Automated Dataset Generation with Blender for Deep Learning-based Object Segmentation

Conference Paper · March 2022

DOI: 10.1109/SAMI54271.2022.9780790

2 authors:

Artúr István Károly
Óbudai Egyetem
14 PUBLICATIONS **140** CITATIONS
SEE PROFILE

Péter Galambos
Óbudai Egyetem
139 PUBLICATIONS **1,457** CITATIONS
SEE PROFILE

Automated Dataset Generation with Blender for Deep Learning-based Object Segmentation

Artúr I. Károly[1]
Antal Bejczy Center for Intelligent Robotics
Óbuda University, Bécsi u. 96/B. H-1034
Budapest, Hungary
artur.karoly@irob.uni-obuda.hu

Péter Galambos
Antal Bejczy Center for Intelligent Robotics
Óbuda University, Bécsi u. 96/B. H-1034
Budapest, Hungary
peter.galambos@irob.uni-obuda.hu

***Abstract*—"Give a man a synthetic dataset and you help him for a project; teach a man to generate his own dataset and you help him for a lifetime." Developments in digital technology led to an unprecedented increase in the amount of data in industry and in science as well, including the field of computer vision. This enabled the evolution of convolutional neural networks (CNNs) and other deep learning (DL) architectures that require a large amount of data for their training. Even though the large-scale collection of visual data is relatively cheap and easy to implement, the annotation of the collected data is a time-consuming process. Although, with the use of pre-trained models only a small-scale fine-tuning dataset is required, the labeling of only a few hundred or few thousand images for object segmentation or instance segmentation is still a laborious task, if done manually. Recent trends suggest that synthetic data will play a significant role in the future of DL and there are numerous research papers promoting various synthetic datasets and/or models trained on them. Unfortunately, there is much less information on how to create such a dataset and what are the important points one has to keep in mind when creating their own. In this paper, we summarize some of the available tools for creating a synthetic dataset for object segmentation and provide a detailed example using the Blender 3D creation suite. During our example, we attract attention to the vital points and considerations for automatic dataset generation for object segmentation.**

I. Introduction

Computer vision have massively increased in popularity since the recent advancements in machine learning algorithms, such as CNNs and DL. Visual data is rich in information and relatively easy to obtain, so it is utilized in a wide variety of tasks, such as self-driving vehicles, robotics [1], medical diagnostics [2] etc. The methods for extracting this information from the image data often utilize machine learning technologies, such as DL models. Such models rely on a large amount of labeled data for their training process. While usually a large amount of image data is either already available, or can be easily obtained for most domains, the labeling of the data is a tedious and time-consuming process. Luckily, in most cases DL approaches are able to take advantage of annotated data from a different domain. This enables such methods, to first train on an appropriate large-scale dataset and then be fine-tuned on a task-specific smaller annotated dataset. This process is often referred to as transfer learning in the context of CNNs [3], [4], [5].

[1]Artúr I. Károly is the PhD students of the Doctoral School of Applied Informatics and Applied Mathematics at Óbuda Universty.

A prerequisite of transfer learning is the existence of easily accessible annotated datasets, such as the famous ImageNet [6], COCO [7], KITTI [8] or Pascal VOC [9] datasets. Among others, models pre-trained on these datasets are readily available for general image processing and image classification (ImageNet), object detection (Pascal VOC), object segmentation (COCO) and self-driving (KITTI) scenarios. However, one cannot expect state-of-the-art accuracy on a specific target domain from DL models which are solely trained on these general large-scale datasets. As a result of this, it is common to set a small training dataset up from the task-specific domain for fine-tuning purposes. In case of image classification the preparation of such a dataset can be done with minimal effort. However, for a more complicated task such as image or instance segmentation, the manual preparation of even a few hundred or a few thousand annotated samples is still a huge undertaking. As a result, the use of synthetic data is a tempting alternative to manual annotation. It offers the benefits of cheap, fast and accurate label generation, ease of control over bias and data distribution and the possibility to generate inherently anonymous data. As a result of this, the ratio of synthetic data out of all the data used for the training of AI models is expected to raise significantly in the following years according to Gartner [10]. There are numerous ways for synthetic data generation, such as regression methods, decision tree/random forest, or neural network solutions, such as generative adversarial networks (GANs) or variational autoencoders (VAEs) [11], [12], [13]. A popular approach is the use simulators (and physics engines if needed). The benefit of using a simulation is a very precise control over the environment and thus the domain, as well as the guaranteed perfect quality of the annotations.

There are many examples of recent research successfully utilizing simulated synthetic data for the training of DL models for various purposes, such as robotic manipulation and object pose estimation [14], [15], segmentation of urban scenes for autonomous driving [16], or crop and weed segmentation for agriculture [17]. All these scientific results suggest that it is worth using synthetic data, some even make the generated dataset publicly available. However, a ready-made dataset is usually limited to its specific purpose or use-case and can hardly ever be directly applied to a new domain. The robotic grasping and 3D object pose estimation datasets can be useful for the training of a DL model for robotic bin picking, but if the objects

978-1-6654-9704-6/22/$31.00 ©2022 IEEE

to pick are unique, or just not part of the datasets, a fine tuning step has to be performed nevertheless.

This problem can be resolved by creating one's own specialized synthetic dataset for their specific needs. Unfortunately, the articles promoting or leveraging synthetic data usually do not elaborate the in-depth details of the dataset generation, which indeed involves several important considerations and pitfalls. As a result of this, synthetic dataset generation with simulation is likely a non-trivial task for a large portion of researchers. This paper aims to provide an overview of some available tools for synthetic data generation and a simple example using the free and open-source Blender 3D creation suite [18]. The most important considerations and technical solutions are introduced as part of the example to provide a guideline for anyone aiming to create their own synthetic dataset.

II. Synthetic data and AI services

Since synthetic data is expected to become more and more significant in the future and there is a high demand for AI solutions in various industries, there is a developing market for synthetic data providers. Recently new services started to pop up that provide synthetic data generation for various domains as well as additional AI related services.

The **Sky Engine AI Platform** offers a full-stack workflow for DL solutions from model development to integration. Their demos showcase various scenarios such as scene segmentation in a warehouse environment or telecommunication antenna detection. The synthetic data is generated from a simulated scene with a physics engine and various virtual sensor modalities.

Datagen provides simulated, photorealistic, domain-specific data for areas where human perception is largely involved, such as AR or VR, robotics applications, smart stores etc. Their data generation process involves a GAN-based solution to sample novel 3D assets on top of the original ones (3D models created by designers or scanned from real-world objects).

CVEDIA's SynCity is a simulation platform that offers high-quality photorealistic synthetic data for DL training and validation with a focus on driving scenarios. Their 2020 white paper compares generalizability of models trained on their synthetic dataset against models trained on other open datasets [19]. They found that the model trained on their synthetic data could generalize better than the others, which is likely due to the deliberate control over the biases and variances in their synthetic dataset.

There are also solutions for elaborate physics simulations for the sake of reinforcement learning for robotics such as NVIDIA Isaac Gym, but these generally put more focus on sensor modalities other than vision.

A common feature of these synthetic data services is that they are centered around applications and thus target companies needing DL solutions for specific use cases. As a result of this, it is really hard to gain hands-on experience in these systems. An open, lightweight and easy-to-implement solution for small-scale research purposes is yet to be seen. Our proposal aims to fill in this gap by by providing an example of a small-scale synthetic data generation project.

III. Preliminary considerations

Using a synthetic dataset for training a DL model, which later on performs inference on real-world data, has its difficulties. These difficulties usually arise from the differences in the simulated and the target domains, such as neglected physics interactions, simplified world model etc. These are even more significant in reinforcement learning, since apart from perception, interaction with the environment is also required [20], [21]. However, the quality of the synthetic dataset is still important in perceptive tasks as well, just as [19] showed. That is why, there are things to learn from the sim-to-real approaches in reinforcement learning, as they can provide requirements that the simulation software used as the basis of the synthetic data generation should fulfill.

The two major takeaways are close-to-reality simulation and domain randomization [20], [21], [22]. In case of object segmentation tasks, the close-to-reality simulation is fulfilled by photorealism. So the first requirement for the simulation software is that it must produce photorealistic renders.

Randomizing certain parameters of the scene, such as lighting conditions, textures, background etc. is proven to improve generalization and thus the adaptation to the real-world domain [22]. This method is referred to as domain randomization. The basis simulation software has to have the functionality to randomize such aspects of the scene as easily as possible.

Apart from randomization and photorealism, the inclusion of physics simulation is a soft-requirement, as one might want to generate organic looking piles of objects (e.g. for robotic bin-picking) that are hard to manually set up.

Last but not least, accessibility and setup costs should also be taken into account. A lightweight simulation software should be preferred over a heavy and hard-to learn one. Open source software has the benefits of versatility and ease of access. A well documented open source simulation software could be a good basis for any synthetic data generation pipeline.

IV. Example: mushrooms

According to the considerations in section III. we chose the Blender 3D suite as the basis for our synthetic data generation workflow. Blender is an open source software with a large and active community [18]. This means, that there are plenty of available online materials for learning its functionalities. Since, the usage of Blender is not in the scope scope of this paper, we kindly refer to said materials, should the reader encounter unknown terminology or have additional questions regarding Blender functionalities. Blender is a general 3D creation tool, so it is not constrained to a single domain type, such as some driving or robotics simulators would be. Its main purpose is creating computer graphics, so it has plenty of functionalities for the manipulation of a visual scene, such as including and modifying 3D object models, lighting and camera setups, geometry modifications, textures and shading, image post-processing etc. It can be used to create photorealistic renders and it includes physics simulation using the Bullet physics engine. It also has a Python API which makes it easy

to integrate into a DL training workflow, since most of the DL frameworks support the Python programming language.

The example task on which we demonstrate the creation of a synthetic data generation tool is mushroom segmentation. The real-world setup includes a single camera and a light source suspended above a mushroom growing box. The task is to **segment** and separate **each mushroom individually** on the images captured by the camera. The purpose is to follow the growth process of the mushrooms in order to aid their harvesting process. There are models already for the simulation of tree growth [23], but according to our knowledge, there isn't a similar publicly available model for simulating mushroom growth. In the future, the collected and segmented data from our solution may help develop such a model for mushroom growth prediction as well. A reference image from the real-world mushrooms can be seen in Figure 1.

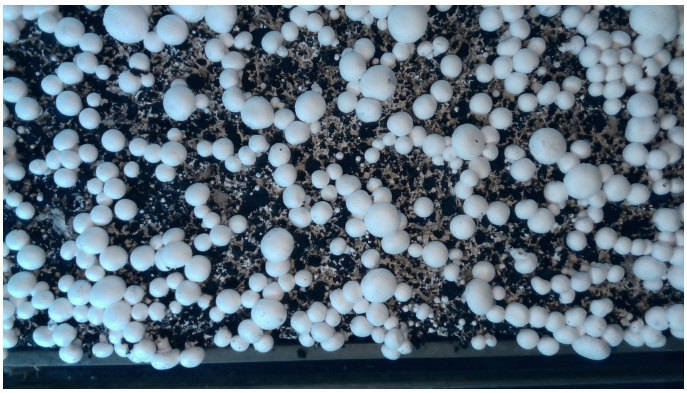

Fig. 1. Real reference image

For the synthetic dataset generation first a starting scene was created. A simple mushroom model was defined by combining a cylinder and a half sphere and performing additional sculpting steps on the mushroom head to make it less uniform. Figure 2. shows our simple mushroom model.

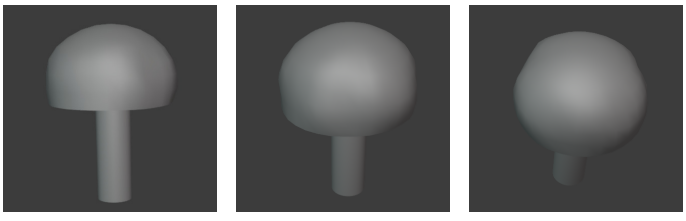

Fig. 2. Simple 3D model of a mushroom

A plane in the scene serves as the background, and a camera and a light source were placed above the plane. The original mushroom model was hidden underneath the plane so it would not be in the camera's view. In order to have a similar distribution of mushrooms as seen in the reference image, a geometry modifier was applied to the plane using the geometry nodes of Blender. With the point distribute and point instance nodes, mushrooms were scattered on the surface of the plane. The mushrooms are distributed according to a Poisson Disk distribution, because that yielded the most organic-looking results (with random uniform distribution we experienced a clustering effect). The resulted scene is shown in Figure 3/a. The creation of the scene most likely will be unique for different use cases so it is not elaborated in detail, but instead we refer to the official Blender documentation and additional community tutorials.

As it can be seen however, all the mushrooms scattered on the surface of the plane are the same. For a realistic image the mushrooms should have various sizes and orientations. This is where randomization can be introduced. The methods shown here can also be applied to changes in textures, materials or other properties such as lighting.

For the randomization of the scene geometry the attribute randomize geometry node was used. Selecting the scale and rotation of the mushroom instances for randomization already yields a convincing result. The problem however, is that the distribution and the attribute randomization will always be the same each time the same seed is passed to the point distribute and the attribute randomize nodes. In order to have multiple different outcomes, the seed values should be changed each time a new frame is rendered (so each frame will be unique). The most straightforward way to do this, is to add simple values to the seed inputs of the respective nodes and add keyframes to the value of each node. Because of the added keyframes, these values can be animated, which means when rendering an animation, different values can be assigned to the seeds at each frame. We used the graph editor and added a noise modifier to each keyframed value to ensure that they would be different for each frame. An important thing to note is that the seed input of geometry nodes expects integers, so the amplitude of the noise modifier has to be increased greatly. Figure 4. shows the node graph after the randomization.

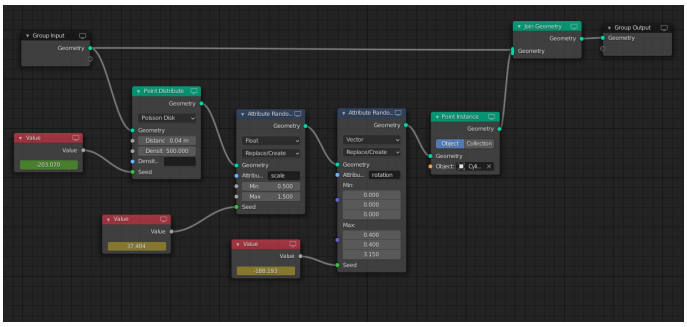

Fig. 4. Complete node graph

The scene after the randomization can be seen in Figure 3/b.

In order to create a photorealistic render a realistic mushroom texture was created using Blender's Principled BSDF shader node and an image texture was assigned for the background plane. The image for the background texture was taken by the real-world setup before the mushrooms started growing. After this, a final fine-tuning of the lighting conditions was performed based on the reference image. These are all the steps needed for the randomized photorealistic image generation. A photorealistic rendered image can be seen in Figure 3/c.

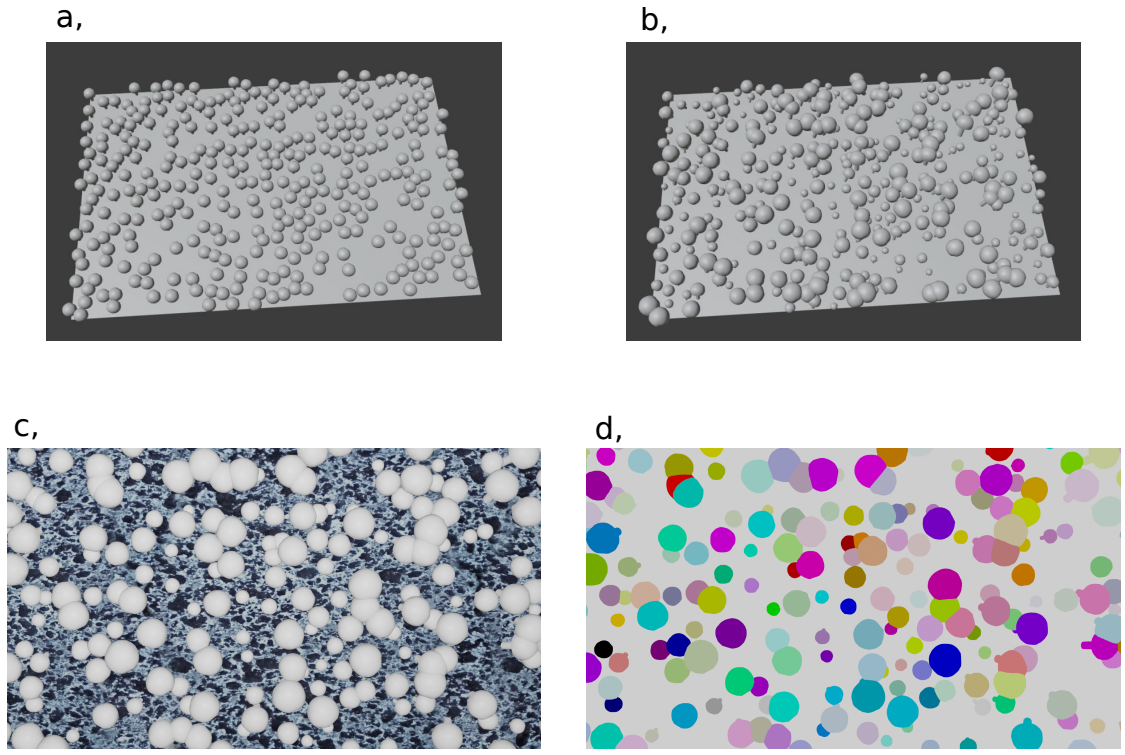

Fig. 3. Scene and render results; a: basic scene without randomization, b: scene with randomized mushroom size and orientation, c: an example photorealistic render, d: instance masks corresponding to 'c'

The last step of the data generation pipeline is to automatically render each frame and to create the corresponding masks. For this we used Blender's Python API. First a photorealistic rendered image is created and saved to disk. After that, since each individual mushroom needs to be distinguished, the script makes each mushroom instance a real object and assigns unique color to them using their object properties. Then a flat-colored viewport render is performed and the resulting image is saved to disk (this serves as the mask, since each mushroom is represented by a unique color). Finally, the mushroom objects are deleted (except the original one) and the whole process starts from the beginning. This process is also expressed in an algorithmic form in algorithm 1. A resulted mask can be seen in Figure 3/d.

Algorithm 1 Frame generation

Require: start_frame, num_frames
  $i = 0$
  while $i <$ num_frames **do**
    set frame to start_frame $+ i$
    render photorealistic image
    save photorealistic image
    mushroom_objects = make_instances_real()
    for mushroom_object in mushroom_objects **do**
      mushroom_object.color = unique_color()
    end for
    render viewport
    save viewport render
    for mushroom_object in mushroom_objects **do**
      delete(mushroom_object)
    end for
    $i+ = 1$
  end while

V. Conclusion

The benefits of using a synthetic dataset for the training of DL models is demonstrated in many previous research papers. However, there is hardly any material concerning the creation of such synthetic datasets. Although there are online services, they usually target companies needing solutions for specific tasks. In this paper we provide an example for creating a synthetic data generation tool for DL using Blender. We account the main considerations one have to take when creating their own synthetic data generation workflow. We hope that our example will be useful for speeding up smaller DL-based research projects in the future.

Acknowledgment

Péter Galambos and Artúr I. Károly thankfully acknowledge the financial support of this work by the project no. 2019-1.3.1-KK-2019-00007 implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK funding scheme.

The research was supported by the Eötvös Loránd Research Network Secretariat under grant agreement no. ELKH KÖ-

40/2020 ('Development of cyber-medical systems based on AI and hybrid cloud methods').

Artúr I. Károly is supported by the UNKP-21-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund.

Péter Galambos is a Bolyai Fellow of the Hungarian Academy of Sciences.

Péter Galambos is supported by the UNKP-21-5 (Bolyai+) New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund.

REFERENCES

[1] A. Vokhmintcev and M. Timchenko, "The new combined method of the generation of a 3d dense map of evironment based on history of camera positions and the robot's movements," *Acta Polytechnica Hungarica*, vol. 17, no. 8, pp. 95–108, 2020, doi: https://doi.org/10.12700/APH.17.8.2020.8.7.

[2] F. Fatemeh, Rashidi, G. Janos, L., and A.-Q. Ikhlas, "Box-trainer assessment system with real-time multi-class detection and tracking of laparoscopic instruments, using cnn," *Acta Polytechnica Hungarica*, vol. 19, no. 2, pp. 7–27, 2022.

[3] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[4] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[5] M. Huh, P. Agrawal, and A. A. Efros, "What makes imagenet good for transfer learning?" *CoRR*, vol. abs/1608.08614, 2016. [Online]. Available: http://arxiv.org/abs/1608.08614

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[7] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[10] S. Castellanos, "Fake it to make it: Companies beef up AI models with synthetic data," *The Wall Street Journal*. [Online]. Available: https://www.wsj.com/articles/fake-it-to-make-it-companies-beef-up-ai-models-with-synthetic-data-11627032601

[11] A. Dandekar, R. A. M. Zen, and S. Bressan, "A comparative study of synthetic dataset generation techniques," in *Database and Expert Systems Applications*, S. Hartmann, H. Ma, A. Hameurlain, G. Pernul, and R. R. Wagner, Eds. Cham: Springer International Publishing, 2018, pp. 387–395.

[12] B. Bhattarai, S. Baek, R. Bodur, and T.-K. Kim, "Sampling strategies for gan synthetic data," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 2303–2307.

[13] Z. Wan, Y. Zhang, and H. He, "Variational autoencoder based synthetic data generation for imbalanced learning," in *2017 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2017, pp. 1–7.

[14] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," *arXiv preprint arXiv:1809.10790*, 2018.

[15] J. Tremblay, T. To, and S. Birchfield, "Falling things: A synthetic dataset for 3d object detection and pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2038–2041.

[16] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3234–3243.

[17] M. Di Cicco, C. Potena, G. Grisetti, and A. Pretto, "Automatic model based dataset generation for fast and accurate crop and weeds detection," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5188–5195.

[18] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[19] M. Benitez, "Quantifying generalization - CVEDIA detection technology vs. seven open source datasets," CVEDIA PTE Ltd, White Paper, 2020.

[20] Y. You, X. Pan, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," *CoRR*, vol. abs/1704.03952, 2017. [Online]. Available: http://arxiv.org/abs/1704.03952

[21] Q. Vuong, S. Vikram, H. Su, S. Gao, and H. I. Christensen, "How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies?" *CoRR*, vol. abs/1903.11774, 2019. [Online]. Available: http://arxiv.org/abs/1903.11774

[22] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[23] B. T. Tóth and S. Szénási, "Tree growth simulation based on ray-traced lights modelling," *Acta Polytechnica Hungarica*, vol. 17, no. 4, 2020, doi: https://doi.org/10.12700/APH.17.4.2020.4.12.