Photorealistic Rendering in the Process of Training Deep Neural Networks

Maciej Stefańczyk[1][0000−0001−9948−6319] and Daniel Chmielewiec[1]

Warsaw University of Technology
Institute of Control and Computation Engineering
Nowowiejska 15/19, 00-665 Warsaw, Poland
maciej.stefanczyk@pw.edu.pl

Abstract. This article aims to analyze the possibilities of using photorealistic rendering to create a training dataset for deep neural networks (object detection). Synthetic datasets are compared with the traditional one made by means of photos. The results answer the question of whether synthetic data can substitute photos in that case. The evaluation and advantages of such an approach are presented.

Keywords: object detection · synthetic training data · rendering

1 Introduction

In recent years, there has been a huge shift towards using deep learning methods for all computer vision tasks. With the change towards the data-driven approach, where the model is no longer explicitly defined, there is a need for big training datasets so that the automatic methods can finally come up with an accurate solution. As a result, a lot more human work is needed for the tedious task of collecting the training data: taking the pictures, processing them, and creating the labels takes a lot of time. Moreover, even with the big existing datasets, there is always a scenario in which someone needs to recognize novel objects, not available in those.

The observed speed-up in the field of deep learning is mainly due to the progress in the hardware: general-purpose GPU computing allowed for the calculations to be done in a reasonable time. The same technological hardware progress is a blessing for the computer graphics field. Photorealistic scenes can now be rendered in virtually no time, making the computer games look better every year. This fact and the availability of many 3D modeling and rendering tools opened the possibility of creating synthetic datasets that look as good as the actual photos.

To create the photorealistic rendering, we need models of the objects we want to use. The whole idea of using synthetic data is useless if the process of model acquisition/creation takes much more time than creating the regular dataset. The models of the objects should resemble the geometry of the original objects, as well as should be properly textured. The easiest way of acquiring

the models is to use the existing datasets. One of the most popular sources is the BOP challenge [2], where there are 11 datasets available, each containing (amongst others) the subpart with the 3D models and textures. Those datasets are, however, mostly meant to be used as a benchmark (as the name of the challenge suggests), so it is unlikely that the required objects can be found there.

There are three main aspects of the presented topic. First of all, the models of the objects are required, along with good quality textures and environment scenes. After that, the actual render must be done. For this purpose, either the general-purpose physical-based renderers or the game engines capable of real-time rendering are used (those two distinct approaches are tested in this article). Finally, the evaluation should be done on the real-life datasets to check whether the trained networks can generalize the knowledge to the actual scenario.

2 Experimental setup

Objects for the experiments were selected to cover the popular shapes and sizes, with different texture features, like glossiness or transparency. Models of ten objects were prepared in Blender software, with textures gathered either by scanning existing ones or downloading them from the internet. The mesh complexity differs from very simple boxes, through flat bags, to more complex bottles and cans (fig. 1).

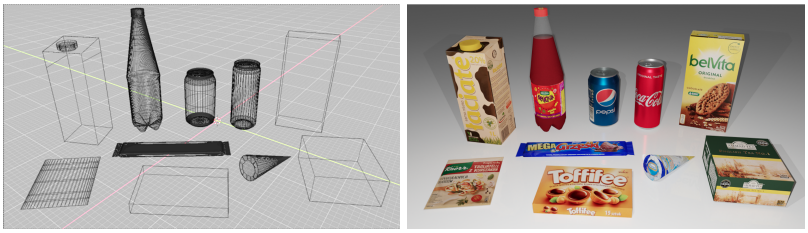

Fig. 1. Preview of the created object models

For the dataset creation process, two popular tools were selected. Both of them outputs the same kind of results, i.e., the RGB image, pixel-perfect segmentation maps, bounding boxes, and others. NVIDIA Deep Learning Dataset Synthesizer (NDDS) is a UE4 plugin from NVIDIA [4]. It takes advantage of the rendering engine in UE4, which is not physically accurate but allows for much faster data generation. The **NDDS** dataset was created using this tool. BlenderProc is an open-source and modular pipeline for rendering photorealistic images of procedurally generated 3D scenes [1]. It uses the Blender as the rendering engine. The pipelines are created using the simple config files, where each step of the process is defined. Images are generated with a physical-based path tracing renderer. It was used to create the **BP** dataset.

For comparison purposes, the third dataset (**photo**) was manually created. For each object, multiple photos were taken (taking into consideration different positions and lighting conditions). From the pictures, objects were cut out and overlaid on different backgrounds in various places (with additional augmentations). Sample pictures from the datasets are presented in fig. 2. Each dataset contained 1000 images per object.

a)  b)  c) 

Fig. 2. Sample images from (a) BP, (b) NDDS, (c) photo dataset

For testing purposes, the additional dataset was created using photos taken in real-life. Three subsets were prepared: easy, with objects clearly visible in good lighting conditions; moderate, with some occlusions, and hard, with objects farther from the camera and worse lighting conditions (fig. 3).

Fig. 3. Sample images from the testing dataset

3 Results and discussion

3.1 Dataset preparation time

The first aspect of the whole process is how long does it take to create every dataset. The preparation of the 3D models and configuring the rendering pipeline took an hour per object on average. Rendering using NDDS achieved around 10 thousand images per hour, BP was ten times slower, averaging 1 thousand images per hour. Because of this high discrepancy in creation time, the smaller BP300 dataset was created, sampling 30% of data from the BP. That made it possible to compare the results based on the similar dataset creation time. In most cases, we don't have hard limits on the dataset creation time, but one can imagine a scenario in which the service robot has to learn a new object, and the time for creating the dataset plays a crucial role here. The photo dataset took 15 hours to take the photos and crop them manually and an additional 1 hour to generate the augmented dataset.

3.2 Detection accuracy

For the main test, the YOLOv3 [3] pretrained on a COCO dataset was selected as a detection network. The training was conducted until convergence on each of the datasets, and the mAP on the testing sets was used as a basis for final comparison. The results are presented in tab. 1. It is important to remember that the training sets were fully synthetic, but the testing set consisted only of natural photos.

Table 1. Detection results (mAP) in different configurations

Testing dataset	NDDS	BP	BP300	Photo
Easy	**0.929**	0.924	0.916	0.926
Moderate	**0.753**	0.724	0.613	0.713
Hard	**0.537**	0.537	0.424	0.483

3.3 Is it worth it?

Synthetic datasets are either on-par or better than manually created ones. Manual datasets, even with a high level of data augmentation, can create only the views of the object that were present on a source photos. Generating data based on 3D models has much more flexibility in both the positioning and lighting conditions. Results achieved using BlenderProc and NDDS are comparable if we use the same number of training pictures. With the current rendering hardware capabilities (like hardware ray-tracing) one can even imagine the *on-the-fly* creation of training data, especially using real-time optimized engines like UE5.

Acknowledgements

M. Stefańczyk is funded by the National Science Centre, Preludium grant no. UMO-2017/25/N/ST6/02358.

References

1. Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., Katam, H.: Blenderproc. arXiv preprint arXiv:1911.01911 (2019)
2. Hodan, T., Michel, F., Brachmann, E., Kehl, W., GlentBuch, A., Kraft, D., Drost, B., Vidal, J., Ihrke, S., Zabulis, X., et al.: Bop: Benchmark for 6d object pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 19–34 (2018)
3. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
4. To, T., Tremblay, J., McKay, D., Yamaguchi, Y., Leung, K., Balanon, A., Cheng, J., Hodge, W., Birchfield, S.: NDDS: NVIDIA deep learning dataset synthesizer (2018), https://github.com/NVIDIA/Dataset_Synthesizer