DOI: 10.33965/cgv2020_202011L014
International Conferences Computer Graphics, Visualization, Computer Vision and Image Processing 2020;
Big Data Analytics, Data Mining and Computational Intelligence 2020;
and Theory and Practice in Modern Computing 2020

SAMPLE SCENES FROM A DISTRIBUTION: A CONTENT CREATION PIPELINE FOR REALISTIC RENDERING FOR NEURAL NETWORKS TRAINING

Vadim Sanzharov[1], Vladimir Frolov[2], Alexey Voloboy[3] and Vladimir Galaktionov[3]
[1]*Gubkin Russian State University of Oil and Gas, Russia*
[2]*Keldysh Institute of Applied Mathematics Russian Academy of Sciences, Moscow State University, Russia*
[3]*Keldysh Institute of Applied Mathematics Russian Academy of Sciences, Russia*

ABSTRACT

In this paper we present our experience with development of a content creation pipeline targeted at generation of realistic image sequences with highly variable content. Our technique allows rendering of a single 3D object or a 3D scene in variety of appearances which includes changing of geometry, materials and lighting. In our work we were able to generate datasets for individual 3D objects and create procedural generator of interior scenes. Our solution is highly controllable and allows generating datasets with desired distribution of features in a reproducible manner. Using synthetic data in training, we have got increase of accuracy in CNN-based models comparing to usage of real-life data only. During our work we had to significantly improve the content creation pipeline for the existing open source GPU rendering system adapting it for our tasks. In this paper we suggest new approach for content creation which we call "sampling scenes from a distribution".

Figure 1. Examples of our applications. Road signs and cars were rendered as individual objects and then augmented to KITTI dataset for further CNN test; realistic rendering of the interior created as a consistent 3D scene randomizing of objects layout, material and lighting

KEYWORDS

Realistic Rendering, Procedural Pipeline, 3D Scene Generator, GPU Rendering, CNN Training Datasets

1. INTRODUCTION

Training of the modern artificial intelligence models faces two fundamental problems in practice: data quantity and data quality. *Data quantity* concerns availability of sufficient amounts of training and testing data. *Data quality* means how balanced is the data – are all the different classes, which the model must recognize, represented enough, do we have precise marking for it or not, how much noise (of any nature) is present in our dataset. Real-life data sets collected by people usually suffer from both insufficient quality and quantity. The real situation is even worse because during their research Compute Vision (CV) engineers need to change input datasets for testing of certain hypotheses. Due to inability to quickly obtain a new dataset the CV engineers have to consider subsets of the existing dataset which significantly limits their study.

Thus, synthetic (i.e. rendered with realistic image synthesis methods) datasets can be an option. The solution to data quantity problem can be achieved by using algorithms for procedurally setting of the optical properties of materials and surfaces. This way it is possible to quickly generate almost unlimited number of training examples with any distribution of objects (and therefore classes) present in generated images. *The quality problem is in fact more interesting*. On the one hand it is solved automatically: renderer produces precise per-pixel masks generating ground truth marking. On the other hand, it is actually not quite clear what a quality does actually mean for CV engineers. Some research papers on this topic (that we will describe in related work section) assume that quality is subjective realism for human eye. *This is important to some extent but wrong in general*. Realistic rendering in comparison to simple or real-time techniques could improve CNN performance (for example, classification precision). But in our opinion the main problem is different: in practice the CV engineers measure CNN performance on specific data sets and in such formulation of the problem *a quality means to look like other examples of target dataset*. This in itself significantly shifts priorities for the developed software.

2. RELATED WORK

There are a huge number of successful applications of realistic rendering for AI training. For us, first of all, *it is interesting how 3D content was obtained in these works*, how the full pipeline was built and what was the advantages and disadvantages of the selected ways.

2.1 Using Existing 3D Content

Firstly [Movshovitz et al] demonstrated importance of the realistic rendering usage in a problem of view point estimation. In their work authors took a database of the 91 3D CAD models obtained from doschdesign.com and turbosquid.com. These models were rendered with randomized direct lighting and random real photos as background image. We believe this work measured the influence of not realism itself, but rather the effect of increasing dataset variability, which to some extent has an effect with even simple lighting models. The reason is obvious: 3D models downloaded from the Internet resources in most cases are not ready for photorealistic rendering due to missing textures, unknown materials models (which were simply incorrectly exported to the current format or not supported by current rendering system) and, finally, just human errors that were made when preparing the 3D model for the aggregator site.

[Zhang et al] report about using 500K images from 45K scenes for training of semantic segmentation and normal/depth estimation problems. The 3D content was obtained from the SUNCG dataset [Song et al]. There are many problems with this approach, even apart of the well-known legal problems with the SUNCG dataset. Since originally the SUNCG 3D models were assembled in the Planner5D program (which is trivial in comparison to such content creation tools as Blender or 3ds Max) they don't contain realistic setup for materials or lighting. They are limited to just textures for Lambert reflection. *This content is not sufficient for photorealistic rendering*. No wonder that the other works where the SUNCG was used [Li and Snavely, 2018; Kirsanov et al, 2019; McCormac et al, 2017] were able to achieve basic level of realism only.

2.2 Augmented Reality

Preparation of 3D content is expensive and time-consuming task. Therefore, in practice it is worth to consider any possibility for reducing man-hours. Augmented reality is very advanced approach here. Even [Movshovitz et al, 2016] actually used it when randomizing backgrounds. [Alhaija et al, 2018] went further applying augmentation of rendered images to the KITTI dataset [Geiger et al, 2013]. Applying of the image-based lighting technique [Debevec, 2002] for individual car models in conjunction with augmenting real photos [Voloboi et al, 2006] gives cheap and quick result: both rendering computation complexity and content preparation is greatly simplified [Alhaija et al, 2018].

The main problem of image-based lighting methods is that they need the real high dynamic range (5-6 orders of magnitude) to be present in HDR image. Often it is not so [Voloboi et al, 2006]. Lighting prediction methods [like Valiev et al, 2008; Song and Funkhouser, 2019; Garon et al, 2019; Sorokin et al, 2020] can help here. They became a quite common task for mixed-reality systems. Other disadvantage of

augmented reality is the task of automatic, correct placing of 3D models inside some reconstructed representation of scene. It can be easily supported in some cases (for example, [Alhaija et al, 2018] used segmentation methods to reconstruct the road plane) but is quite hard in general (for example, for interior photographs). Thus, the approximate geometry, lighting and camera parameters including tone mapping should be reconstructed in some way and this reconstruction itself is a significant problem. Nevertheless, the main advantage of augmented reality is that it can work directly in the image domain of CV researcher interest expanding existing datasets with new examples. *And thus, Domain Adaptation methods [Tremblay, et al, 2018] are not needed* which seems to be mandatory in practice for achieving high quality with other synthetic ways.

2.3 Procedural and Simulation Approaches

Procedural modeling is expensive, time-consuming and usually restricted to some specific task (for example, dirt or fire modeling) and sometimes to specific rendering algorithm. However, once created, procedural models are highly variable (infinite number of examples), give great quality for rendering (often with infinite-resolution details) and is traditionally used in many cinema content creation pipelines. [Tsirikoglou et al, 2017] used procedural modeling approach to generate cities and showed improvement for neural networks. Buildings, roads and city plan were procedural while other models (like cars, pedestrians, vegetation, bicycles) were sampled randomly from prepared data base. Also, some parameters were randomized for cars: type, count, placement, color.

Approaches based on the procedural modeling via machine learning look promising [Risi and Togelius, 2019; Spick et al, 2019] but there we meet the chicken and egg problem: we need to train these models first. Besides, these approaches are rather new and not proven for practice of general rendering tasks yet.

[Hodan et al, 2019] achieved high quality by using the existing cinema production content creation pipeline with Autodesk Maya and Arnold rendering system, high quality 3D models and physics simulation which was the main randomizing tool. Obviously, this approach suffers from the main disadvantage of current cinema production pipelines: high cost and high labor input. [Hodan et al, 2019] used only 6 scenes with 30 different objects. This was fundamentally different from previous approaches. Other disadvantages of the Hodan's approach include the use of non-freely available tools Maya and Arnold (which limits the adoption of this work), and slow rendering reported by authors – 15 to 720 seconds per image (depending on quality) on 16-core Xeon CPUs with 112 GB of RAM. Reasonably fast dataset generation is therefore possible only if significant computational resources are available. It is important to note that authors reported that accurate modeling of the scene context was more significant (+16% for CNN precision) in comparison to accurate light transport simulation (+6% for CNN precision).

Other works [Shah et al, 2017] show that modern game engines could also be used for AI training. The choice between a game engine or a cinema production rendering system should be done primarily basing on how easy to model target real-world situation.

2.4 Content Creation Pipeline

Unlike other works mentioned earlier a specialized content creation pipeline is proposed in [Denninger et al 2019]. And, as it was stated by the authors precisely, the design of open-source and universal pipeline was their goal. So, it is very close to our work. Sampler modules provide randomization capabilities which are the most interesting part of the pipeline. Sampler modules can generate positions for object placement (cameras, lights, 3d models) with various distributions and constraints like a proximity check. So, for example, object positions can be generated on the spherical surface but the objects will not be too close (for example, cameras will not look straight at the wall) and/or will not collide with each other. Sampler modules are also able to select objects basing on user-defined conditions and manipulate their properties (for example, enable physics simulation for 3d models). The MaterialManipulator and MaterialRandomizer modules were implemented to produce variation in appearance.

The main drawback of this work is the fact that until now it was not yet applied to any specific learning task unlike many previously mentioned papers. So, while it looks very good on a paper and in examples provided by authors, its performance in real-life applications for CNN training is not known yet. As mentioned before the authors intended their work to be a universal pipeline which can be adapted and used

by CV researchers. Therefore, real evaluation of their work can be made only when its applications will be published. We believe that the [Denninger et al 2019] approach is right in general. However, having experience of similar pipeline creation and its application to various AI training tasks, we would discuss several modifications that need to be done in order to apply the Denninger approach in practice.

3. SUGGESTED APPROACH

The first difference between our work and existing approaches is reproducibility of scene generation results which allows generation of the same *desired features* of a scene for the same input setup. In fact, this turns dataset generation into a sampling problem. Having an input vector (x_0, x_1, ... x_n) of random numbers from 0 to 1 we assign to each random variable some specific meaning during randomization script development. For example, if we would like to render cars inside HDR environment, like [Movshovitz et al, 2016; Alhaija et al, 2018] did for 3D model pose/camera parameters estimation, we could:

1. Assign (x_0, x_1) pair for 3D model (car) rotation (additionally we need setting of allowable angle intervals, for example from 0 to 70 degrees for vertical angle and from 0 to 360 for horizontal).
2. Assign x_2 for car body color selection from palette (to restrict desired colors).
3. Assign x_3 for environment select.
4. Assign x_4 for environment rotation around vertical (Y) axis.
5. Assign x_5 for car 3D model selection.

We call this process *a scene sampling*. Thus, sampling of 6D vector (x_0 ... x_5) we obtain images for input samples which gives us important properties and thus powerful capabilities for further experiments:

- Applying quasi-random sequences (we used the Sobol sequence) we kill two birds with one stone: (1) Starting of the Sobol sequence from the beginning for each 3D model (i.e. generate x_0 ... x_5 with quasi random approach, but select 3D models one by one instead of sampling x_6) allows us to generate same camera positions for all 3D models which is important for training of pose estimation algorithms. (2) The Sobol sequence gives good uniform distribution in 5D-6D space. So, the generated samples would be also evenly distributed among all desired parameters: car colors, camera rotation and lighting conditions (which are set from environment map).
- Applying of non-uniform distribution for x_5, allows us to sample more cars of desired types and applying of tabulated distribution for x_2 we can use more cars of specified (for example, black and grey) colors.
- Setting of different values and distributions for training and validation datasets we can easily check hypothesis of our interest. For example, we can generate training dataset with camera rotation of only 25, 50 and 70 degrees but then check if this is enough for rotations 10, 30 and 60 degrees in our test dataset.
- When we move from the current frame to the next one, we would like to localize the scene changes. In such a case the rendering system does not have to perform expensive operations of loading content from disk to GPU memory often. And so, we efficiently apply caching technique. In our experiments we can automatically obtain desirable caching for rendering system by clustering the multidimensional input points. For example, we can group all input samples on (x_3, x_5) pair. In such a way we consequently render scenes with the same 3D model and the same environment map.

We would like to make a special note on the last point. One may suppose that this optimization could be omitted or achieved in other ways. We consider the following reasons support our position in practice:

- When a researcher writes randomization script (presumably a CV engineer) he most likely does not know details about the rendering system. So effective cache usage should be applied automatically without polluting the randomizer script logic with such optimization issues.
- Some specific procedural approaches could be done directly on GPU (that we actually did for procedural textures). But it is almost impossible to avoid costly interaction with disk or CPU simulation tools (for example, vector displacement, physics based animation or simulation) because for realistic simulation/modeling often we have to rely on a variety of existing CPU based products that could work for minutes or even hours (although giving realistic results).
- This optimization is critical for systems based on the real-time rendering algorithms. They directly feed their result to the neural network on GPU. Even in our case (we use the “offline” GPU path

tracing renderer) we gain acceleration from 1.5 to 3 times for considered case because the input HDR images have high resolution 8Kx4K (this is normal size for most of existing HDR environment assets). And it is simply not possible to load such texture from disk fast enough.

3.1 Suggested Solution in Details

We selected the 3ds Max and the Hydra Renderer [Frolov et al, 2018] due to full-fledged content creation pipeline, advanced GPU rendering and well-specified XML scene representation which is suitable to work with during randomization. The open source Natron software was used as post-processing and compositing tool. Post-processing is needed to produce an additional data (for example, 2d bounding boxes) or to combine the rendered images with photos in case of augmentation (compositing operations and screen-space effects).

3.1.1 Material Setup

Our material editor exploits two observations. Firstly, if we have two materials of the same nature (for example, wood) they actually cannot be interchanged in any arbitrary case. Let's consider a wooden pencil and a wooden floor for example. Both of these objects are constructed from wood but they have different microstructure of the wood and (most important!) they have different scale of texture matrices for applying of texture coordinates. Therefore, applying the same material for floor and pencil would not just look strange but is completely wrong. In the best case we will be able to see small copies of, for example, parquet boards on the pencil when the camera approaches to it. But most likely we will get just some undefined color (especially from far viewpoint). Second observation is the fact that there are many materials that can be applied for certain types of objects only. For example, the TV screen or computer monitor image may appear on an advertising poster and vice versa, but it cannot appear on a cup or a fork. So, there are a lot of objects in the interior scene where it cannot appear and we know such a relationship.

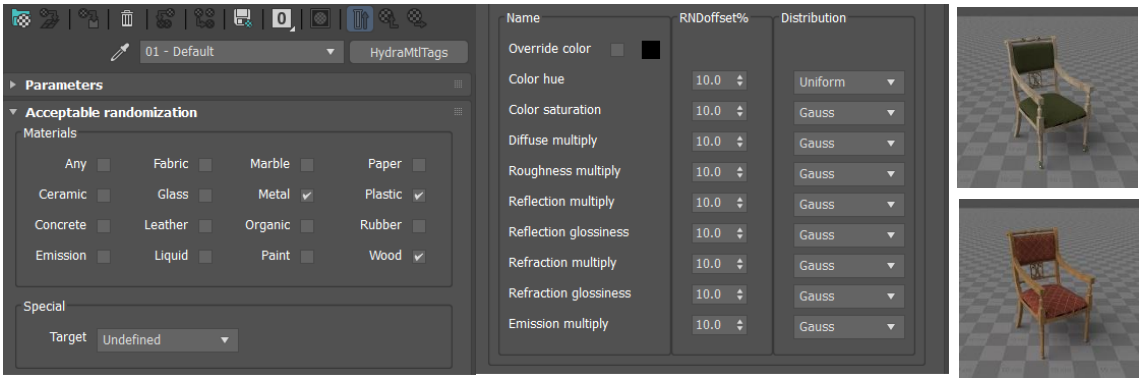

Figure 2. Material randomizer GUI. The "Acceptable randomization" tab is responsible for setting up universal (or tagged) materials. The "Special" tab is responsible for special (or targeted) materials. The middle part of GUI is responsible for distributions for separate material parameters. The right part is preview of object with assigned material

The artist's work was separated in two phases: creation of the material database and filling the database of objects. During *the first phase* artist creates *final materials* and indicates its *tags or target*. A single material is allowed to have several types. It can be combined with other materials or used as a part of some complex material model. This is why *we prefer the term material tag over material type*. And finally, the desired distribution parameters are set separately for each *final material* (see the right part of our GUI on *Figure 2*). The distribution is tested on some simple object (for example, sphere) during the first phase.

During the *second phase* artist takes 3D objects (floor, doors, walls, furniture or any other) and assigns *dummy materials* to them indicating material tags or targets.

3.1.2 Procedural Textures Setup

As we mentioned before, we heavily use procedural textures in our approach. The main goal of procedural textures usage is to provide variety of additional details to the rendered 3d models to produce more realistic images. The rendered objects usually have crisp and clear "idealistic" look. Usually real-life objects do not

preserve the pristine condition for long. Various scratches, dirt and dust are almost always present on them. So, we implemented several procedural textures, simulating effects such as dirt, rust, scratches etc. (*Figure 3*) to improve the realistic look of our generated objects. It is important to note that in some cases the procedural approach is the only possible way. It is so when the input 3D models have not texture coordinates and usage of world-position 3D procedural textures is the only available option.

Figure 3. Examples of applying procedural textures in our pipeline during domain randomization

3.1.3 Interior Sampling

Our content creation pipeline for interiors was assembled from different implementations and is semi-automatic. We broke up “interior sampling” into three different subtasks: layout generation, furniture placement and object placement. For layout generation we have implemented a method based on combination of "dense packing" [Koenig and Knecht, 2014] and "inside-out" [Martin, 2006] methods. For furniture layout we have selected an artist-controlled approach which was implemented via the 3ds Max Scripting. Although existing advanced methods of furniture placement can be applied [Qi et al, 2016; Wang et al, 2018], but for us it was important to use a controlled method which in the same time fits in our sampling concept. Therefore, we first traverse the edges of the office perimeter and place furniture according to 1D pattern. Then an artist draws several binary 2D pattern images to arrange furniture inside the room. We assign four input random numbers to frequency and pattern number for 1D and 2D patterns. During sampling we randomly selected pattern and frequency.

Finally, we also used template-based method for object placement on tables and desktops. Performing some experiments with physics simulation in the beginning of our projects (analogously to [Hodaň et al, 2018]) we found that in practice it is difficult to achieve adequate and reproducible result with it: objects were piled in a heap, fell under a table or were in impossible configurations (like a book on the keyboard or a mouse on the monitor and etc.). We settled on approach when an artist prepares several templates with configured bounding boxes and rotation angles for each type of object on the table or on the floor surface: monitor, system unit, keyboard and etc. Now the objects can appear only within the allowed 2D bounding boxes on the floor or table planes. If a collision is happened, we apply “tabula rasa” method: clearing everything and regenerating it again until there are no collisions [Krauth, 2015]. The resulting generated content can be seen n rendered images on Figure 4 and Figure 1 (right).

Figure 4. Examples of our “interior sampling” approach

4. VALIDATION

We have validated our pipeline for road signs detection and segmentation and cars detection problems. For interiors, though, our validation is not yet ready. For cars we used KITTI-15 dataset [Geiger et al, 2013] for experimental evaluation. It consists of training and test parts. Each contains 200 frames. Since there is no labelling for the test set, we manually labelled bounding boxes for it. We used 50 different 3D models of cars, procedural texture for dirt and rust modeling, and random HDR environments for lighting. Car positions were sampled along the piecewise-linear trajectories that were drawn by artists in 3D (fig. 5, left) for all 200 images. Road plane was reconstructed based on the fact that all images were obtained with single camera parameters. We augmented each image of training set 20 times with 2-5 added random cars in each frame (example on fig. 5, right). As a result, we had 4000 frames in augmented training set. We also made experiments with random horizontal flips (probability of flipping is 0.5) of input images in order to estimate the benefits of our approach over simple methods of increasing training data size. We trained Faster-RCNN [Ren et al, 2015] with ResNet-50 FPN backbone on three training sets: (real images, real + flips, augmented).

We used mAP metric (average areas under curve for different IoU thresholds) for detector quality evaluation. Resulting mAPs are shown in table 1. Precision-recall curves for different IoU thresholds are shown on Figure 6. We clearly can see that our approach with augmentation of frames with rendered cars improves quality of detector.

Figure 5. Examples of augmented KITTI-15. Left column: images with marked trajectories for placing cars, right column: frames augmented with rendered cars

Table 1. mAP metric values for our experiments with cars

Dataset	Example
Real images	38.67 %
Real + flipped	40.30 %
Augmented (ours)	**43.26 %**

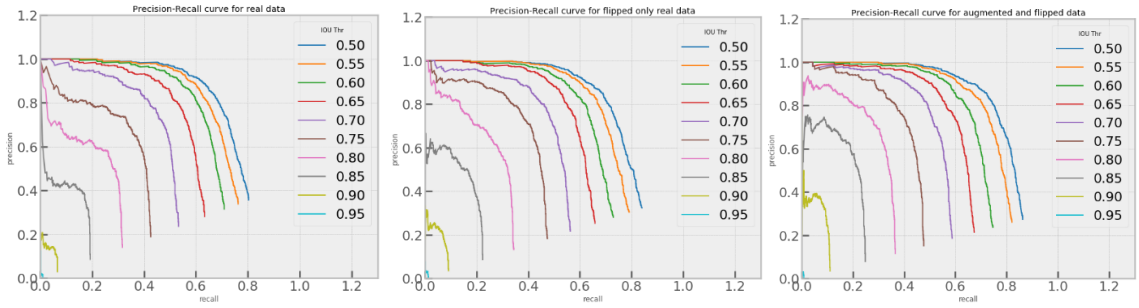

Figure 6. Real (left), real + flipped (middle) and augmented (ours, right) precision-recall curves

5. CONCLUSION

In this work we proposed the content creation pipeline for improving AI training by rendering synthetic datasets and augmenting them to real images. Unlike previous works, where the generator for single problem or the general-purpose pipeline but out of application (like [Denninger et al, 2019]) was proposed, we tested our pipeline in several scenarios, and therefore we can generalize results. Our solution is highly controllable and allows generating datasets with desired distribution of features in a reproduceable manner. This is achieved by the fact, that an artist, while tuning the distributions parameters and creating content, is guided by the requirements of CV researcher, on the one hand, and by the formalized scene sampling process, on the other hand. We always get the same rendered scene sampling the same point of input parameters in multidimensional space. In practice, this turned out to be one of the critical features for further use in neural network training. Because CNN is actually approximate multidimensional distribution, the generator/render should also "sample the scene from distribution" and the content creation pipeline should provide ability for tuning desired distribution, not just generate some random data.

REFERENCES

Alhaija, H. A., Mustikovela, S. K., Mescheder, L., Geiger, A., and Rother, C. 2018. Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes. *International Journal of Computer Vision*, vol. 126, No. 9, pp. 961–972.

Debevec, P., Image-Based Lighting: Tutorial *SIGGRAPH*, 2002, pp. 26–34.

Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A. and Katam, H. 2019. BlenderProc. arXiv preprint *arXiv:1911.01911*.

Frolov, V., Sanzharov, V., Trofimov, M., Pavlov, D. and Galaktionov, V. 2018 Hydra Renderer. Open source GPU based rendering system. https://github.com/Ray-Tracing-Systems/HydraAPI

Garon, M., Sunkavalli, K., Hadap, S., Carr, N., and Lalonde, J. F. 2019. Fast Spatially-Varying Indoor Lighting Estimation, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* pp. 6908-6917

Geiger, A., Lenz, P., Stiller, C. and Urtasun, R., 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, Vol. 32, No. 11, pp.1231-1237.

Hodaň, T., Vineet, V., Gal, R., Shalev, E., Hanzelka, J., Connell, T., Urbina, P., Sinha, S.N. and Guenter, B. 2019. Photorealistic image synthesis for object instance detection. arXiv preprint *arXiv:1902.03334*

Hodan, T., Michel, F., Brachmann, E., Kehl, W., GlentBuch, A., Kraft, D., Drost, B., Vidal, J., Ihrke, S., Zabulis, X. and Sahin, C. 2018. BOP: Benchmark for 6D object pose estimation. *ECCV*, pp. 19-34

Kirsanov, P., Gaskarov, A., Konokhov, F., Sofiiuk, K., Vorontsova, A., Slinko, I., Zhukov, D., Bykov, S., Barinova, O. and Konushin, A. 2019. DISCOMAN: Dataset of Indoor Scenes for Odometry, Mapping And Navigation. arXiv preprint *arXiv:1909.12146*.

Koenig, R., Knecht, K. 2014. Comparing two evolutionary algorithm-based methods for layout generation: Dense packing versus subdivision. *AI EDAM*. Vol 28, No. 3, pp. 285-299

Krauth, Werner. 2015. Advanced Monte Carlo algorithms course lecture. pp 11-12. http://www.lps.ens.fr/~krauth/images/5/50/BadHonnef_2.pdf

Li, Z., and Snavely, N. 2018. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 371–387.

Martin, J. 2006. Procedural house generation: A method for dynamically generating floor plans. *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pp. 1-2.

McCormac, J., Handa, A., Leutenegger, S., and Davison, A. J. 2017. SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2678-2687.

Movshovitz-Attias, Y., Kanade, T., and Sheikh, Y. 2016. How useful is photo-realistic rendering for visual learning. *European Conference on Computer Vision*., pp. 202–217.

Qi, S., Zhu, Y., Huang, S., Jiang, C., and Zhu, S. C. 2016. Human-centric Indoor Scene Synthesis Using Stochastic Grammar. arXiv preprint *arXiv:1808.08473v1*.

Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in neural information processing systems*. pp. 91-99

Risi, S., and Togelius, J. 2019 Increasing Generality in Machine Learning through Procedural Content Generation. arXiv preprint *arXiv:1911.13071*.

Sanzharov, V. and Frolov, V. 2019 Level of Detail for Precomputed Procedural Textures *Programming and Computer Software*, 2019, V. 45, No. 4, pp. 187-195

Shah, S., Dey, D., Lovett, C., and Kapoor, A. 2017. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. arXiv preprint *arXiv:1705.05065*.

Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M. and Funkhouser, T. 2016. Semantic Scene Completion from a Single Depth Image. arXiv preprint, *arXiv:1611.08974*.

Song, Shuran and Funkhouser, Thomas. 2019. Neural Illumination: Lighting Prediction for Indoor Environments, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6911-6919

Sorokin, M., Zhdanov, D., Zhdanov, A., Potemin, I., and Bogdanov N. 2020 Restoration of Lighting Parameters in Mixed Reality Systems Using Convolutional Neural Network Technology Based on RGBD Images. *Programming and Computer Software*, 2020, Vol. 46, No. 3, pp. 203-212

Spick, R.J., Cowling, P. and Walker, J.A. 2019. Procedural Generation using Spatial GANs for Region-Specific Learning of Elevation Data. *2019 IEEE Conference on Games (CoG)*, pp. 1-8.

Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S. and Clarkson, A. 2019. The replica dataset: A digital replica of indoor spaces, arXiv preprint *arXiv:1906.05797*

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S. and Birchfield, S. 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. arXiv preprint *arXiv:1804.06516*, 2018. 2

Tsirikoglou, A., Kronander, J., Wrenninge, M. and Unger, J. 2017. Procedural Modeling and Physically Based Rendering for Synthetic Data Generation in Automotive Applications. arXiv preprint *arXiv:1710.06270*

Valiev I., Voloboy A., Galaktionov V., Improved model of IBL sunlight simulation, *Proceedings of the 24-th int. Spring Conference on Computer Graphics (SCCG)*, 2008, pp.37-42. DOI:10.1145/1921264.1921274

Voloboi A.G., Galaktionov V.A., Kopylov E.A., Shapiro L.Z., Simulation of natural daylight illumination determined by a high dynamic range image, *Programming and Computer Software*, Vol. 32, No. 5, 2006, pp. 284-297. DOI:10.1134/S0361768806050057

Wang, K., Savva, M., Chang, A.X. and Ritchie, D. 2018. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*. Vol. 37, No. 4, pp. 1-14.

Zhang, Y., Song, S., Yumer, E., Savva, M., Lee, J.Y., Jin, H. and Funkhouser, T. 2017. Physically-based rendering for indoor scene understanding using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5287–5295.