BlenderProc: Reducing the Reality Gap with Photorealistic Rendering

Maximilian Denninger*, Martin Sundermeyer*, Dominik Winkelbauer*, Dmitry Olefir*, Tomas Hodan†, Youssef Zidan*, Mohamad Elbadrawy*, Markus Knauer*, Harinandan Katam*, Ahsan Lodhi*

*Institute of Robotics and Mechatronics, German Aerospace Center (DLR), {first}.{last}@dlr.de
†Visual Recognition Group, Czech Technical University in Prague, hodantom@cmp.felk.cvut.cz

***Abstract*—BlenderProc is an open-source and modular pipeline for rendering photorealistic images of procedurally generated 3D scenes which can be used for training data-hungry deep learning models. The presented results on the tasks of instance segmentation and surface normal estimation suggest that our photorealistic training images reduce the gap between the synthetic training and real test domains, compared to less realistic training images combined with domain randomization. BlenderProc can be used to train models for various computer vision tasks such as semantic segmentation or estimation of depth, optical flow, and object pose. By offering standard modules for parameterizing and sampling materials, objects, cameras and lights, BlenderProc can simulate various real-world scenarios and provide means to systematically investigate the essential factors for sim2real transfer.**

I. Introduction

With the advent of deep neural networks, the demand for accurately annotated training images has grown dramatically, resulting in large real and synthetic datasets [1]–[4]. However, the neural networks are robust only when trained on sufficient in-domain data. In dynamic robotic environments and tasks like object pose estimation, the labeling effort often prohibits the use of real annotated data.

Learning in simulation is appealing, especially when considering methods that let the robot automatically generate the required models. On the other hand, the gap between the synthetic training and real test domains (*a.k.a.* the sim2real gap) prevents algorithms to generalize to the real domain. However, several works [5]–[7] have already shown that more realistic training images lead to better sim2real results.

BlenderProc [8] offers a fully configurable pipeline for procedurally generating scenes and rendering photorealistic training images. The pipeline is built on top of Blender [9], an open-source project which offers a variety of relevant features through a stable API with years of optimization and an active community. The Blender's physically accurate ray tracer, *cycles*, is used for rendering. A typical run of the pipeline consists of loading a set of objects, sampling object poses with the aid of a physics engine, randomizing object materials, sampling cameras and lights, and rendering that generates modalities such as color, depth, surface normals, semantic segmentation or optical flow. All these contributions go far beyond the Blender feature set and are fully automated using a single, portable config file.

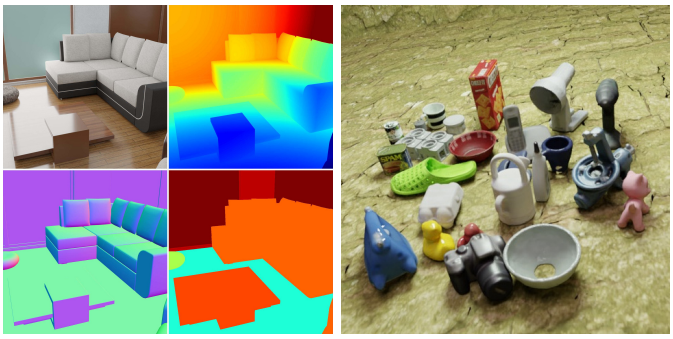

Fig. 1: Photorealistic images and labels synthesized with BlenderProc. *Left:* Color, depth, surface normal and semantic labels. *Right:* An example training image annotated with 6D object poses used in the presented sim2real experiment.

II. Related Work

A crucial feature of BlenderProc is the possibility to render with a ray tracer, which gives significantly better results than any rasterizer can produce because it resembles the actual physical process of light transport. Equally important are material properties which are determined by factors like roughness and specularity. Even if they are not accurately modelled, we show that they can still be used for physically plausible domain randomization. In Table I we compare features with the NVIDIA Deep learning Dataset Synthesizer (NDDS) [10], ViSII - A VIrtual Scene Imaging Interface [11], AI Habitat [12] and Stilleben [13].

	NDDS[10]	ViSII[11]	Habitat[12]	Stillleben[13]	**BP**
real ray tracing	⊗	✓	⊗	⊗	✓
semantic segm.	✓	✓	✓	✓	✓
depth rendering	✓	✓	✓	✓	✓
optical flow	⊗	✓	✓	⊗	✓
surface normals	⊗	✓	✓	✓	✓
object pose	✓	(✓)	✓	✓	✓
bounding box	✓	(✓)	✓	⊗	✓
physics module	✓	✓	✓	✓	✓
camera sampling	✓	✓	✓	✓	✓
pbr materials	⊗	✓	⊗	⊗	✓
docu for each fct.	⊗	✓	✓	⊗	✓
editing via GUI	✓	⊗	⊗	⊗	✓

TABLE I: Features in BlenderProc (BP) and other simulators (2020).

In comparison to these four simulators, BlenderProc offers richer features towards simplified sim2real transfer. Real-time capability is often not required and the whole BlenderProc pipeline takes less than 2 seconds for images like in Fig. 1 on a single GPU machine. Large datasets (>100K images) can be generated on an 8-GPU server in a few hours. Furthermore, we already support loaders for a variety of datasets: SceneNet [14], ShapeNet [2], Replica [15], SUNCG [3], T-LESS [16], Linemod [17], Linemod-Occluded [18], MVTec ITODD [19], HomebrewedDB [20], YCB-Video [21], Rutgers APC [22], Doumanoglou et al. [23], Tejani et al. [24], TUD Light [4], and Toyota Light [4]. Additionally, we are already in the process of integrating 3D-Front and 3D-Future [25], [26].

III. Experiments

In the following, we investigate the sim2real capability of photo-realistic data generated by BlenderProc on the task of instance segmentation on the LineMod-Occluded (LM-O) dataset [18]. The dataset provides eight object models with imperfect geometry and texture acquired with KinectFusion-like techniques. We train a Mask R-CNN with Resnet50 backbone on 50K synthetic RGB images that are originally generated for the BOP Challenge 2020 [4].

The objects are placed into an open cube mapped with randomized PBR textures [27]. Object material properties like metallicness, roughness and specularity are also randomized. For details and exact data reproduction we refer to the BlenderProc config file [28]. An example of the synthetic training data is depicted in Fig. 1 on the right.

To assess the sim2real performance of our data, we compare it to the popular render&paste technique using OpenGL renderings on real backgrounds combined with strong domain randomization [29], using 50K images as well. Qualitative test results on LM-O are shown in Fig. 2. The Mask R-CNN models were pretrained on COCO and fine-tuned on the respective datasets. As found by Hinterstoisser et al. [30], when fine-tuning on the domain-randomized OpenGL data, freezing the complete backbone is necessary to avoid overfitting to the synthetic domain. This is accounted to the domain gap of low-level image statistics between real and OpenGL data. Interestingly, training on synthetic data generated by BlenderProc allows unfreezing the backbone which even slightly improves results. With default hyperparameters from [31], we achieve a clear improvement from 26.2 (OpenGL+DR) to 33.8 (BlenderProc) Mask mAP50 (+7.6).

Furthermore, we evaluated the reconstruction of surface normals from color images. We tested here a typical U-Net architecture, which was trained on simulated images from BlenderProc in SUNCG scenes and we have shown some results here on the Replica dataset, see Fig. 3.

(a) OpenGL + real backgrounds + domain randomization (DR). (b) Training on photorealistic images from BlenderProc.

Fig. 2: Sim2Real Instance Segmentation on LM-O. Notice that the OpenGL + DR scheme results in missed instances, false positives and failures under strong occlusions. Using the BlenderProc data for training all 8 LM-O instances are segmented well even under challenging conditions.

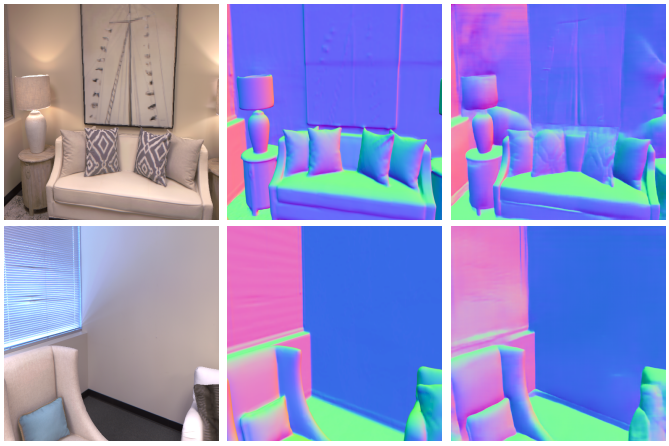

Fig. 3: From left to right we show here the color input image, the GT surfaces normals and the reconstructed normals. The network was solely trained on simulation data from BlenderProc on SUNCG scenes.

IV. Conclusion

In this work, we have shown sim2real results on instance segmentation using BlenderProc, that compare favorably against the standard render&paste approach using OpenGL renderings on real images and strong image-based domain randomization. Additionally, we provided qualitative surface normal reconstruction results.

The transfer from simulation to the real world has become significantly easier with BlenderProc. Physically-plausible domain randomization, like the randomization of material properties and light, enables sim2real transfer even with imperfect models like the LineMOD objects. This is crucial to replace accurate, manual modelling with automatized real2sim methods performed by robots themselves to close the real2sim2real cycle.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[2] 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," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[3] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2017.28

[4] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother, "BOP: Benchmark for 6D object pose estimation," *ECCV*, 2018.

[5] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. Sinha, and B. Guenter, "Photorealistic image synthesis for object instance detection," *ICIP*, 2019.

[6] Z. Li and N. Snavely, "Cgintrinsics: Better intrinsic image decomposition through physically-based rendering," in *ECCV*, 2018.

[7] 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 *CVPR*, 2017.

[8] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, "Blenderproc," 2019.

[9] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2019. [Online]. Available: http://www.blender.org

[10] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield, "NDDS: NVIDIA deep learning dataset synthesizer," 2018, https://github.com/NVIDIA/Dataset_Synthesizer.

[11] N. V. Morrical, *ViSII - A VIrtual Scene Imaging Interface*, 2020. [Online]. Available: https://github.com/owl-project/ViSII

[12] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[13] M. Schwarz and S. Behnke, "Stillleben: Realistic scene synthesis for deep learning in robotics," *IEEE International Conference on Robotics and Automation (ICRA), May 2020*, 2020.

[14] A. Handa, V. Patraucean, S. Stent, and R. Cipolla, "SceneNet: An annotated model generator for indoor scene understanding," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2016. [Online]. Available: https://doi.org/10.1109%2Ficra.2016.7487797

[15] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[16] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *WACV*, 2017.

[17] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Computer Vision – ACCV 2012*. Springer Berlin Heidelberg, 2013, pp. 548–562. [Online]. Available: https://doi.org/10.1007%2F978-3-642-37331-2_42

[18] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 536–551. [Online]. Available: https://doi.org/10.1007%2F978-3-319-10605-2_35

[19] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, and C. Steger, "Introducing MVTec ITODD — a dataset for 3d object recognition in industry," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, oct 2017. [Online]. Available: https://doi.org/10.1109%2Ficcvw.2017.257

[20] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic, "HomebrewedDB: RGB-d dataset for 6d pose estimation of 3d objects," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, oct 2019. [Online]. Available: https://doi.org/10.1109%2Ficcvw.2019.00338

[21] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, jun 2018. [Online]. Available: https://doi.org/10.15607%2Frss.2018.xiv.019

[22] C. Rennie, R. Shome, K. E. Bekris, and A. F. D. Souza, "A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1179–1185, jul 2016. [Online]. Available: https://doi.org/10.1109%2Flra.2016.2532924

[23] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6d object pose and predicting next-best-view in the crowd," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2016.390

[24] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 462–477. [Online]. Available: https://doi.org/10.1007%2F978-3-319-10599-4_30

[25] H. Fu, B. Cai, L. Gao, L. Zhang, R. Jia, B. Zhao, and H. Zhang, *3D-FRONT*, Alibaba-inc, 2020. [Online]. Available: https://tianchi.aliyun.com/specials/promotion/alibaba-3d-scene-dataset?spm=5176.12282016.0.0.6367f6f26nD9d5

[26] H. Fu, R. Jia, L. Gao, M. Gong, B. Zhao, S. Maybank, and D. Tao, *3D-FUTURE*, Alibaba-inc, 2020. [Online]. Available: https://tianchi.aliyun.com/specials/promotion/ijcai-alibaba-3d-future-workshop

[27] L. Demes, *CC0 Textures*, 2017. [Online]. Available: https://cc0textures.com/

[28] "BlenderProc config file for the linemod dataset." https://github.com/DLR-RM/BlenderProc/blob/master/examples/bop_challenge/config_lm_upright.yaml.

[29] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1301–1310.

[30] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[31] F. Massa and R. Girshick, "maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch," https://github.com/facebookresearch/maskrcnn-benchmark, 2018, accessed: [Insert date here].