knowledge, is the first of such kind available in the community.

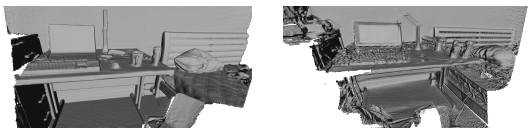
Recently, Song et al. [28] introduced an RGB-D dataset for evaluating several scene understanding algorithms. This dataset has approximately ten thousand frames and is mainly designed for evaluating object detection and recognition algorithms. Our dataset is of a larger scale and allows exploration of other applications such as intrinsic decomposition and novel view synthesis.

Handa et al. [13] created a CAD model dataset for scene understanding. While it can be extended to include an infinite number of objects, how well the CAD models and their textures resemble real-world scenes is subject to the skills of the human modellers or the performance of scene synthesis algorithms being applied. The dataset can mainly be applied for depth-based scene understanding. In addition, the complexity of real-world scenes, for example, object cluttering, is non-trivial to model. The Clutterpalette [41] could be used to add clutter with user interaction, but material and reflectance could still be difficult to model, especially for rough and textured surfaces. In contrast, our dataset is created from real-world data and each scene has both geometry and texture information, which can support different applications as we will demonstrate.

Table 1 summarizes the datasets we discuss and highlights their differences. We compare existing datasets based on the total number of frames and scenes, annotation, delivered scene format, existence of camera pose, and the designated applications of the datasets. Previous datasets only provide RGB-D images or point cloud of their scenes. Our dataset provides reconstructed triangle meshes of the scenes in addition to RGB-D images, to facilitate a wider range of applications, *e.g.*, shape completion and intrinsic decomposition in 3D. For a more comprehensive survey of publicly available datasets for other applications, we refer the readers to the excellent summary by Firman [9].

Dataset	Quantity	Annotation	Format	Pose	Applications
NYU v2 [27]	1449 frames	All	Image	N	U O
SUN RGB-D [28]	10K frames	All	Image	N	U O
RGB-D v2 [17]	17 scenes	All	Cloud	Y	R O
TUM [29]	47 scenes	N.A.	Image	Y	R
SUN3D [36]	254 scenes	8 scenes	Cloud	Y	R L
Ours	100 scenes	All	Mesh	Y	R L I S

Table 1. Comparison with existing datasets. Our dataset has more than 100 scenes densely annotated in 3D and 2D. The number of frames per scene is between 2,000 and 10,000 frames. Notation for applications: R (3D reconstruction), O (object detection), U (scene understanding), L (scene labeling), I (intrinsic decomposition), S (shape completion).



Asus Xtion PRO Kinect v2

Figure 3. Reconstruction of a room corner captured by an Asus Xtion Pro (structured light sensor) and a Kinect v2 (time-of-flight sensor), respectively. The depth images captured by the Kinect v2 have a lower pixel resolution yet the scene reconstructed from them contains more surface details. However, the Kinect v2 fails to capture depth values accurately near object boundaries, and the reconstructed mesh is noisier.

3. Reconstruction

3.1. Data Capturing

We captured the RGB-D videos of different scenes using a consumer-grade depth camera. The scenes were captured steadily, while the depth and color images were displayed in real time. Due to inaccuracies in depth values at far ranges, pixels of unreliable depth values were highlighted to inform the operator to seek more reliable depth sources. In our experiments, we set a threshold of 2.5 meters to ensure that the depth values are reliable enough for reconstruction.

Our scenes are from a variety of categories, including bedrooms, living rooms, kitchens and offices. The number of objects per scene ranges from 7 to 63. The average floor size is 22.6 square meters. We define an additional measure to estimate the area that the operator stayed when capturing a scene. We call this measure *operator coverage*. This measure is estimated by projecting the camera positions gone through when capturing a scene to the floor plane of that scene, and then computing the area of the convex hull of all projected camera positions. Using a convex hull to define this measure is appropriate since most of the scenes we captured is box-shaped. We found that the operator coverage is about 65% of the floor area in average. This is generally consistent with the setup where the operator needs to approach the surfaces to capture short-range depth values.

The average camera velocity is 0.1 meter per second, which is slow enough for camera tracking and for producing sharp color frames. The average camera angular velocity is 30 degrees per second which shows that the camera remained stable throughout the capturing process.

For convenience, we assign a unique ID to each scene. Each ID is represented by three digits, *e.g.*, 311, 322, and is used in our experiments to refer to the corresponding scene.

3.2. Sensor Comparison

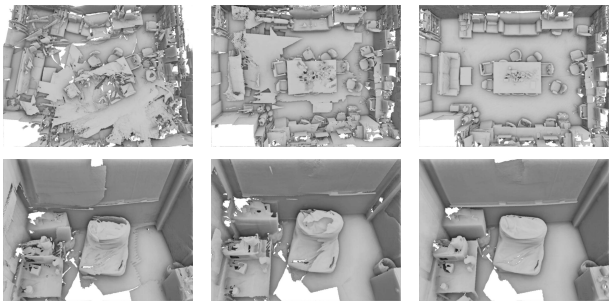
We used an Asus Xtion PRO and a second generation Microsoft Kinect (Kinect v2) for data capturing. Data from both sensors are processed using a single reconstruction and annotation pipeline. The Asus Xtion PRO is comparable to a first generation Microsoft Kinect (Kinect v1) but is more lightweight and does not require an external power supply to function. For the Asus Xtion PRO, both color and depth streams are captured with a 640×480 resolution. For the Kinect v2, the color stream is captured with a 1920×1080 resolution and the depth stream is captured with a 512×424 resolution. In addition, the aligned color images to depth camera space and the aligned depth images to color camera space are also stored. We modified the Kinect v2 to function on an external battery to enhance mobility during capturing.

Both the Asus Xtion PRO (a structured light sensor) and the Kinect v2 (a time-of-flight sensor) produce incomplete depth images. The Asus Xtion PRO suffers from data loss, *i.e.*, depth measurements along object boundaries are missing. In contrast, the Kinect v2 suffers from data drift, *i.e.*, depth measurements along object boundaries are not reliable. Figure 3 shows a comparison of the reconstruction quality of both sensors. Geometry quality from Kinect v2 data is inferior to that of Asus Xtion PRO. Therefore, we captured about 90% of the scenes in our dataset using Asus Xtion PRO. A small number of scenes are captured by Kinect v2 for further analysis and comparison.

3.3. 3D Reconstruction

There are several techniques for 3D reconstruction from RGB-D images. KinectFusion [19] and its moving volume extension [33, 23, 20] can be applied to reconstructing a scene in real time. However, such techniques do not globally optimize surface alignment. The camera pose tends to drift over time due to error accumulation, which leads to poor-quality reconstruction. In robotics, Kerl et al. [16] proposed to use keyframes and an entropy metric to detect loop closure and eliminate drift. Whelan et al. [34] proposed to detect local loop closure by model-to-model registration, and global loop closure by place recognition using fern encoding. These method runs in real time, but the reconstruction quality is inferior to offline methods.

A notable offline approach for reconstruction is to split an RGB-D sequence into segments, each containing a small



Kerl et al. [16] Whelan et al. [34] Choi et al. [6]

Figure 4. Comparison of state-of-the-art reconstruction methods. The method by Choi et al. [6] produces the most visually pleasing geometry.

number of frames. A standard RGB-D fusion approach can be applied on each segment to obtain a geometric fragment. The 3D reconstruction problem becomes how to align the fragments globally. Choi et al. [6] proposed to align the pairwise fragments and then perform an optimization based on switchable constraints to prune false alignments. Zhou and Koltun [46] proposed to deform the fragments using control lattices for nonrigid alignment. In a subsequent work [45], they proposed to represent nonrigid deformation by a camera calibration function that can be optimized much faster. These approaches might require a few hours to complete reconstruction but can produce highly accurate surface alignment.

Recently, Fioraio et al. [8] proposed an online reconstruction system that can reduce the running time to the order of minutes but still achieve good geometry quality. Xiao et al. [36] proposed to employ user interaction for segmentation and provide constraints for 3D reconstruction. However, their bundle adjustment with object bounding box constraints may produce inaccurate surface alignment.

Figure 4 shows a few typical scenes in our dataset reconstructed by the method by Kerl et al. [16], Whelan et al. [34], and Choi et al. [6]. As can be seen, the method by Choi et al. produces the most accurate surface alignment, leading to the most visually pleasing geometry. Therefore, we choose this method for reconstruction of our dataset.

The implementation details are as follows. The volume size is set to four meters, and the fragments are registered by rigid alignment using a Gauss-Newton optimization. Only depth values that are less than 2.5 meters are considered for reconstruction. In post processing, all unreferenced vertices from the output mesh are removed. In addition, to alleviate noise, connected components whose diameter is smaller than 10 cm are deleted. Finally, we modify the system to support Kinect v2 sensor. This makes it possible to reconstruct and annotate data from both Asus Xtion and Kinect v2 sensors using the same pipeline.

In our reconstruction, we use a simple camera model of

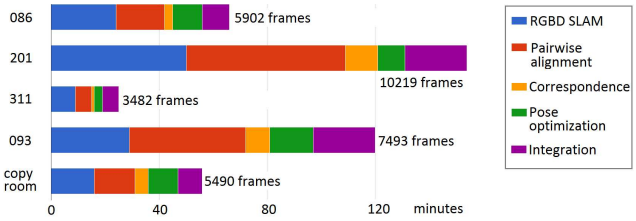


Figure 5. Running time breakdown of the approach by Choi et al. [6] applied on some of our scenes. The Copyroom scene is courtesy of Zhou et al. [44].

which the focal length is calibrated using a checkerboard; the principal points are set to the center of the image. This model yields a relatively low depth reprojection error and is convenient to use with existing graphics API like OpenGL. Experiments with more complex calibration are available in the supplementary document.

Figure 5 reports the performance of the method by Choi et al. [6] on some of our scenes. We use the time needed to reconstruct the Copyroom scene captured by Zhou et al. [44] as the baseline for comparison. As can be seen, the most time consuming stage is the RGBD SLAM and pairwise alignment. The time complexity of the pairwise alignment is almost quadratic to the number of input frames, which becomes a performance bottleneck for long input sequences.

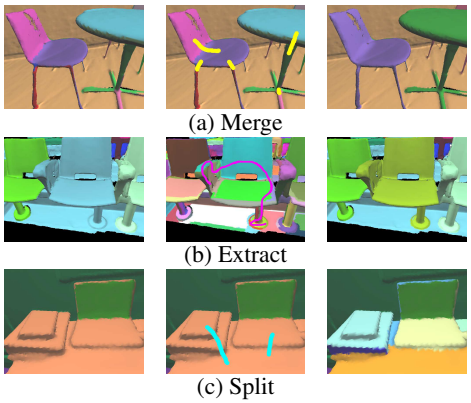
3.4. Texturing

After reconstruction, the scene meshes can be textured using color images. We implement a simple texturing approach as a baseline. To texture, we assign color to each mesh vertex. This is done by reprojecting all mesh vertices to the camera image space and recording the color distribution for each vertex. To remove noise, the median value of the color distribution is assigned as the vertex color.

Since the triangles in the mesh could be very small, several nearby vertices could fall onto a single pixel. In such cases, we should regard all such vertices as unoccluded and color them all. To do so, visibility test is performed by a soft depth test, *i.e.*, z-buffering with a small threshold to accept vertices that have very similar depth values on a pixel.

4. Annotation

Our reconstructed scenes are fully segmented and annotated. Our annotation is based on an automatic two-level segmentation, fine and coarse, followed by user interaction to fine tune the segmentation and annotation. In automatic segmentation, fine segments are obtained by applying the graph-based segmentation algorithm of Felzenszwalb et al. [7] to the mesh vertices. Coarse segments are then determined by merging the fine segments by optimizing a Markov random field. From the initial fine and coarse



(a) Merge (b) Extract (c) Split

Figure 6. Segmentation refinement. The leftmost and rightmost images illustrate the segmentation before and after applying the operation, respectively. *Merge* and *split* work on the coarse segments, while *extract* combines multiple fine segments into a coarse segment, *e.g.* in (b), the chair is extracted from the floor.

segmentation, we then employ user strokes to refine the segments to fit scene semantics. User can switch between the coarse and fine segmentation display and perform three operations: *merge*, *extract*, and *split*, to create desired segments. Figure 6 demonstrates these operations. We found that two-level segmentation combined with user strokes are robust enough to segment our scenes. After segmentation, user tags the objects with text labels. The annotation in 3D is then projected onto 2D frames, resulting in dense labelling in both 3D and 2D. More technical details of our annotation tool could be found in the technical report [32].

To facilitate scene understanding, we enrich our dataset with per-object annotation as follows:

- *Axis-aligned Bounding Box (AABB)*. We provide axis-aligned bounding box for each segmented object. The floor plane is segmented and its normal is estimated. The scene geometry is rotated such that the floor aligns with the xz-plane.

- *Oriented Bounding Box (OBB)*. For each object, the eigenvectors of its vertices are used to define an oriented bounding box which tightly encloses the object.

- *Object Pose*. Each object could be rotated so that its front direction matches the Z-axis. This transformation is particularly useful when the front direction of an object is needed, *e.g.*, furniture arrangement.

- *Object Images*. For each object, we extract color and depth images in which the object could be observed. This attribute is useful for training and testing object recognition algorithms. It also allows the creation of object datasets [31].

4.1. Annotation Transfer

While creating a scene dataset that involves geometry reconstruction and interactive annotation, sometimes the scene geometry needs to be updated after annotation. This situation arises when a scene needs to be recaptured or reconstructed using a new technique or new calibration parameters. These changes result in slightly different geometry. In such cases, it would be preferable to transfer existing annotations to the new scene. In this section, we describe a technique for automatic annotation transfer.

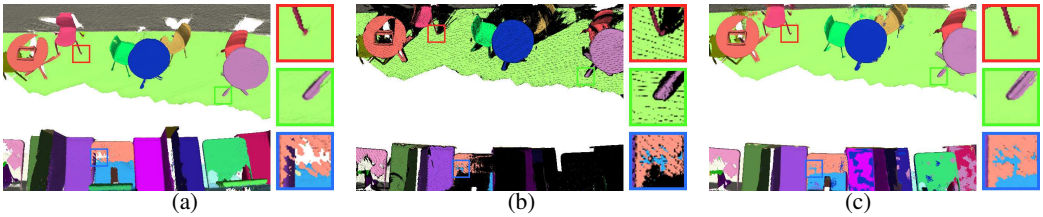
We assume that the source and target scenes share the same set of RGB-D images. This allows us to transfer the annotation by using reprojection. Our approach is as follows. First, we assign the reliable annotations to vertices. As a vertex could be seen in multiple views, we keep track of a set of candidate annotations of the vertex by projecting the vertex to 2D and collect the annotations of the corresponding pixels. The annotation that dominates more than 90% of the set is marked reliable and assigned to the vertex.

This step usually transfers a large amount of annotations from the source mesh to the target mesh. However, due to error in the reconstruction, some vertices may not have reliable annotations. This problem often occurs near object boundaries, highly occluded regions, and areas where the camera poses are not tracked accurately. We handle this problem by propagating reliable annotations to vertices that are not annotated in the previous step. The propagation is done by a nearest neighbor search. Specifically, we treat the vertices with reliable annotations as seeds. All seeds are added to a 6-dimensional kd-tree. Each leaf node contains the position and normal vector of its seed. For each vertex without an annotation, we determine its nearest seed. The annotation of the seed is then assigned to the vertex. The nearest neighbor search considers both the positions and normals of the vertices. We use $\|\mathbf{p}-\mathbf{p}_i\|+\lambda(1-\mathbf{n}^\top\mathbf{n}_i)$ to measure the distance between a query point $(\mathbf{p},\mathbf{n})$ to a tree node $(\mathbf{p}_i,\mathbf{n}_i)$, where $\mathbf{p}$ and $\mathbf{n}$ denote the position and normal vector respectively. The parameter λ is chosen based on scene scale. In our experiments, we set $\lambda=100$.

Figure 7 shows an example of annotation transfer. As can be seen, this approach can effectively transfer most of the annotations from the source scene to the target scene. In our implementation, the entire transfer process takes about two minutes to complete. After propagation, noisy or mislabeled regions could be refined by user interaction.

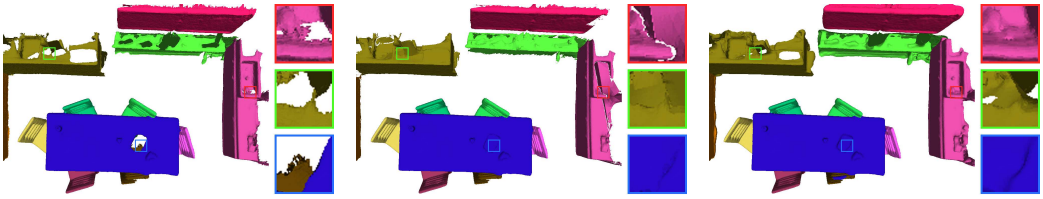
5. Proof-of-Concept Applications

The annotated scene meshes and the extracted statistics of our dataset can serve as useful inputs for multiple applications. In this section, we demonstrate how they can be used for benchmarking shape completion algorithms, scene relighting by applying intrinsic decomposition, scene syn-



(a) (b) (c)

Figure 7. Annotation transfer example. (a) Source mesh with annotations. (b) Target mesh with annotations transferred from the source mesh. Note the subtle differences in geometry. Black regions correspond to unreliable annotation transfer. (c) Target mesh after propagating annotations using kd-tree search. Please refer to supplementary document for more scenes.



Input PolyMender [14] Poisson [15]

Figure 8. Visual comparison of shape completion at scene level. Both methods can only complete small holes. Poisson reconstruction has less cracks and holes in general. Please refer to supplementary document for enlarged images and more scenes.

thesis, and novel view synthesis. We run existing algorithms with our scenes and provide insights based on the results.

5.1. Benchmarking Shape Completion Algorithms

The surface reconstructed from the depth-only pipeline may contain holes at locations where depth is not available or consistent. Shape completion is an important post-processing technique for filling holes and improving geometry quality. We describe how our dataset can be used to benchmark data-driven techniques on shape completion.

We first study how the objects could be repaired by low-level geometry processing techniques. Figure 8 shows a visual comparison of shape completion using PolyMender [14] and screened Poisson surface reconstruction [15], two popular techniques for reconstructing and repairing polygons as suggested by Attene et al. [1]. In particular, for each scene, annotated objects are extracted. Completion is performed on these objects and the resulting objects are added back to the scene. We experimented with completion on the entire mesh but found that it gives worse results than completion on individual objects.

As can been seen, PolyMender works quite well in filling backfaces and small holes. Screened Poisson reconstruction ignores the existing surfaces and solves for a new surface that best fit the input oriented point set. The results have extended surfaces which could be trimmed. However, trimming might cause holes to reappear. In general, these methods are not effective for missing object parts such as chair legs or big holes. More sophisticated approaches are required to utilize object semantics for shape completion.

A recent technique for shape completion that considers object similarity from a database is 3D ShapeNets [35]. This method fuses together CAD models from its database to create a complete object such that it is structurally similar to an input scan. Nguyen et al. [31] combine Markov random field with 3D ShapeNets to find an optimal object that is consistent to input color images.

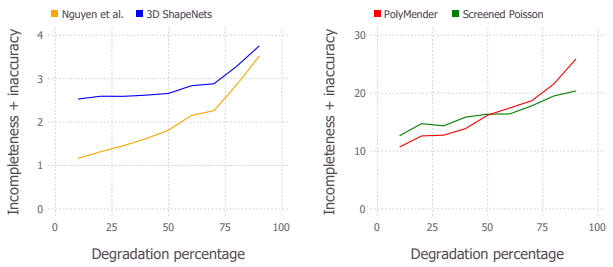
We benchmark the performances of these techniques since their implementations are publicly available. It is required that the objects are converted to volume representation and downsampled to $30 \times 30 \times 30$. This volume resolution is low because their system is designed towards scene understanding, where such resolution is sufficient.

Following the idea by Nguyen et al. [31], we treat the objects as ground truth, and add random holes to the geometry of the objects. The modified objects are completed and compared to the ground truth. We use incompleteness and inaccuracy to measure the performance of the algorithms. Figure 9 reports the benchmark results. We also include the results for PolyMender and screened Poisson reconstruction evaluated at high-resolution volume at $512 \times 512 \times 512$.

Recently, Rock et al. [22] proposed to complete an object by retrieving a similar depth map from a dataset, and then find its corresponding 3D model and deforming the model to fit the input scan. Sung et al. [30] used an object-part representation to achieve more robust completion. The distributions of object parts are learned from a CAD model dataset. It would be interesting to apply these techniques to our dataset for more benchmark results.

5.2. Scene Relighting

Novel lighting could be generated for the scenes in our dataset. To create relightable scenes, we use intrinsic decomposition to obtain the reflectance map of the mesh. Un-



(a) (b)

Figure 9. Benchmark of shape completion algorithms for objects. The methods in (a) and (b) use a $30 \times 30 \times 30$ and $512 \times 512 \times 512$ volume representation, respectively.

fortunately, the state-of-the-art methods are designed for images, and while theoretically the extension to mesh surfaces is possible, there is no such available implementation yet. Therefore, we opt to implement a simple intrinsic decomposition technique to demonstrate this application.

We apply the Retinex method for mesh vertices which is one of the simplest but effective decomposition algorithm [3]. The variant of the Retinex algorithms that we use is described as follows. Given the texture I, the shading component S can be optimized by minimizing:

$$E(S) = \sum_i \sum_{j \in N(i)} (S_i - S_j)^2 + \omega_{ij} \left((I_i - I_j) - (S_i - S_j) \right)^2, \quad (1)$$

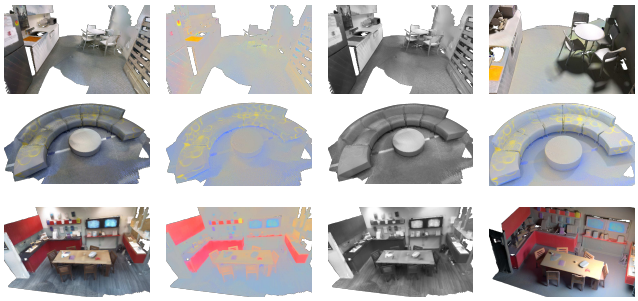
where i, j refer to vertex indices, $N(\cdot)$ the neighbor query operator.

The weight ω_{ij} is computed based on the chromaticity difference between two neighbor vertices. It is set to a large value if the chromaticity values are very similar, and thus penalizes any local reflectance changes. The weight therefore works as a classifier that distributes edges into reflectance and shading components.

This cost function is quadratic. Zhao et al. [42] showed that the shading S is the solution of the linear system $AS = b$, where A is a positive definite matrix, $A_{ii} = \sum_j (1 + \omega_{ij})$ and $A_{ij} = -(1 + \omega_{ij})$, and $b_i = \sum_j \omega_{ij} (I_i - I_j)$, where $j \in N(i)$.

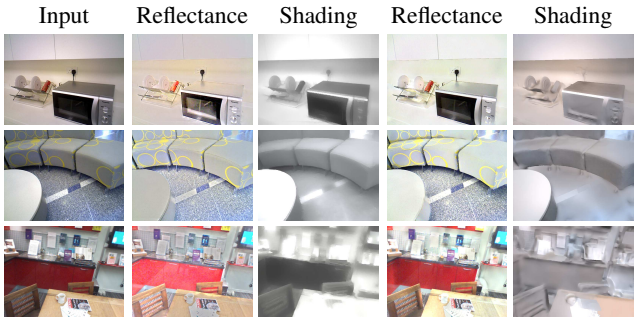
Figure 10 shows the results of our intrinsic decomposition approach. We implemented our approach in C++, and it takes less than 10 seconds to process a mesh with two million vertices on a laptop equipped with an Intel Core i7 CPU and 16 GB of RAM. In our examples, the relighting result shows that the Retinex method works well and the reflectance is qualitatively good.

Since there is no publicly available implementation of intrinsic decomposition algorithms on mesh, we can only evaluate state-of-the-art methods with a subset of RGB-D images with complex materials, illumination, and occlusion from our dataset. Figure 11 visually compares the results



Input Reflectance Shading Relighting

Figure 10. Intrinsic decomposition using the Retinex method applied on the mesh texture and the relighting results.



Chen and Koltun [4] Barron and Malik [2]

Figure 11. Visual comparison of the state-of-the-art intrinsic decomposition algorithms on RGB-D images. Qualitatively, the reflectance by Chen and Koltun [4] and the shading by Barron and Malik [2] closely resembles the reflectance and shading observed in the input images.

produced by the state-of-the-art methods.

In our experiments, the method by Chen and Koltun [4] takes about 10 to 20 minutes per frame, while the method by Barron and Malik [2] takes from 1 to 2 hours per frame including preprocessing. Recently, Hachama et al. [12] proposed a technique to perform intrinsic decomposition for multiple RGD-D frames. It is interesting to apply their method to our data once their system is available.

5.3. Scene Synthesis by Extracted Statistics

Some useful scene statistics could be extracted from our datasets for novel applications such as scene synthesis and furniture arrangement. For example, object occurrence and co-occurrence probabilities can be extracted, which are commonly used for indoor scene synthesis [10, 39, 40]. In the previous scene synthesis approaches, object distribution statistics are extracted from synthetic scenes created by CAD modeling software. These synthetic scenes are not only tedious to create but also unrealistic compared to real-world scenes. Scene synthesis approaches can easily make use of the realistic object distribution statistics extracted from the scenes in our dataset.

To illustrate, we show the object co-occurrence matrix of

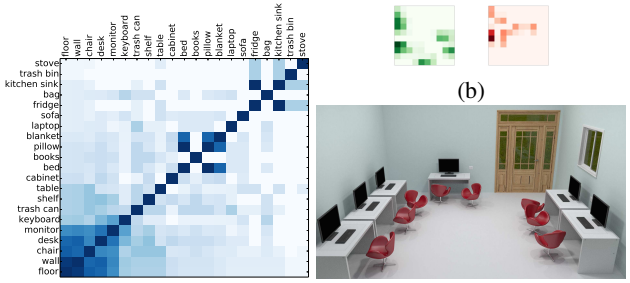


Figure 12. (a) Object co-occurrence of the 22 most common object classes in our dataset. (b) Object placement probabilities of desks (green) and chairs (red) computed from our scenes. Darker colors correspond to higher probability. The horizontal and vertical axis correspond to the X-axis and Z-axis in the world space. (c) A synthesized scene based on the statistics.

the scenes in Figure 12(a). From this matrix, common pairwise or group relationships between objects can be detected, which is called *structural groups* by Xu et al. [38]. We also compute *object placement probabilities* used by Chen et al. [5] for scene synthesis, as shown in Figure 12(b). We first transform our scenes into a common coordinate system, where the y-axis points upwards, and the x-axis and z-axis are aligned to walls (if any). Then the bounding box of the entire scene is divided into a 10×10 grid before computing the probabilities. We used 20 scenes in total to compute the statistics. Note that only scenes of the same category are used to compute the statistics.

Figure 12(c) shows a synthesized workplace scene based on the extracted statistics. The dominant objects in the scene includes desks, chairs, monitors and keyboards, which match the object co-occurrence statistics. The probability maps in Figure 12(b) are used to arrange the desks and chairs. As the object placement probability cannot avoid object overlap, we verify object bounding box intersections to discard overlap object placements.

5.4. Novel View Synthesis

Our 3D scene dataset can also be used for 2D applications. For example, it is possible to synthesize new views that are hard to capture in the real world. Figure 13 shows two examples. In the bedroom scene, images can be rendered with a virtual camera looking from the ceiling. In the kitchen scene, images with a wide field of view can be rendered.

The color and depth images synthesized at novel viewpoints can potentially be used to boost scene recognition capability, to recognize scene images taken from these viewpoints. Quattoni and Torralba [21] proposed a method to recognize indoor scenes from color images. Zhou et al. [43] proposed the Places dataset and a deep learning technique to extract features of both indoor and outdoor scenes. The recognition is only trained with photographs from the Internet. Our dataset could be used to enrich the training data for these methods.

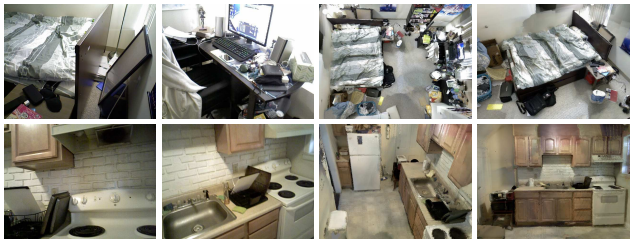


Figure 13. Novel views generated from a bedroom scene and a kitchen scene. Empty region which has no mesh surface is inpainted. For the kitchen at the lower right, the white fridge is removed from the scene to render a non-occluded front view of the kitchen storage and the sink.

6. Conclusion

This paper presents a new dataset of annotated scene meshes. Coupled with this dataset is a rich set of features that aim to facilitate the study of several practical applications such as benchmarking shape completion algorithms, scene relighting, scene synthesis and novel view synthesis.

We aim to expand this dataset by releasing our tools to the community. Our scenes could also be used for applications such as relocalization [26]. The idea of sparse reflectance [25] could also be exploited to improve intrinsic decomposition. We are also interested in extending our Retinex implementation to the model proposed by Chen and Koltun [4] as both leverage least squares optimization.

Acknowledgement. We thank Fangyu Lin for his assistance with the data capture. We also thank Shotton et al. [26] for the Kitchen scene in Figure 8, 10, 11, and BlenderSwap for the models in Figure 12. Lap-Fai Yu is supported by the University of Massachusetts Boston StartUp Grant P20150000029280 and by the Joseph P. Healey Research Grant Program provided by the Office of the Vice Provost for Research and Strategic Initiatives & Dean of Graduate Studies of the University of Massachusetts Boston. This research is supported by the National Science Foundation under award number 1565978. We also acknowledge NVIDIA Corporation for graphics card donation. Sai-Kit Yeung is supported by Singapore MOE Academic Research Fund MOE2013-T2-1-159 and SUTD-MIT International Design Center Grant IDG31300106. We acknowledge the support of the SUTD Digital Manufacturing and Design (DManD) Centre which is supported by the National Research Foundation (NRF) of Singapore. This research is also supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative.

References

[1] M. Attene, M. Campen, and L. Kobbelt. Polygon mesh repairing: An application perspective. *ACM Computing Surveys*, 45(2):15:1–15:33, Mar. 2013. 6

[2] J. T. Barron and J. Malik. Intrinsic scene properties from a single rgb-d image. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 7

[3] S. Bell, K. Bala, and N. Snavely. Intrinsic images in the wild. *ACM Transactions on Graphics (TOG)*, 33(4), 2014. 7

[4] Q. Chen and V. Koltun. A simple model for intrinsic image decomposition with depth cues. In *The IEEE International Conference on Computer Vision (ICCV)*, 2013. 7, 8

[5] X. Chen, J. Li, Q. Li, B. Gao, D. Zou, and Q. Zhao. Image2scene: Transforming style of 3d room. In *The ACM Conference on Multimedia Conference (MM)*, 2015. 8

[6] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, Sept. 2004. 4

[8] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi. Large-scale and drift-free surface reconstruction using online subvolume registration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4

[9] M. Firman. RGBD Datasets: Past, Present and Future. In *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*, 2016. 2

[10] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan. Example-based synthesis of 3d object arrangements. In *ACM SIGGRAPH Asia*, 2012. 7

[11] R. Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 1

[12] M. Hachama, B. Ghanem, and P. Wonka. Intrinsic scene decomposition from rgb-d images. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 7

[13] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla. Understanding real world indoor scenes with synthetic data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[14] T. Ju. Robust repair of polygonal models. In *ACM Transactions on Graphics (TOG)*, pages 888–895, New York, NY, USA, 2004. ACM. 6

[15] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29:1–29:13, July 2013. 6

[16] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *The International Conference on Intelligent Robot Systems (IROS)*, 2013. 3, 4

[17] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *The IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 3

[18] T. Narihira, M. Maire, and S. X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 1

[19] 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 *The IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. 3

[20] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 2013. 3

[21] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 8

[22] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem. Completing 3d object shape from one depth image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 6

[23] H. Roth and M. Vona. Moving volume kinectfusion. In *British Machine Vision Conference*, 2012. 3

[24] 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 (IJCV)*, 77(1):157–173, 2007. 2

[25] L. Shen, C. Yeo, and B.-S. Hua. Intrinsic image decomposition using a sparse representation of reflectance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Dec 2013. 8

[26] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 8

[27] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *The European Conference on Computer Vision (ECCV)*, 2012. 1, 3

[28] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2, 3

[29] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *The International Conference on Intelligent Robot Systems (IROS)*, 2012. 3

[30] M. Sung, V. G. Kim, R. Angst, and L. Guibas. Data-driven structural priors for shape completion. *ACM Transactions on Graphics*, 34(6), Oct. 2015. 6

[31] D. Thanh Nguyen, B.-S. Hua, K. Tran, Q.-H. Pham, and S.-K. Yeung. A field model for repairing 3d shapes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5, 6

[32] D. Thanh Nguyen, B.-S. Hua, L.-F. Yu, and S.-K. Yeung. A robust 3d-2d interactive tool for scene segmentation and annotation. *Computing Research Repository (CoRR)*, 2016. 5

[33] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012. 3

[34] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems (RSS)*, 2015. 3, 4

[35] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 6

[36] J. Xiao, A. Owens, and A. Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *The IEEE International Conference on Computer Vision (ICCV)*, 2013. 2, 3, 4

[37] S. Xie and Z. Tu. Holistically-nested edge detection. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 1

[38] K. Xu, K. Chen, H. Fu, W.-L. Sun, and S.-M. Hu. Sketch2scene: Sketch-based co-retrieval and co-placement of 3d models. *ACM Transactions on Graphics (TOG)*, 32(4):123:1–123:12, 2013. 8

[39] Y.-T. Yeh, L. Yang, M. Watson, N. D. Goodman, and P. Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Transactions on Graphics (TOG)*, 31(4):56:1–56:11, July 2012. 7

[40] L.-F. Yu, S. K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. Osher. Make it home: automatic optimization of furniture arrangement. *ACM Transactions on Graphics (TOG)*, 30(4):86, 2011. 7

[41] L.-F. Yu, S.-K. Yeung, and D. Terzopoulos. The clutterpalette: An interactive tool for detailing indoor scenes. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2015. 2

[42] Q. Zhao, P. Tan, Q. Dai, L. Shen, E. Wu, and S. Lin. A closed-form solution to retinex with nonlocal texture constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, July 2012. 7

[43] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 8

[44] Q. Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics*, 32(4):112:1–112:8, 2013. 4

[45] Q. Y. Zhou and V. Koltun. Simultaneous localization and calibration: Self-calibration of consumer depth cameras. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 4

[46] Q.-Y. Zhou, S. Miller, and V. Koltun. Elastic fragments for dense scene reconstruction. In *The IEEE International Conference on Computer Vision (ICCV)*, 2013. 4

[47] T. Zhou, P. Krahenbuhl, and A. A. Efros. Learning data-driven reflectance priors for intrinsic image decomposition. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 1