SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation?

John McCormac, Ankur Handa, Stefan Leutenegger, Andrew J. Davison
Dyson Robotics Laboratory at Imperial College, Department of Computing,
Imperial College London
{brendan.mccormac13,s.leutenegger,a.davison}@imperial.ac.uk, handa.ankur@gmail.com

Abstract

We introduce SceneNet RGB-D, a dataset providing pixel-perfect ground truth for scene understanding problems such as semantic segmentation, instance segmentation, and object detection. It also provides perfect camera poses and depth data, allowing investigation into geometric computer vision problems such as optical flow, camera pose estimation, and 3D scene labelling tasks. Random sampling permits virtually unlimited scene configurations, and here we provide 5M rendered RGB-D images from 16K randomly generated 3D trajectories in synthetic layouts, with random but physically simulated object configurations. We compare the semantic segmentation performance of network weights produced from pre-training on RGB images from our dataset against generic VGG-16 ImageNet weights. After fine-tuning on the SUN RGB-D and NYUv2 real-world datasets we find in both cases that the synthetically pre-trained network outperforms the VGG-16 weights. When synthetic pre-training includes a depth channel (something ImageNet cannot natively provide) the performance is greater still. This suggests that large-scale high-quality synthetic RGB datasets with task-specific labels can be more useful for pre-training than real-world generic pre-training such as ImageNet. We host the dataset at http://robotvault.bitbucket.io/scenenet-rgbd.html.

1. Introduction

A primary goal of computer vision research is to give computers the capability to reason about real-world images in a human-like manner. Recent years have witnessed large improvements in indoor scene understanding, largely driven by the seminal work of Krizhevsky *et al.* [19] and the increasing popularity of Convolutional Neural Networks (CNNs). That work highlighted the importance of large scale labelled datasets for supervised learning algorithms.

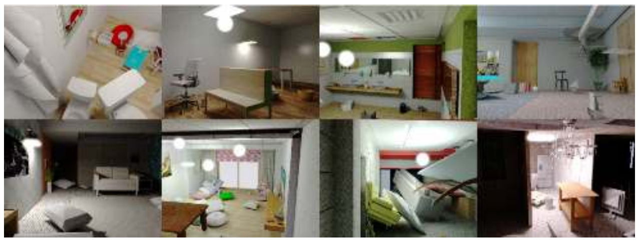


Figure 1. Example RGB rendered scenes from our dataset.

In this work we aim to obtain and experiment with large quantities of labelled data without the cost of manual capturing and labelling. In particular, we are motivated by tasks which require more than a simple text label for an image. For tasks such as semantic labelling and instance segmentation, obtaining accurate per-pixel ground truth annotations by hand is a pain-staking task, and the majority of RGB-D datasets have until recently been limited in scale [28, 30].

A number of recent works have started to tackle this problem. Hua *et al.* provide sceneNN [15], a dataset of 100 labelled meshes of real world scenes, obtained with a reconstruction system with objects labelled directly in 3D for semantic segmentation ground truth. Armeni *et al.* [1] produced 2D-3D-S dataset with 70K RGB-D images of 6 large-scale indoor (educational and office) areas with 270 smaller rooms, and the accompanying ground-truth annotations. Their work used 360° rotational scans at fixed locations rather than a free 3D trajectory. Very recently, ScanNet by Dai *et al.* [6] provided a large and impressive real-world RGB-D dataset consisting of 1.5K free reconstruction trajectories taken from 707 indoor spaces, with 2.5M frames, along with dense 3D semantic annotations obtained manually via mechanical turk.

Obtaining other forms of ground-truth data from real-world scenes, such as noise-free depth readings, precise camera poses, or 3D models is even harder and often can only be estimated or potentially provided with costly additional equipment (e.g. LIDAR for depth, VICON for cam-

	NYUv2 [28]	SUN RGB-D [30]	sceneNN [15]	2D-3D-S [1]	ScanNet [6]	SceneNet [12]	SUN CG* [31, 32]	**SceneNet RGB-D**
RGB-D videos available	✓	✗	✗	✓	✓	✗	✗	✓
Per-pixel annotations	Key frames	Key frames	Videos	Videos	Videos	Key frames	Key Frames	Videos
Trajectory ground truth	✗	✗	✗	✓	✓	✗	✓	✓
RGB texturing	Real	Real	Real	Real	Real	Non-photorealistic	Photorealistic	Photorealistic
Number of layouts	464	-	100	270	1513	57	45,622	57
Number of configurations	464	-	100	270	1513	1000	45,622	16,895
Number of annotated frames	1,449	10K	-	70K	2.5M	10K	400K	5M
3D models available	✗	✗	✓	✓	✓	✓	✓	✓
Method of design	Real	Real	Real	Real	Real	Manual and Random	Manual	Random

Table 1. A comparison table of 3D indoor scene datasets and their differing characteristics. sceneNN provides annotated 3D meshes instead of frames, and so we leave the number of annotated frames blank. 2D-3D-S provides a different type of camera trajectory in the form of rotational scans at positions rather than free moving 3D trajectories. *We combine within this column the additional recent work of physically based renderings of the same scenes produced by Zhang *et al.* [32], it is that work which produced 400K annotated frames.

era pose tracking). In other domains, such as highly dynamic or interactive scenes, synthetic data becomes a necessity. Inspired by the low cost of producing very large-scale synthetic datasets with complete and accurate ground-truth information, as well as the recent successes of synthetic data for training scene understanding systems, our goal is to generate a large photorealistic indoor RGB-D video dataset and validate its usefulness in the real-world.

This paper makes the following core contributions:

- We make available the largest (5M) indoor synthetic video dataset of high-quality ray-traced RGB-D images with full lighting effects, visual artefacts such as motion blur, and accompanying ground truth labels.

- We outline a dataset generation pipeline that relies to the greatest degree possible on fully automatic randomised methods.

- We propose a novel and straightforward algorithm to generate sensible random 3D camera trajectories within an arbitrary indoor scene.

- To the best of our knowledge this is the first work to show that a RGB-CNN pre-trained from scratch on synthetic RGB images can outperform an identical network initialised with the real-world VGG-16 ImageNet weights [29] on a real-world indoor semantic labelling dataset, after fine-tuning.

In Section 3 we provide a description of the dataset itself. Section 4 describes our random scene generation method, and Section 5 discusses random trajectory generation. In Section 6 we describe our rendering framework. Finally, Section 7 details our experimental results.

2. Background

A growing body of research has highlighted that carefully synthesised artificial data with appropriate noise models can be an effective substitute for real-world labelled data in problems where ground-truth data is difficult to obtain. Aubry *et al.* [2] used synthetic 3D CAD models for learning visual elements to do 2D-3D alignment in images, and similarly Gupta *et al.* [10] trained on renderings of synthetic objects to do alignment of 3D models with RGB-D images. Peng *et al.*[22] augmented small datasets of objects with renderings of synthetic 3D objects with random textures and backgrounds to improve object detection performance. FlowNet [8] and FlowNet 2.0 [16] both used training data obtained from synthetic flying chairs for optical flow estimation; and de Souza *et al.* [7] used procedural generation of human actions with computer graphics to generate large dataset of videos for human action recognition.

For semantic scene understanding, our main area of interest, Handa *et al.* [12] produced SceneNet, a repository of labelled synthetic 3D scenes from five different categories. That repository was used to generate per-pixel semantic segmentation ground truth for depth-only images from random viewpoints. They demonstrated that a network trained on 10K images of synthetic depth data and fine-tuned on the original NYUv2 [28] and SUN RGB-D [30] real image datasets shows an increase in the performance of semantic segmentation when compared to a network trained on just the original datasets.

For outdoor scenes, Ros *et al.* generated the SYNTHIA [26] dataset for road scene understanding, and two independent works by Richter *et al.* [24] and Shafaei *et al.* [27] produced synthetic training data from photorealistic gaming engines, validating the performance on real-world segmentation tasks. Gaidon *et al.* [9] used the Unity engine to create the Virtual KITTI dataset, which takes real-world seed videos to produce photorealistic synthetic variations to evaluate robustness of models to various visual factors. For indoor scenes, recent work by Qui *et al.* [23] called UnrealCV provided a plugin to generate ground truth data and photorealistic images from the UnrealEngine. This use of gaming engines is an exciting direction, but is can be limited by proprietary issues either by the engine or the assets.

Our SceneNet RGB-D dataset uses open-source scene layouts [12] and 3D object repositories [3] to provide textured objects. For rendering, we have built upon an open-source ray-tracing framework which allows significant flex-

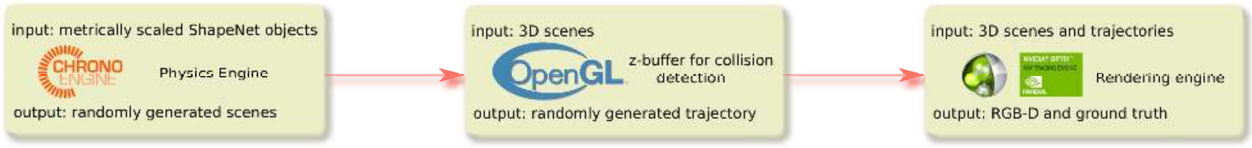

input: metrically scaled ShapeNet objects
CHRONO ENGINE
Physics Engine
output: randomly generated scenes
input: 3D scenes
OpenGL
z-buffer for collision detection
output: randomly generated trajectory
input: 3D scenes and trajectories
Rendering engine
output: RGB-D and ground truth

Figure 2. Flow chart of the different stages in our dataset generation pipeline.

ibility in the ground truth data we can collect and visual effects we can simulate.

Recently, Song *et al.* released the SUN-CG dataset [31] containing ≈46K synthetic scene layouts created using Planner5D. The most closely related approach to ours, and performed concurrently with it, is the subsequent work on the same set of layouts by Zhang *et al.* [32], which provided 400K physically-based RGB renderings of a randomly sampled still camera within those indoor scenes and provided the ground truth for three selected tasks: normal estimation, semantic annotation, and object boundary prediction. Zhang *et al.* compared pre-training a CNN (already with ImageNet initialisation) on lower quality OpenGL renderings against pre-training on high quality physically-based renderings, and found pre-training on high quality renderings outperformed on all three tasks.

Our dataset, SceneNet RGB-D, samples random layouts from SceneNet [12] and objects from ShapeNet [3] to create a practically unlimited number of scene configurations. As shown in Table 1, there are a number of key differences between our work and others. Firstly, our dataset explicitly provides a randomly generated sequential video trajectory within a scene, allowing 3D correspondences between viewpoints for 3D scene understanding tasks, with the ground truth camera poses acting in lieu of a SLAM system [20]. Secondly, Zhang *et al.* [32] use manually designed scenes, while our randomised approach produces chaotic configurations that can be generated on-the-fly with little chance of repeating. Moreover, the layout textures, lighting, and camera trajectories are all randomised, allowing us to generate a wide variety of geometrically identical but visually differing renders as shown in Figure 7.

We believe such randomness could help prevent overfitting by providing large quantities of less predictable training examples with high instructional value. Additionally, randomness provides a simple baseline approach against which more complex scene-grammars can justify their added complexity. It remains an open question whether randomness is preferable to designed scenes for learning algorithms. Randomness leads to a simpler data generation pipeline and, given a sufficient computational budget, allows for dynamic on-the-fly generated training examples suitable for active machine learning. A combination of the two approaches, with a reasonable manually designed scene layouts or semantic constraints along-side physically simulated randomness, may in the future provide the best of both worlds.

3. Dataset Overview

The overall pipeline is depicted in Figure 2. It was necessary to balance the competing requirements of high frame-rates for video sequences with the computational cost of rendering many very similar images, which would not provide significant variation in the training data. We decided upon 5 minute trajectories at 320×240 image resolution, with a single frame per second, resulting in 300 images per trajectory (the trajectory is calculated at 25Hz, however we only render every 25th pose). Each view consists of both a shutter open and shutter close camera pose. We sample from linearly interpolations of these poses to produce motion blur. Each render takes 2–3 seconds on an Nvidia GTX 1080 GPU. There is also a trade off between rendering time and quality of renders (see Figure 6 in Section 6.2).

Various ground truth labels can be obtained with an extra rendering pass. Depth is rendered as the first ray intersection euclidean distance, and instance labels are obtained by assigning indices to each object and rendering these. For ground truth data a single ray is emitted from the pixel centre. In accompanying datafiles we store, for each trajectory, a mapping from instance label to a WordNet semantic label. We have 255 WordNet semantic categories, including 40 added by the ShapeNet dataset. Given the static scene assumption and the depth map, instantaneous optical flow can also be calculated as the time-derivative of a surface points projection into camera pixel space with respect to the linear interpolation of the shutter open and shutter close poses. Examples of the available ground-truth is shown in Figure 3, and code to reproduce it is open-source.[1]

Our dataset is separated into train, validation, and test sets. Each set has a unique set of layouts, objects, and trajectories. However the parameters for randomly choosing lighting and trajectories remain the same. We selected two layouts from each type (bathroom, kitchen, office, living room, and bedroom) for the validation and test sets making the layout split 37-10-10. For ShapeNet objects within a scene we randomly divide the objects within each WordNet class into 80-10-10% splits for train-val-test. This ensures that some of each type of object are in each training set. Our final training set has 5M images from 16K room configurations, and our validation and test set have 300K images from 1K different configurations. Each configuration has a single trajectory through it.

[1] https://github.com/jmccormac/pySceneNetRGBD

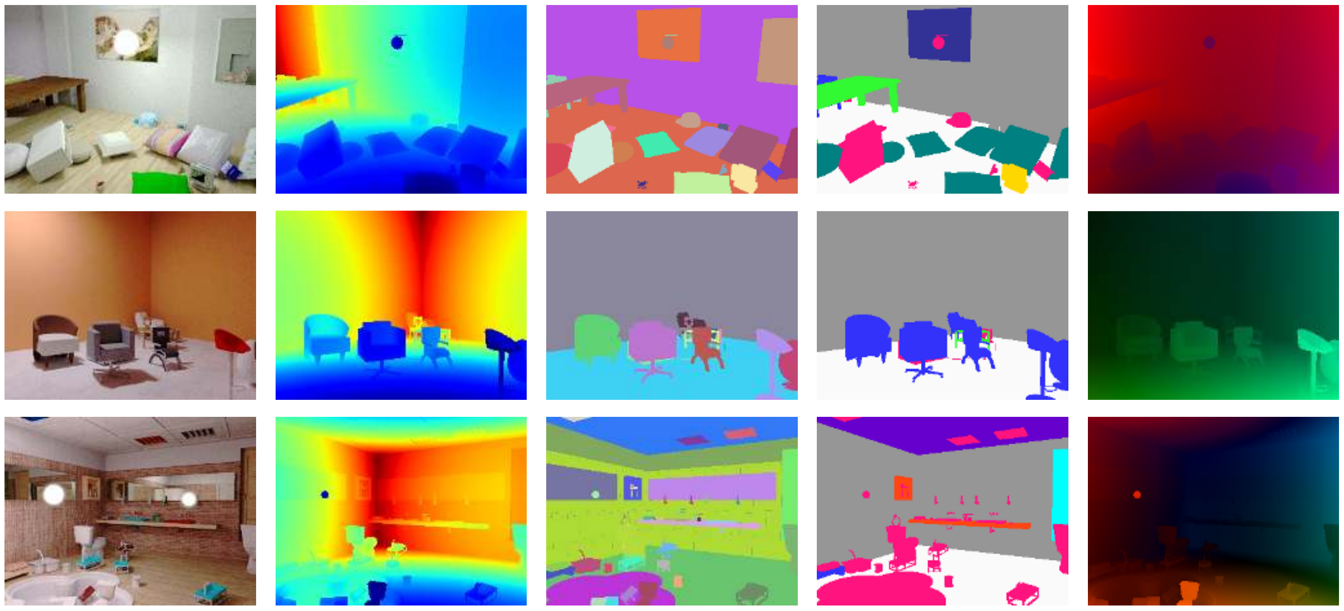


Figure 3. Hand-picked examples from our dataset; (a) rendered images and (b)–(e) the ground truth labels we generate.

4. Generating Random Scenes with Physics

To create scenes, we randomly select a density of objects per square metre. In our case we have two of these densities. For large objects we choose a density between 0.1 and 0.5 objects m^{-2}, and for small objects ($<$0.4m tall) we choose a density between 0.5 and 3.0 objects m^{-2}. Given the floor area of a scene, we calculate the number objects needed. We sample objects for a given scene according to the distribution of objects categories in that scene type in the SUN RGB-D real-world dataset. We do this with the aim of including relevant objects within a context *e.g.* a bathroom is more likely to contain a sink than a microwave. We then randomly pick an instance uniformly from the available models for that object category.

We use an off-the-shelf physics engine, Project Chrono,[2] to dynamically simulate the scene. The objects are provided with a constant mass (10kg) and convex collision hull and positioned randomly within the 3D space of the layouts axis-aligned bounding box. To slightly bias objects towards the correct orientation, we offset the center of gravity on the objects to be below the mesh. Without this, we found very few objects were in their normal upright position after the simulation. We simulate 60s of the system to allow objects to settle to a physically realistic configuration. While not organised in a human manner, the overall configuration aims to be physically plausible *i.e.* avoiding configurations where an object cannot physically support another against gravity or with unrealistic object intersections.

[2] https://projectchrono.org/

5. Generating Random Trajectories

As we render videos at a large scale, it is imperative that the trajectory generation be automated to avoid costly manual labour. The majority of previous works have used a SLAM system operated by a human to collect hand-held motion: the trajectory of the camera poses returned by the SLAM system is then inserted into a synthetic scene and the corresponding data is rendered at discrete or interpolated poses of the trajectory [11, 13]. However, such reliance on humans to collect trajectories quickly limits the potential scale of the dataset.

We automate this process using a simple random camera trajectory generation procedure which we have not found in any previous synthetic dataset work. For our trajectories, we have the following desiderata. Our generated trajectories should be random, but slightly biased towards looking into central areas of interest, rather than for example panning along a wall. It should contain a mix of fast and slow rotations like those of a human operator focussing on nearby and far away points. It should also have limited rotational freedom that emphasises yaw and pitch rather than rolling, which is a less prominent motion in human trajectories.

To achieve the desired trajectories we simulate two physical bodies. One defines the location of the camera, and another the point in space that it is focussing on as a proxy for a human paying attention to random points in a scene. We take the simple approach of locking roll entirely, by setting the up vector to always be along the positive y-axis. These two points completely define the camera coordinate system.

We simulate the motion of the two bodies using a physical motion model. We use simple Euler integration to simulate the motion of the bodies and apply randomly sampled 3D directional force vectors as well as drag to each of the bodies independently, with a maximum cap on the permitted speed. This physical model has a number of benefits. Firstly, it provides an intuitive set of metric physical properties we can set to achieve a desired trajectory, such as the strength of the force in Newtons and the drag coefficients. Secondly, it naturally produces smooth trajectories. Finally, although not currently provided in our dataset, it can automatically produce synthetic IMU measurements, which could prove useful for Visual-Inertial systems.

We initialise the pose and 'look-at' point from a uniform random distribution within the bounding box of the scene, ensuring they are less than 50cm apart. As not all scenes are convex, it is possible to initialise the starting points outside of a layout, for example in an 'L'-shaped room. Therefore, we have two simple checks. The first is to restart the simulation if either body leaves the bounding volume. The second is that within the first 500 poses at least 10 different object instances must have been visible. This prevents trajectories external to the scene layout with only the outer wall visible.

Finally, to avoid collisions with the scene or objects we render a depth image using the z-buffer of OpenGL. If a collision occurs, the velocity is simply negated in a 'bounce', which simplifies the collision by assuming the surface normal is always the inverse of the velocity vector. Figure 4 visualises a two-body trajectory from the final dataset.

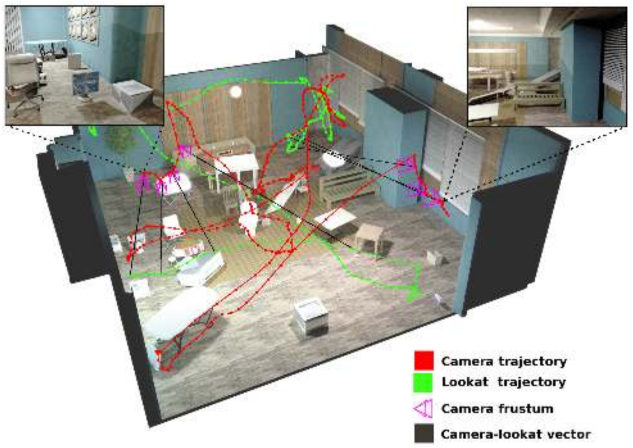
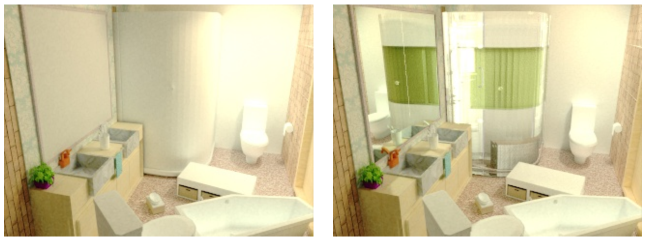


Figure 4. Example camera and lookat trajectory through a synthetic scene (with rendered views from the first and last frustum).

6. Rendering Photorealistic RGB Frames

The rendering engine used was a modified version of the Opposite Renderer[3] [21], a flexible open-source ray-tracer built on top of the Nvidia OptiX framework. We do not have strict real-time constraints to produce photorealistic rendering, but the scale and quality of images required does mean the computational cost is an important factor to consider. Since OptiX allows rendering on the GPU it is able to fully utilise the parallelisation offered by readily-available modern day consumer grade graphics cards.

[3]http://apartridge.github.io/OppositeRenderer/



(a) No reflections & transparency (b) With reflections & transparency

Figure 5. Reflections and transparency

6.1. Photon Mapping

We use a process known as photon mapping to approximate the rendering equation. Our static scene assumption makes photon mapping particularly efficient as we can produce photon maps for a scene which are maintained throughout the trajectory. A good tutorial on photon mapping is given by its creators Jensen *et al.*[18]. Normal ray-tracing allows for accurate reflections and transparency renderings, but photon mapping provides a global illumination model that also approximates indirect illumination, colour-bleeding from diffuse surfaces, and caustics. Many of these effects can be seen in Figure 5.

6.2. Rendering Quality

Rendering over 5M images requires a significant amount of computation. We rendered our images on 4-12 GPUs for approximately one month. An important trade-off in this calculation is between the quality of the renders and the quantity of images. Figure 6 shows two of the most important variables dictating this balance within our rendering framework. Our final dataset was rendered with 16 samples per pixel and 4 photon maps. This equates to approximately 3s per image on a single GPU.

6.3. Random Layout Textures and Lighting

To improve the variability within our 57 layouts, we randomly assign textures from a curated library of selected seamless textures to their components. Each layout object has a material type, which then gives a number of random texture images for that type. For example, we have a large number of different wall textures, floor textures, and curtain textures. We also generate random indoor lighting for the scene. We have two types of lights: spherical orbs, which serve as point light sources, and parallelograms which act

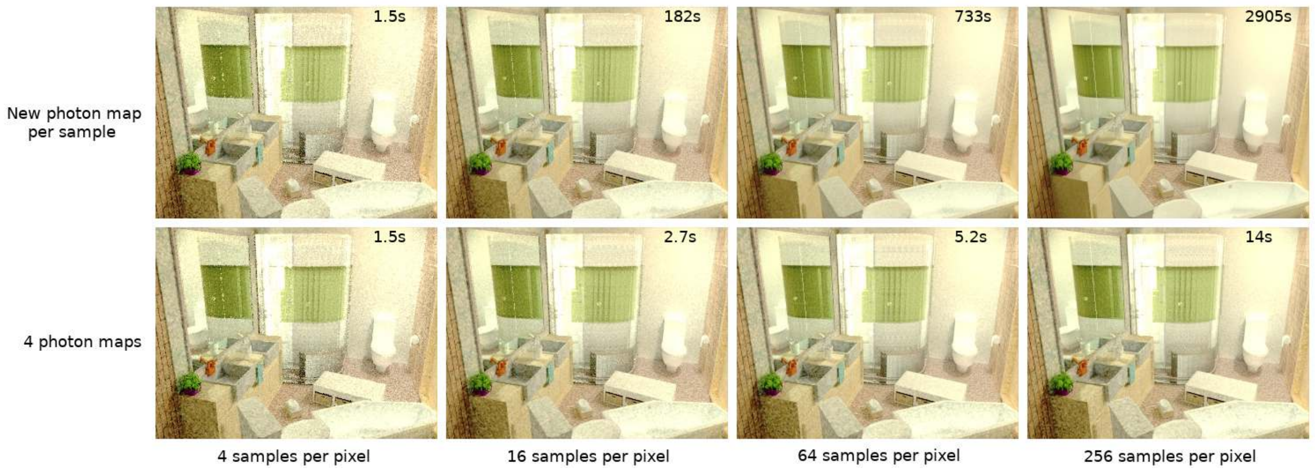


Figure 6. Trade off between rendering time and quality. Each photon map contains approximately 3M stored photons.

as area lights. We randomly pick a hue and power of each light and then add them to a random location within the scene. We bias this location to be within the upper half of the scene. This approach allows an identical geometric layout to result in multiple renders with very different visual characteristics (see Figure 7).

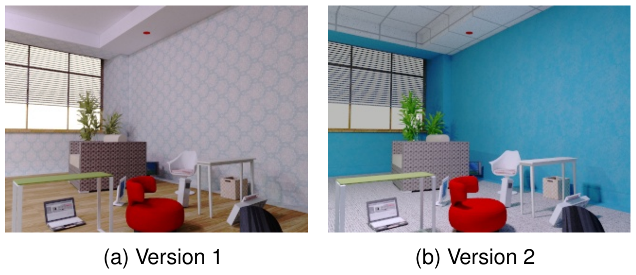


Figure 7. Same scene with different lighting and layout textures.

6.4. Camera Model and CRF

Our simulated camera is a simple global shutter pinhole model, with a focal length of 20cm, a horizontal FoV of 60°and vertical FoV of 45°. In order to make sure that the rendered images are a faithful approximation to real-world images, we also apply a non-linear Camera Response Function (CRF) that maps irradiance to quantised brightness as in a real camera. We do not explicitly add camera noise or distortion to the renderings, however the random ray-sampling and integration procedure from ray-tracing naturally adds a certain degree of noise to the final images.

6.5. Motion Blur

For fast motion we integrate incoming rays throughout a shutter exposure to approximate motion blur — this can be efficiently performed within the rendering process by changing the poses from which samples are drawn for each pixel and integrating the irradiance value rather than for example averaging RGB values after rendering. For an example rendering using this technique see Figure 8. The motion blur does not affect the ground truth outputs of depth or instance segmentations. For these images we set the pose to be the exact midpoint of the shutter exposure.

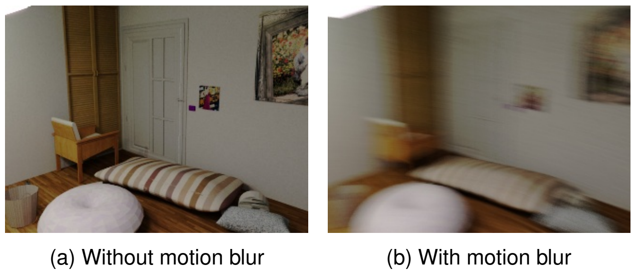


Figure 8. Motion blur examples.

7. Experiments

We test the value of SceneNet RGB-D as a training set for semantic segmentation, using the real image datasets NYUv2 and SUN RGB-D as our benchmarks. More specifically, we compare the performance of three different pre-trained network weights on the task of per-pixel semantic labelling. The three weight initialisations are: a network trained from scratch with initialisation proposed by He *et al.* [14], a network (originally initialised with [14]) pre-trained on the 5M synthetic RGB images from the SceneNet RGB-D dataset, and a network initialised with the VGG-16 ImageNet weights. As the VGG-16 ImageNet weights are for a classification task, the second 'upsampling' half of the network architecture described below is not available and is also initialised with the scheme proposed by He *et al.* [14] before fine-tuning.

The comparison of ImageNet vs. synthetic RGB-only is particularly challenging as the ImageNet weights are trained using 1M real-world images, just as our final test datasets are drawn from the real-world. The important question is whether the task-specific information available in our dataset (indoor scene classes and per-pixel ground truth labelling instead of generic classification tags such as cats and dogs) combined with 5$\times$ more images is enough to counterbalance the advantages of real-world images.

We also experiment with another advantage our synthetic dataset provides, that of a wider variety of potential input domains. We train a slightly modified network architecture to include a depth-channel and train this from scratch using the SceneNet RGB-D dataset. We compare this against training from scratch (scaling depth input in each dataset by a constant factor) but do not directly compare the depth network against ImageNet pre-training for a number of reasons. First, ImageNet does not provide depth data — there is no depth channel for publicly available weights to directly compare against. Second, in the RGB-D network architecture described below we maintained a similar number of feature maps for the first half of the network, split evenly between depth and RGB due to memory constraints; this unfortunately prevents a direct mapping of VGG-16 weights into even the RGB-only part of the network.

For both the NYUv2 and SUN RGB-D datasets we choose the 13 class semantic mapping defined by Couprie *et al.* [5]. We manually map each of the WordNet ids in our dataset to one of the 13 semantic classes. We give three accuracy metrics: global pixel accuracy (proportion of correctly classified pixels out of all ground truth labelled pixels), class average accuracy (average over classes of the proportion of correctly classified pixels of a class to the ground truth labels of that class, the accuracy of each class is also given), and mean Intersection over Union or IU (average over classes of the proportion of correctly classified pixels of a class to the ground truth labels of that class plus false positives of that class).

7.1. Network Architectures

We choose the relatively straightforward U-Net [25] architecture as the basis for our experiments, with the slight modification of applying Batch Normalisation [17] between each convolution and non-linearity. Our inputs are RGB 320$\times$240 images and our output is 320$\times$240$\times$14 class probabilities (the first class being 'missing ground-truth label' as observed in SUN RGB-D and NYUv2 and ignored in the loss function for training purposes). The RGB-only network contains 22M free parameters.

To accommodate a depth channel in the RGB-D CNN we modify the U-Net architecture to have an additional column of depth-only convolutions. The second half of U-Net remains unchanged; we simply concatenate the output of both the depth and RGB feature maps during the upsampling portion of the CNN. We maintain the same number of feature maps in the first half of the network step, but split them evenly for depth and RGB, i.e. the first 64-channel RGB convolution becomes a 32-channel RGB convolution and 32-channel depth convolution. Finally, to maintain approximately consistent memory usage and batch sizes during training, we skip the final max-pooling layer and 1024-channel convolutions. Instead we simply concatenate the last two 512-channel blocks of RGB and Depth feature maps to produce the 1024-channel in the first stage of upsampling in the U-Net. Overall these changes reduce the number of free-parameters to 17.2M.

7.2. Training

Our network implementation is built within the `Torch7` [4] framework, and trained on a multi-GPU system. We use the standard per-pixel cross-entropy loss after the softmax layer. We shuffle the full dataset at the beginning of each epoch, and train with the largest batch-size that would fit in memory. Depending on the network architecture the batch-size varied from 25-30. For all experiments the learning rate was initialised at 0.1, and scaled by 0.95 after every 30 epochs. Our networks pre-trained on SceneNet RGB-D were both maintained at a constant learning rate as they were below 30 epochs - the RGB CNN was pre-trained for 15 epochs which took approximately 1 month on 4 Nvidia Titan X GPUs, and the RGB-D CNN was pre-trained for 10 epochs, taking 3 weeks.

We use the Stochastic Gradient Descent optimiser with momentum of 0.95 and no weight regularisation. Unfortunately there is no official validation set on the NYUv2 or the SUN RGB-D dataset, so for all fine-tuning experiments we perform early stopping and give the validation performance. Generally fine-tuning required $\approx$50 epochs.

7.3. Results

The results of our experiments are summarised in Tables 2 & 3. For RGB we see that in both datasets the network pre-trained on SceneNet RGB-D outperformed the network initialised with VGG-16 ImageNet weights on all three metrics. For the NYUv2 the improvement was +8.4%,+5.9%, and +8.1% for the class average, pixel average, and mean IU respectively, and for the SUN RGB-D dataset the improvement was +1.0%,+2.1%, and +3.5% respectively. This result suggests that a large-scale high-quality synthetic RGB dataset with task-specific labels can be more useful for CNN pre-training than even a quite large ($\approx$1M image) real-world generic dataset such as ImageNet, which lacks the fine-grained ground truth data (per-pixel labelling), or domain-specific content (indoor semantics). The trend between the two datasets is also clear, the improvement from pre-training on the synthetic data is less significant when fine-tuning on the larger 5K image SUN

NYUv2: 13 Class Semantic Segmentation

Pre-training	bed	books	ceiling	chair	floor	furniture	objects	painting	sofa	table	tv	wall	window	class avg.	pixel avg.	mean IU
No pre-training RGB	42.6	11.8	57.5	15.5	76.1	55.5	37.9	48.2	17.9	18.8	35.5	78.4	51.1	42.1	55.8	29.0
ImageNet	45.7	29.4	56.4	35.9	84.1	51.3	43.4	58.6	24.4	26.2	18.8	80.8	63.8	47.6	60.2	33.7
SceneNet RGB	54.7	**30.0**	65.0	48.3	86.2	59.7	52.5	**62.1**	50.2	32.5	40.5	82.4	64.5	56.0	66.1	41.8
No pre-training RGB-D	58.2	1.7	**75.5**	48.7	**94.8**	54.6	47.6	38.2	42.2	32.8	23.2	82.5	43.0	49.5	63.4	36.9
SceneNet RGB-D	**69.4**	22.7	71.1	**63.8**	**94.8**	**64.4**	**56.8**	61.2	**68.9**	**41.0**	**43.1**	**84.3**	**66.9**	**62.2**	**71.7**	**48.2**

Table 2. **NYUv2 validation set results**: Segmentation performance using U-Net architectures described in the text. After the listed form of pre-training, all networks are then fine-tuned on the NYUv2 train set. All evaluations performed at 320 $\times$ 240 resolution.

SUN RGB-D: 13 Class Semantic Segmentation

Pre-training	bed	books	ceiling	chair	floor	furniture	objects	painting	sofa	table	tv	wall	window	class avg.	pixel avg.	mean IU
No pre-training RGB	47.0	26.9	50.0	57.7	88.9	38.4	31.2	39.7	43.0	55.8	18.2	84.6	61.2	49.4	68.9	36.5
ImageNet RGB	56.8	**42.2**	58.0	64.7	88.6	42.9	**49.2**	**59.1**	59.2	51.9	22.6	84.1	59.2	56.8	71.2	40.3
SceneNet RGB	56.4	29.0	66.7	68.9	90.1	53.0	43.4	50.4	51.6	60.1	28.7	83.5	**69.0**	57.8	73.3	43.8
No pre-training RGB-D	**69.0**	19.3	68.0	68.8	**94.3**	46.7	37.8	41.6	55.8	59.0	5.7	86.7	50.5	54.1	73.9	41.4
SceneNet RGB-D	67.2	33.8	**76.2**	**71.6**	93.9	**55.4**	46.5	56.1	**63.1**	**72.8**	**35.8**	**88.2**	61.7	**63.3**	**78.3**	**49.8**

Table 3. **SUN RGB-D validation set results**: Segmentation performance using the U-Net architectures described in the text. After the listed form of pre-training, all networks are then fine-tuned on the SUN RGB-D train set. All evaluations performed at 320 $\times$ 240 resolution.

RGB-D dataset vs. the 795 image NYUv2 dataset.

The inclusion of depth as an additional input channel resulted in a significant performance improvement over RGB-only in both datasets (+6.2%,+5.6%,+6.4% in the NYUv2, and +5.5%,+5.0%,+6.0% in the SUN RGB-D for the class average, pixel average, and mean IU respectively). When compared against training from scratch for the depth channel, pre-training showed a clear performance improvement. We found training the architecture from scratch on the depth modality challenging, taking longer to converge (300 epochs) and with infrequent classes, such as books and TV showing particularly poor results in both datasets.

8. Conclusion

Our aim was to produce and evaluate realistic synthetic per-pixel labelled data of indoor scenes. We anticipate the scale and quality of this dataset could help to better bridge the gap between simulations and reality and be suitable for domain adaption tasks [**?**]. We highlight problems we have tackled such as physically realistic scene layouts, random camera trajectory generation, and photorealistic rendering.

The randomness inherent in our pipeline also allows for a continuous stream of unseen training examples, dynamically designed to target the current limitations of a model being trained. In the future, it is likely that the generation of training data and the training of models will become more tightly interleaved, as the advantages of automatically generated tailored training data becomes clear.

The results of our experiments, to the best of our knowledge, are the first to show an RGB-only CNN pre-trained from scratch on synthetic RGB images improve upon the performance of an identical network initialised with weights from a CNN trained on the large-scale real-world ImageNet dataset. This illustrates the value synthetic datasets can bring to real-world problems. The additional performance improvement achieved by including the depth also serves to highlight the importance of the flexibility and ease with which synthetic data can produce alternative modalities.

Certain capabilities of the dataset still remain to be explored. The availability of video data enables experimentation on CNNs with a temporal capacity. The ability to modify identical geometric scenes with varying lighting and textures enables training invariance to such changes. Finally, the smooth trajectories, and availability of perfect camera pose data provides multi-view correspondences, and lends itself to the exploration of dense semantic SLAM problems.

9. Acknowledgements

Research presented in this paper has been supported by Dyson Technology Ltd.

References

[1] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *arXiv preprint arXiv:1702.01105*, 2017. 1, 2

[2] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014. 2

[3] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. 2, 3

[4] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A Matlab-like Environment for Machine Learning. In *Neural Information Processing Systems (NIPS)*, 2011. 7

[5] C. Couprie, C. Farabet, L. Najman, and Y. LeCun. Indoor semantic segmentation using depth information. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013. 7

[6] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[7] C. R. DeSouza, A. Gaidon, Y. Cabon, and A. M. López Peña. Procedural generation of videos to train deep action recognition networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[8] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 2

[9] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[10] S. Gupta, P. A. Arbeláez, R. B. Girshick, and J. Malika. Aligning 3D Models to RGB-D Images of Cluttered Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[11] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. Real-Time Camera Tracking: When is High Frame-Rate Best? In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. 4

[12] A. Handa, V. Pătrăucean, V. Badrinarayanan, S. Stent, and R. Cipolla. SceneNet: Understanding Real World Indoor Scenes With Synthetic Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3

[13] A. Handa, T. Whelan, J. B. McDonald, and A. J. Davison. A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 4

[14] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 6

[15] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung. Scenenn: A scene meshes dataset with annotations. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2016. 1, 2

[16] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015. 7

[18] H. W. Jensen and N. J. Christensen. A practical guide to global illumination using photon maps. *Siggraph 2000 Course 8*, 2000. 5

[19] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, 2012. 1

[20] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 3

[21] S. A. Pedersen. Progressive photon mapping on gpus. Master's thesis, NTNU, 2013. 5

[22] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3d models. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 2

[23] W. Qiu and A. Yuille. UnrealCV: Connecting computer vision to unreal engine. *arXiv preprint arXiv:1609.01326*, 2016. 2

[24] S. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2

[25] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015. 7

[26] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. 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 (CVPR)*, 2016. 2

[27] A. Shafaei, J. J. Little, and M. Schmidt. Play and Learn: Using Video Games to Train Computer Vision Models. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 2

[28] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. 1, 2

[29] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 2

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

[31] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic Scene Completion from a Single Depth Image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3

[32] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3