

Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks

Yinda Zhang^{†*} Shuran Song^{†*} Ersin Yumer[‡] Manolis Savva[†]
 Joon-Young Lee[‡] Hailin Jin[‡] Thomas Funkhouser[†]

[†]Princeton University

[‡]Adobe Research

Abstract

Indoor scene understanding is central to applications such as robot navigation and human companion assistance. Over the last years, data-driven deep neural networks have outperformed many traditional approaches thanks to their representation learning capabilities. One of the bottlenecks in training for better representations is the amount of available per-pixel ground truth data that is required for core scene understanding tasks such as semantic segmentation, normal prediction, and object boundary detection. To address this problem, a number of works proposed using synthetic data. However, a systematic study of how such synthetic data is generated is missing. In this work, we introduce a large-scale synthetic dataset with 500K physically-based rendered images from 45K realistic 3D indoor scenes. We study the effects of rendering methods and scene lighting on training for three computer vision tasks: surface normal prediction, semantic segmentation, and object boundary detection. This study provides insights into the best practices for training with synthetic data (more realistic rendering is worth it) and shows that pretraining with our new synthetic dataset can improve results beyond the current state of the art on all three tasks.

1. Introduction

Indoor scene understanding is crucial to many applications including but not limited to robotic agent path planning, assistive human companions, and monitoring systems. One of the most promising approaches to tackle these issues is using a data-driven method, where the representation is learned from large amount of data. However, real world data is very limited for most of these tasks, such as the widely used indoor RGBD dataset for normal prediction introduced by Silberman *et al.* [21], which contains merely 1449 images. Such datasets are not trivial to collect due to various requirements such as depth sensing tech-

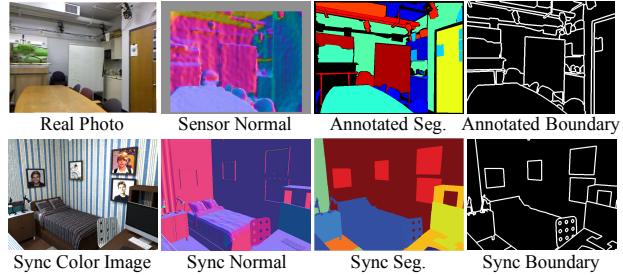


Figure 1. Real data (top) vs. synthetic data (bottom). For the real data, note the noise in normal map and the diminishing accuracy at object boundaries in the semantic labels.

nology [21, 23] and excessive human effort for semantic segmentation [14, 8]. Moreover, current datasets lack pixel level accuracy due to sensor noise or labeling error (Fig. 1).

This has recently led to utilizing synthetic data in the form of 2D render pairs (RGB image and per-pixel label map) from digital 3D models [2, 6, 11, 30, 24, 17]. However, there are two major problems that have not been addressed: (1) studies of how indoor scene context affect training have not been possible due to the lack of large scene datasets, so training is performed mostly on repositories with independent 3D objects [4]; and (2) systematic studies have not been done on how such data should be rendered; unrealistic rendering methods often are used in the interest of efficiency.

To address these problems, we introduce a large scale (500K images) synthetic dataset that is created from 45K 3D houses designed by humans [20]. Using such realistic indoor 3D environments enable us to create 2D images for training in realistic context settings where support constructs (e.g. such as walls, ceilings, windows) as well as light sources exist together with common household objects. Since we have access to the source 3D models, we can generate dense per-pixel training data for all tasks, virtually with no cost.

Complete control over the 3D scenes enables us to systematically manipulate both outdoor and indoor lighting,

* indicates equal contributions.

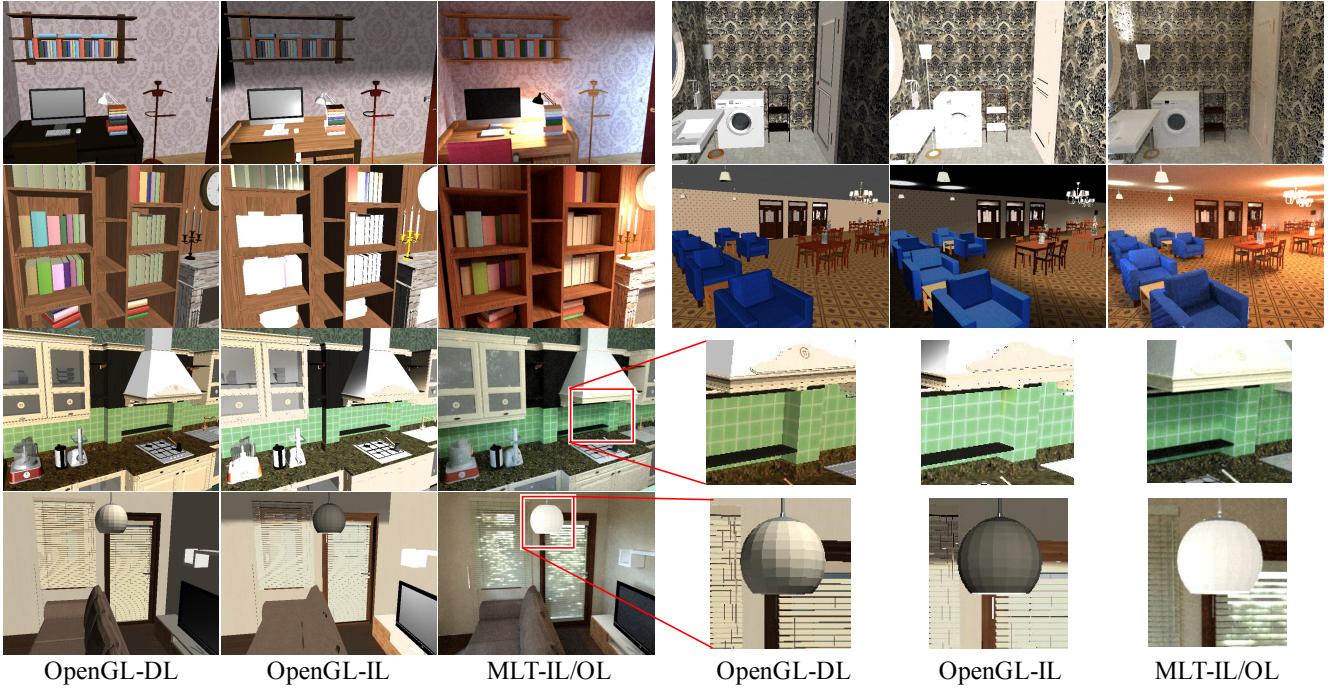


Figure 2. Render output examples with OPENGL-DL, OPENGL-IL, and MLT-IL/OL. The physically based rendering with proper illumination provides the best rendering quality with soft shadow and realistic material, highlighted in the zoomed in view. First two rows show four typical examples in our dataset, last two rows show two examples with zoomed in views.

sample as many camera viewpoints as required, use the shapes in-context or out-of-context, and render with either simple shading methods, or physically based based rendering. For three indoor scene understanding tasks, namely normal prediction, semantic segmentation, and object edge detection, we study how different lighting conditions, rendering methods, and object context effects performance.

We use our data to train deep convolutional neural networks for per-pixel prediction of semantic segmentation, normal prediction, and object boundary prediction, followed by finetuning on real data. Our experiments show that for all three indoor scene understanding tasks, we improve over the state of the art performance. We also demonstrate that physically based rendering with realistic lighting and soft shadows (which is not possible without context) is superior to other rendering methods.

In summary, our main contributions are as follows:

- We introduce a dataset with 500K synthetic image instances where each instance consists of three image renders with varying render quality, per-pixel accurate normal map, semantic labels and object boundaries. The dataset will be released.
- We demonstrate how different rendering methods effect normal, segmentation, and edge prediction tasks. We study the effect of object context, lighting and rendering methodology on performance.
- We provide pretrained networks that achieve the state of the art on all of the three indoor scene understanding tasks after fine-tuning.

2. Background

Using synthetic data to increase the data density and diversity for deep neural network training has shown promising results. To date, synthetic data have been utilized to generate training data for predicting object pose [24, 17, 9], optical flow [6], semantic segmentation [12, 11, 30, 18], and investigating object features [2, 13].

Su *et al.* [24] used individual objects rendered in front of arbitrary backgrounds with prescribed angles relative to the camera to generate data for learning to predict object pose. Similarly, Dosovitskiy *et al.* [6] used individual objects rendered with arbitrary motion to generate synthetic motion data for learning to predict optical flow. Both works used unrealistic OpenGL rendering with fixed lights, where physically based effects such as shadows, reflections were not taken into account. Movshovitz *et al.* [17] used environment map lighting and showed that it benefits pose estimation. However, since individual objects are rendered in front of arbitrary 2D backgrounds, the data generated for these approaches lack correct 3D illumination effects due to their surroundings such as shadows and reflections from nearby objects with different materials. Moreover, they also lack realistic context for the object under consideration.

Handa *et al.* [12, 11] introduced a laboriously created 3D scene dataset and demonstrated the usage on semantic segmentation training. However, their data consisted rooms on the order of tens, which has significantly limited variation in context compared to our dataset with 45K realistic house

layouts. Moreover, their dataset has no RGB images due to lack of colors and surface materials in their scene descriptions, hence they were only able to generate depth channels. Zhang *et al.* [30] proposed to replace objects in depth images with 3D models from ShapeNet [4]. However, there is no guarantee whether replacements will be oriented correctly with respect to surrounding objects or be stylistically in context. In contrast, we take advantage of a large repository of indoor scenes created by human, which guarantees the data diversity, quality, and context relevance.

Xiang *et al.* [27] introduced a 3D object-2D image database, where 3D objects are manually aligned to 2D images. The image provides context, however the 3D data contains only the object without room structures, it is not possible to extract per-pixel ground truth for the full scene. The dataset is also limited with the number of images provided (90K). In contrast, we can provide as many (rendered image, per-pixel ground truth) pairs as one wants.

Recently, Richter *et al.* [18] demonstrated collecting synthetic data from realistic game engine by intercepting the communication between game and the graphics hardware. They showed that the data collected can be used for semantic segmentation task. Their method ensures as much context as there is in the game (Although it is limited to only outdoor context, similar to the SYNTHIA [19] dataset). However they largely reduced the human labor in annotation by tracing geometric entities across frames, the ground truth (i.e. per-pixel semantic label) collection process is not completely automated and error prone due to the human interaction: even though they track geometry through frames and propagate most of the labels, a person needs to label new objects emerging in the recorded synthetic video. Moreover, it is not trivial to alter camera view, light positions and intensity, or rendering method due to lack of access to low level constructs in the scene. On the other hand, our data and label generation process is automated, and we have full control over how the scene is lit and rendered.

3. Data

We modify the 3D scene models from the SUNCG dataset [20] to generate synthetic data. In SUNCG, there are 45,622 scenes with over 5M instances of 2644 unique objects in 84 object categories. The object models provide surface materials, including reflectance, texture, and transparency, which are used to obtain photo-realistic renderings. One of the important aspects of this dataset is the fact that the indoor layouts, furniture/object alignment, and surface materials are designed by people to replicate existing settings. However, these raw 3D models lack sufficiently accurate geometry (e.g. solid walls) and materials (e.g. emissive surfaces for lighting) for physically based rendering. We fix these problems, and release the accurate full 3D scene models ready for rendering on our [project webpage](#).



Figure 3. Typical camera samples in our dataset, and corresponding images rendered from these viewpoints.

3.1. Camera Sampling

For each scene, we select a set of cameras with a process that seeks a diverse set of views seeing many objects in context. Our process starts by selecting the “best” camera for each of six horizontal view direction sectors in every room. For each of the six views, we sample a dense set of cameras on a 2D grid with 0.25 resolution, choosing a random viewpoint within each grid cell, a random horizontal view direction within the 60 degree sector, a random height 1.5-1.6m above the floor, and a downward tilt angle of 11 degrees, while excluding viewpoints within 10cm of any obstacle to simulate typical human viewing conditions. For each of these cameras, we render an item buffer and count the number of pixels covered by each visible “object” in the image (everything except wall, ceiling, and floor). For each view direction in each room, we select the view with the highest pixel coverage, as long it has at least three different visible objects each covering at least 1% of the pixels. This process yields $6N$ candidate cameras for N rooms. Figure 3 shows the cameras sampled from an example house.

3.2. Image Rendering

We render images from these selected cameras using four combinations of rendering algorithms and lighting conditions, ranging from fast/unrealistic rendering with directional lights using the OpenGL pipeline to physically-based rendering with local lights using Mitsuba.

OpenGL with Directional Lights (OPENGL-DL). Our first method renders images with the OpenGL pipeline. The scene is illuminated with three lights: a single directional headlight pointing along the camera view direction and two directional lights pointing in nearly opposite diagonal directions with respect to the scene. No local illumination, shadows, or indirect illumination is included.

OpenGL with Indoor Lights (OPENGL-IL). Our second method also uses the OpenGL pipeline. However,

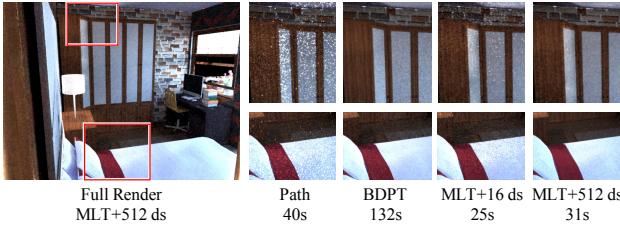


Figure 4. Quality and running time of different rendering techniques. Path tracing does not converge well and introduces white dot artifacts. Bidirectional path tracing works well but is very slow. Metropolis Light Transport (MLT) with low sampler rate for direct illumination still occasionally introduces white dot artifacts. We take MLT with high sampler rate for direct illumination.

the scene is augmented with local lights approximating the emission of indoor lighting appliances. For each object emitting light, we create a set of OpenGL point lights and spot lights approximating its emission patterns. We then render the scene with these lights enabled (choosing the best 8 lights sources for each object based on illumination intensity), and no shadows or indirect illumination is included.

Physically Based Rendering with Outdoor Lights (MLT-OL). Our third method replicates the physics of correct lighting as much as possible to generate photo-realistic rendering. In order to do so, we setup outdoor illumination which is in the form of an environment mapping with real high-definition spherical sky panoramas. The environment map that replicates outdoor lighting is cast through windows and contributes to the indoor lighting naturally. All windows are set as fully transparent to prevent artifacts on glasses and facilitate the outdoor lights to pass through. Person and plant are removed from the scene as the models are not realistic. The default wall texture is set as purely white. We use Mitsuba [1] for physically based rendering. We use Path Space Metropolis Light Transport (MLT) integrator [26] since it handles complicate structure and materials more efficiently. A comparison of rendering quality versus time with different integrators is shown in Figure 4. We can see that MLT integrator with direct illumination sampler rate 512 produces almost artifact-free renderings with affordable computation time. All the materials are set as two-sided to prevent flipped surface normal.

The images rendered using raw models from SUNCG show severe light leakage in room corners. The reason is that the walls, floors, and ceilings are represented by single planar surfaces so light rays can pass through at boundaries. We fix this problem by assigning walls with thickness (10cm in our experiments) such that each wall is represented by two surfaces. We also force the connecting walls to solidly intersect with each other to prevent light leakage caused by floating number accuracy problems during the rendering.

Physically Based Rendering with Indoor Lights (MLT-IL/OL). We also setup indoor illumination for light resulting from lighting appliances in the scene. However, the 3D dataset is labeled at the object level (e.g. lamp), and the specific *light generating* parts (e.g. bulb) is unknown. Therefore, we manually labeled all *light generating* parts of objects in order to generate correct indoor lighting. For light appliances that do not have a bulb, representing geometry in cases where bulb is deemed to be not seen, we manually added a spherical bulb geometry at the proper location. The bulb geometries of the lighting appliances are set as area emitter to work as indoor lights. Similar to the outdoor lighting, we use Mitsuba and MLT integrator for physically based indoor lights. Figure 2 shows several examples of images generated by different rendering techniques under the same camera. We can see, especially from the zoomed in view, that MLT-IL/OL produces soft shadow and natural looking materials.

3.3. Image Selection

The final step of our image synthesis pipeline is to select a subset of images to use for training. Ideally, each of the images in our synthetic training set will be similar to ones found in a test set (e.g., NYUv2). However not all of them are good due to insufficient lighting or atypical distributions of depths (e.g., occlusion by a close-up object). We perform a selection procedure to keep only the images that are similar to those in NYUv2 dataset in terms of color and depth distribution. Specifically, we first compute a normalized color histogram for each real image in the NYUv2 dataset. For each image rendered by MLT-IL/OL, we also get the normalized color histograms and calculate the histogram similarity with those from NYUv2 as the sum of minimal value of each bin (Figure 5). Then for each synthesized image, we assign it the largest similarity compared with all NYUv2 images as the score and do the same for the depth channel. Finally, we select all the images with color score and depth score both larger than 0.70. This process selects 568,793 images from the original 779,342 rendered images. Those images form our synthetic training set, and is referred as **MLT** in the latter part of this paper.

3.4. Ground Truth Generation

We generate per-pixel ground truth images encoding surface normal, semantic segmentation, and object boundary for each image. Since we have the full 3D model and camera viewpoints, generating these ground images can be done via rendering with OpenGL (e.g., with an item buffer).

4. Indoor Scene Understanding Tasks

We investigate three fundamental scene understanding tasks: (1) surface normal estimation, (2) semantic segmentation, and (3) object boundary detection. For all tasks we show how our method and synthetic data compares with

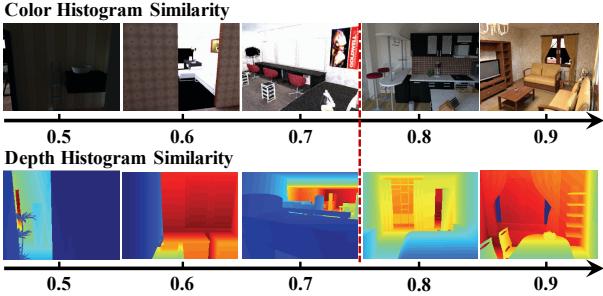


Figure 5. Histogram similarity between synthetic data and real data from NYUv2, based on which we do the image selection.

state of the art works in the literature. Specifically, we compare with Eigen *et al.* [7] for normal estimation, with Long *et al.* [15] and Yu *et al.* [29] for semantic segmentation, and with Xie *et al.* [28] for object boundary detection. We perform these comparisons systematically using different rendering conditions introduced in Section 3. In addition, for normal estimation, we also add *object without context* rendering, which allows us to investigate the importance of context when using synthetic data as well.

4.1. Normal Estimation

Method. We utilize a fully convolutional network [15] (FCN) with skip-layers for normal estimation, by combining multi-scale feature maps in VGG-16 network [22] to perform normal estimation. Specifically, the front-end encoder remains the same as conv1-conv5 in VGG-16, and the decoder is symmetric to the encoder with convolution and unpooling layers. To generate high resolution results and alleviate the vanishing gradient problems, we use skip links between each pair of corresponding convolution layers in downstream and upstream parts of the network. To further compensate the loss of spatial information with max pooling, the network remembers pooling switches in downstream, and uses them as unpooling switches at upstream in the corresponding layer. We use the inverse of the dot product between the ground truth and the estimation as loss function similar to Eigen *et al.* [7]

Object without Context. To facilitate a systematic comparison with object-centric synthetic data, where correct context is missing, we use shapes from ShapeNet[4], in addition to the rendering methodologies introduced in Sec. 3.2. We randomly pick 3500 models from furniture related categories (e.g. bed, chair, cabinet, etc.) and set up 20 cameras from randomly chosen distances and viewing directions. More specifically, we place the model at the center of a 3D sphere and uniformly sample 162 points on the sphere by subdividing it into faces of an icosahedron. For each camera a random vertex of the icosahedron is selected. This point defines a vector together with the sphere center. The camera is placed at a random distance from the center between $1.5 \times$ to $4.5 \times$ of object bounding box diagonal, and points towards the center.

Training. We directly pretrain on our synthetic data, followed by finetuning on NYUv2 similar to Bansal *et al.* [3]. We use RMSprop [25] to train our network. The learning rate is set as 1×10^{-3} , reducing to half every $300K$ iterations for the pretraining; and 1×10^{-4} reducing to half every $10K$ iterations for finetuning. The color image is zero-centered by subtracting 128. We use the procedure provided by [21] to generate the ground truth surface normals on NYUv2 as it provides more local details resulting in more realistic shape representation compared to others [16]. The ground truth also provides a score for each pixel indicating if the normal converted from local depth is reliable. We use only reliable pixels during the training.

Experiments. We conduct normal estimation experiments on NYUv2 with different training protocols. First, we directly train on NYUv2. Then we pretrain on various of MLT and OpenGL render settings respectively and finetune on NYUv2. Table 1 shows the performance. We can see that:

- The model pretrained on MLT and finetuned on NYUv2 (the last row) achieves the best performance, which outperforms the state of the art.
- Without finetuning, pretrained model on MLT significantly outperforms model pretrained on OpenGL based rendering and achieves similar performance with the model directly trained on NYUv2. This shows that physically based rendering with correct illumination is essential to encode useful information for normal prediction task.
- The model trained with images after image selection achieves better performance than using all rendered images, which demonstrates that good quality of training image is important for the pretraining.
- The MLT with both indoor and outdoor lighting significantly outperforms the case with only outdoor lighting, which suggests the importance of indoor lighting.

Figure 6 shows visual results for normal estimation on NYUv2 test split. We can see that the result from the model pretrained on MLT rendering provides sharper edges and more local details compared to the one from the model further finetuned on NYUv2, which is presumably because of the overly smoothed and noisy ground truth. Figure 6 last-column visualizes the angular error of our result compared to the ground truth, and we can see that a significant portion of the error concentrates on the walls, where our purely flat prediction is a better representation of wall normals. On the other hand, the ground truth shows significant deviation from the correct normal map. Based on this observation, we highlight the importance of high quality of ground truth. It is clear that training on synthetic data helps our model outperform and correct the NYUv2 ground truth data at certain regions such as large flat areas.

| Pre-Train | Finetune | Selection | Mean ($^{\circ}$) \downarrow | Median($^{\circ}$) \downarrow | 11.25 $^{\circ}$ (%) \uparrow | 22.5 $^{\circ}$ (%) \uparrow | 30 $^{\circ}$ (%) \uparrow |
|-------------------------|----------|-----------|----------------------------------|-----------------------------------|---------------------------------|--------------------------------|------------------------------|
| Eigen <i>et al.</i> [7] | | | 22.2 | 15.3 | 38.6 | 64.0 | 73.9 |
| NYUv2 | | | 27.30 | 21.12 | 27.21 | 52.61 | 64.72 |
| MLT Object | - | - | 48.78 | 47.49 | 3.56 | 12.79 | 21.35 |
| MLT-OL | - | No | 49.33 | 42.30 | 7.47 | 23.24 | 34.09 |
| MLT-IL/OL | - | No | 28.82 | 22.66 | 24.08 | 49.70 | 61.52 |
| MLT-IL/OL | - | Yes | 27.90 | 21.29 | 26.76 | 52.21 | 63.75 |
| OPENGL-DL | - | Yes | 34.02 | 28.00 | 18.56 | 41.14 | 52.90 |
| OPENGL-IL | - | Yes | 33.06 | 26.68 | 20.89 | 43.46 | 54.66 |
| OPENGL-IL | NYUv2 | Yes | 23.38 | 16.12 | 35.98 | 62.93 | 73.17 |
| MLT-IL/OL | NYUv2 | Yes | 21.74 | 14.75 | 39.37 | 66.25 | 76.06 |

Table 1. Performance of Normal Estimation on NYUv2 with different training protocols. The first three column lists the dataset for pretraining and finetuning, and if image selection is done. The evaluation metrics are mean and median of angular error, and percentage of pixels with error smaller than 11.25 $^{\circ}$, 22.5 $^{\circ}$, and 30 $^{\circ}$.

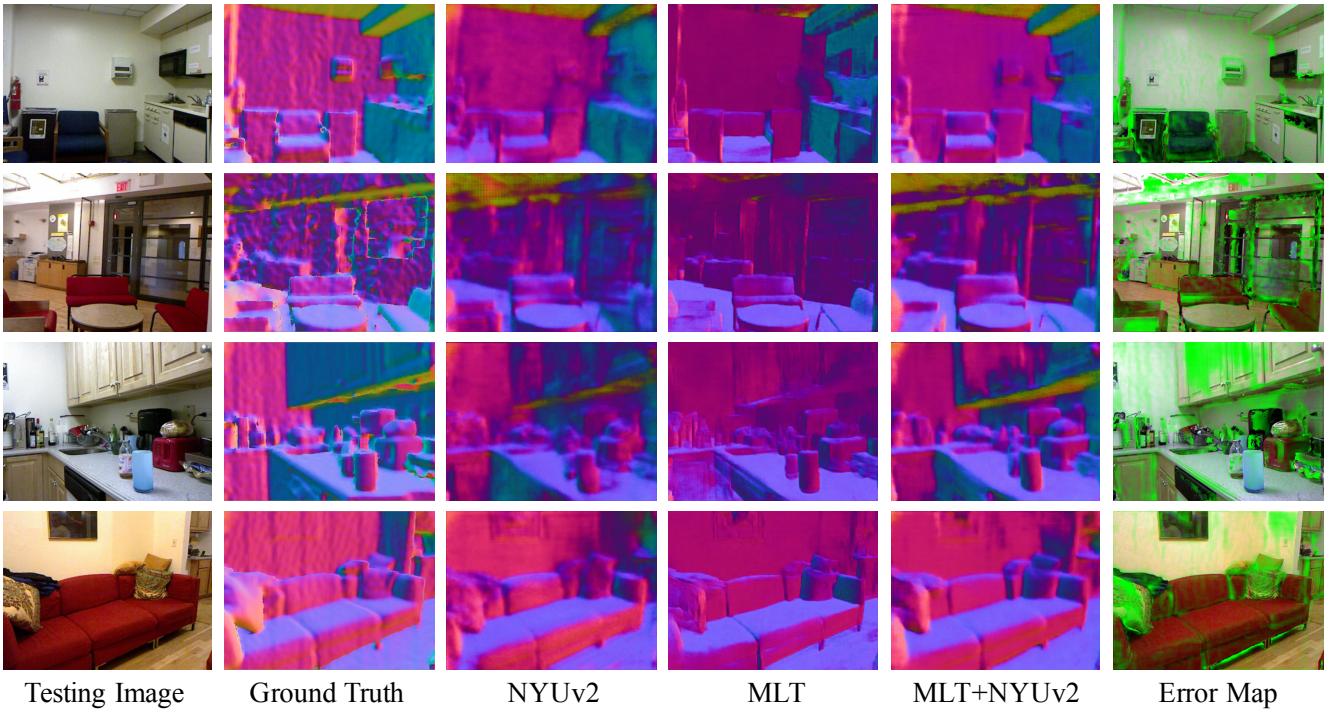


Figure 6. **Normal estimation results.** The pretrained model on MLT provides more local details, and model further finetuned on NYUv2 provides the best performance. The last column shows color image overlaid with angular error map. We can see a considerable amount of error happens on wall where ground truth is noisy.

4.2. Semantic Segmentation

Method. We use the network model proposed in [29] for semantic segmentation. The network structure is adopted from the VGG-16 network [22], however using dilated convolution layers to encode context information, which achieves better performance than [15] on NYUv2 in our experiments. We initialize the weights using the VGG-16 network [22] trained on ImageNet classification task using the procedure described in [29]. We evaluate on the same 40 semantic classes as [10].

Training. To use synthetic data for pretraining, we map our synthetic ground truth labels to the appropriate class

name in these 40 classes (note that some categories do not present in our synthetic data). We first initialize the network with pretrained weights from ImageNet. We then follow with pretraining on our synthetic dataset, and finally finetune on NYUv2. We also replicate the corresponding state of the art training schedules by pretraining on ImageNet, followed directly by finetuning on NYUv2, for comparison. We use stochastic gradient descent with learning rate of 1×10^{-5} for training on synthetic data and NYUv2.

Experiments. We use the average pixel-level intersection over union (IoU) to evaluate performance on semantic segmentation. We pretrained the model on our synthetic data

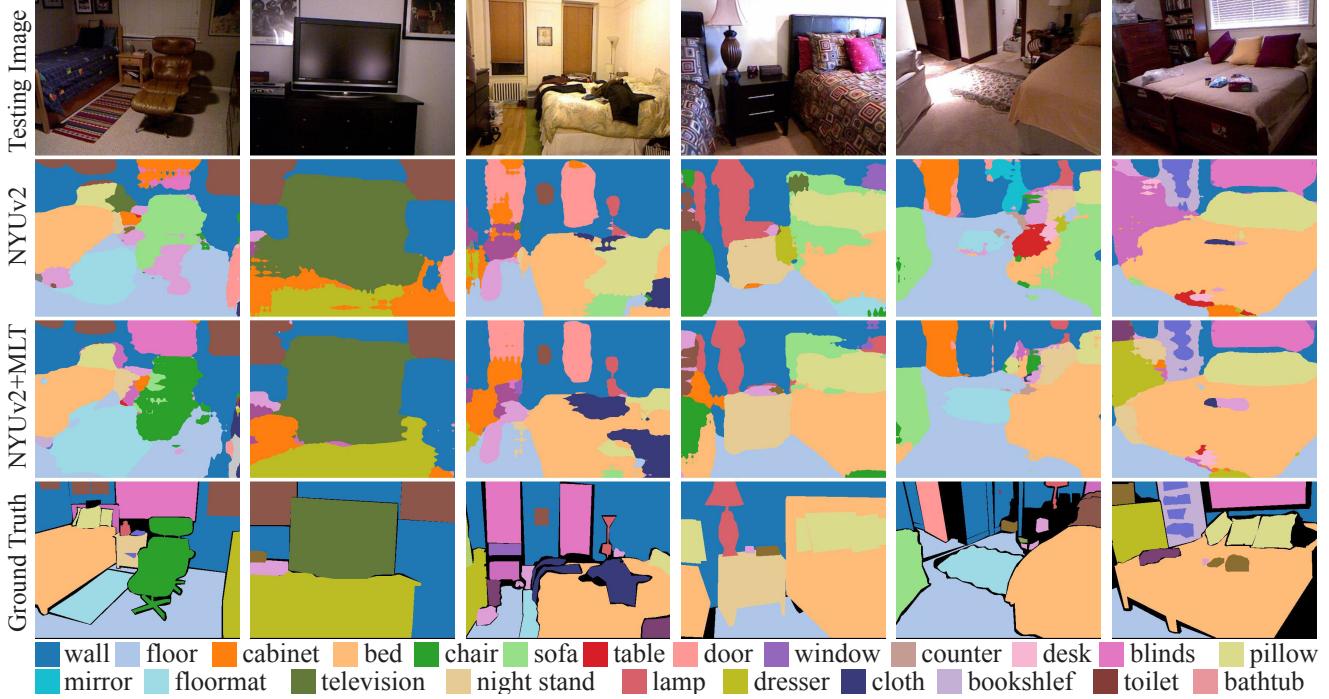


Figure 7. **Semantic Segmentation results.** The model pretrained on synthetic rendering data gives more accurate segmentation result. For example the model trained only with NYU data mis-labeled the chair, whereas the model pretrained on the synthetic data predicts correctly.

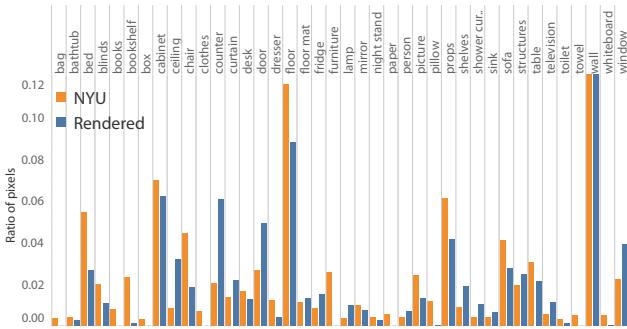


Figure 8. Distribution of classes in our data.

with different rendering method: depth, OpenGL color rendering, and MLT color rendering. For the depth based model we encode the depth using HHA same as [9]. Overall, pretraining on synthetic data helps improve the performance in semantic segmentation, compared to directly training on NYUv2 as seen in Figure 7, and Table 4.2. This shows that the synthetic data helps the network learn richer high level context information than limited real data.

Handa *et al.* [11] use only rendered depth to train their 11 class semantic segmentation model due to the lack of realistic texture and material in their dataset (see HHA results in Table 4.2). However, our results demonstrate that color information is critical for more fine gained semantic segmentation task: in the 40 class task Model trained with color information achieves significantly better performance. For the color based models, pretraining on physically based rendering images helps to achieve better performance than pretraining on OpenGL rendering. This finding is consistent with normal estimation experiments.

| Input | Pre-train | Mean IoU |
|-------|-------------------------|-------------|
| HHA | ImageNet | 27.6 |
| | ImageNet+OpenGL | 30.2 |
| | Long <i>et al.</i> [15] | 31.6 |
| | Yu <i>et al.</i> [29] | 31.7 |
| RGB | ImageNet + OPENGL | 32.8 |
| | ImageNet + MLT | 33.2 |

Table 2. Performance of Semantic Segmentation on NYUv2 with different training setting. All models are fine-tuned on NYUv2.

4.3. Object Boundary Detection

Method. We adopt Xie *et al.*'s [28] network architecture for object boundary detection task as they reported performance on NYUv2. The network starts with the front end of VGG-16, followed by a set of auxiliary-output layers, which produce boundary maps in multiple scales from fine to coarse. A weighted-fusion layer then learns the weights to combine boundary outputs in multi-scale to produce the final result. To evaluate the network, we follow the setting in [10], where the boundary ground truth is defined as the boundary of instance level segmentation.

Training. Similar to the semantic segmentation, we first initialize the network with pretrained weights on ImageNet. We then pretrain on our synthetic dataset, and finetune on NYUv2. We also replicate the state of the art training procedure by pretraining on ImageNet, and directly finetune on NYUv2, for comparison. To highlight the difference between multiple rendering techniques, we only train on color image without using depth. We follow the same procedure

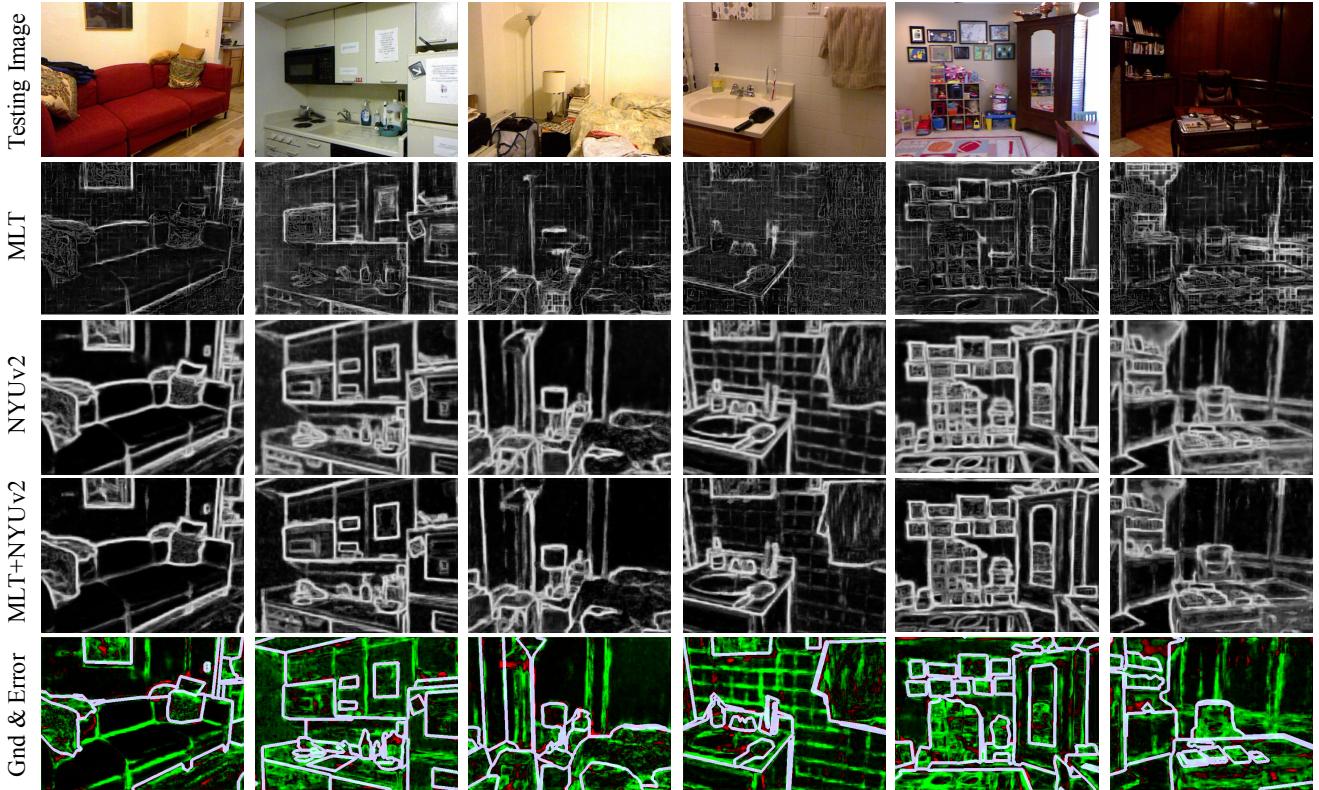


Figure 9. **Boundary estimation results.** The last row shows ground truth overlaid with the difference between model without (NYUv2) and with (MLT+NYUv2) synthetic data pretraining. Red and green indicates pixels enhanced and suppressed by MLT+NYUv2. The model with synthetic data pretraining successfully suppresses texture and background edges compared to the model without.

| Pre-train | Finetune | OSD↑ | OIS↑ | AP↑ | R50↑ |
|-----------|----------|--------------|--------------|--------------|--------------|
| NYUv2[28] | - | 0.713 | 0.725 | 0.711 | 0.267 |
| OPENGL-IL | - | 0.523 | 0.555 | 0.511 | 0.504 |
| MLT-IL/OL | - | 0.604 | 0.621 | 0.587 | 0.749 |
| OPENGL-IL | NYUv2 | 0.716 | 0.729 | 0.715 | 0.893 |
| MLT-IL/OL | NYUv2 | 0.725 | 0.736 | 0.720 | 0.887 |

Table 3. Performance of boundary detection on NYUv2

introduced in [28]. The standard stochastic gradient descend is used for optimization. The learning rate is initially set to be smaller (2×10^{-7}) to deal with larger image resolution of NYUv2, and is reduced even more, to $1/10$ after each $10K$ iterations on NYUv2. For synthetic data, similar to our procedure in like normal estimation task, the learning rate is reduced every $300k$ iterations.

Experiments. We train the model proposed in Xie *et al.*'s [28] with multiple different protocols and show our comparison and evaluation on NYUv2 in Table 3. Following the setting of [28], we take the average of the output from 2nd to 4th multiscale layers as the final result and perform non-maximum suppression and edge thinning. We use the ground truth in [10], and evaluation metrics in [5].

We train with the code released by [28] and achieve the performance shown in the first row of Table 3. We could not replicate the exact number in the paper but we were fairly close, which might be due to the randomized nature of training procedure. We first finetune the model based on the ImageNet initialization on the synthetic dataset and further

finetune on NYUv2. Table 3 shows that the synthetic data pretraining provides consistent improvement on all evaluation metrics. Consistently, we see the model pretrained with MLT rendering achieves the best performance.

Figure 9 shows a comparison between results from different models. Pretrained model on synthetic data, prior to finetuning on real data produces sharper results but is more sensitive to noise. The last column highlights the difference between model with and without pretraining on our synthetic data. We can see that edges within objects themselves as well as the ones in the background (green) are suppressed and true object boundary (red) are enhanced by the model with pretraining on synthetic.

5. Conclusion

We introduce a large-scale synthetic dataset with 500K rendered images of contextually meaningful 3D indoor scenes with different lighting and rendering settings, as well as indoor scenes models they were rendered from. We show that pretraining on our physically based rendering with realistic lighting boosts the performance of indoor scene understanding tasks upon the state of the art methods.

Acknowledgments

This work is supported by Adobe, Intel, Facebook, and NSF (IIS-1251217 and VEC 1539014/ 1539099). It makes use of data from Planner5D and hardware provided by NVIDIA and Intel.

References

- [1] Mitsuba physically based renderer. <http://www.mitsuba-renderer.org/>. 4
- [2] M. Aubry and B. C. Russell. Understanding deep features with computer-generated imagery. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2875–2883, 2015. 1, 2
- [3] A. Bansal, B. C. Russell, and A. Gupta. Marr revisited: 2D–3D alignment via surface normal prediction. In *Conference on Computer Vision and Pattern Recognition*, 2016. 5
- [4] 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, 3, 5
- [5] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1558–1570, 2015. 8
- [6] A. Dosovitskiy, P. Fischer, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, et al. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766. IEEE, 2015. 1, 2
- [7] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015. 5, 6
- [8] 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
- [9] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Aligning 3d models to rgb-d images of cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4731–4740, 2015. 2, 7
- [10] 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. 6, 7, 8
- [11] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla. Scenenet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041*, 2015. 1, 2, 7
- [12] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014. 2
- [13] B. Kaneva, A. Torralba, and W. T. Freeman. Evaluation of image features using a photorealistic virtual world. In *2011 International Conference on Computer Vision*, pages 2282–2289. IEEE, 2011. 2
- [14] 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
- [15] 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. 5, 6, 7
- [16] B. Z. Lubor Ladický and M. Pollefeys. Discriminatively trained dense surface normal estimation. 5
- [17] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh. How useful is photo-realistic rendering for visual learning? *arXiv preprint arXiv:1603.08152*, 2016. 1, 2
- [18] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016. 2, 3
- [19] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016. 3
- [20] S. Shuran, Y. Fisher, Z. Andy, X. C. Angel, S. Manolis, and F. Thomas. Semantic Scene Completion from a Single Depth Image. In *arXiv*, 2016. 1, 3
- [21] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012. 1, 5
- [22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5, 6
- [23] S. Song, S. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, 2015. 1
- [24] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015. 1, 2
- [25] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012. 5
- [26] E. Veach and L. J. Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76. ACM Press/Addison-Wesley Publishing Co., 1997. 4
- [27] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Motaghi, L. Guibas, and S. Savarese. Objectnet3d: A large scale database for 3d object recognition. In *European Conference on Computer Vision*, pages 160–176. Springer, 2016. 3
- [28] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015. 5, 7, 8
- [29] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 5, 6, 7
- [30] Y. Zhang, M. Bai, P. Kohli, S. Izadi, and J. Xiao. Deepcontext: Context-encoding neural pathways for 3d holistic scene understanding. *arXiv preprint arXiv:1603.04922*, 2016. 1, 2, 3