Article

Generating Images with Physics-Based Rendering for an Industrial Object Detection Task: Realism versus Domain Randomization

Leon Eversberg * and Jens Lambrecht

Chair Industry Grade Networks and Clouds, Technische Universität Berlin, Straße des 17. Juni 135, 10623 Berlin, Germany; lambrecht@tu-berlin.de

* Correspondence: leon.eversberg@tu-berlin.de

Abstract: Limited training data is one of the biggest challenges in the industrial application of deep learning. Generating synthetic training images is a promising solution in computer vision; however, minimizing the domain gap between synthetic and real-world images remains a problem. Therefore, based on a real-world application, we explored the generation of images with physics-based rendering for an industrial object detection task. Setting up the render engine's environment requires a lot of choices and parameters. One fundamental question is whether to apply the concept of domain randomization or use domain knowledge to try and achieve photorealism. To answer this question, we compared different strategies for setting up lighting, background, object texture, additional foreground objects and bounding box computation in a data-centric approach. We compared the resulting average precision from generated images with different levels of realism and variability. In conclusion, we found that domain randomization is a viable strategy for the detection of industrial objects. However, domain knowledge can be used for object-related aspects to improve detection performance. Based on our results, we provide guidelines and an open-source tool for the generation of synthetic images for new industrial applications.

Keywords: data-centric AI; deep learning; domain randomization; image synthesis; object detection; physics-based rendering; synthetic images

Citation: Eversberg, L.; Lambrecht, J. Generating Images with Physics-Based Rendering for an Industrial Object Detection Task: Realism versus Domain Randomization. *Sensors* **2021**, *21*, 7901. <https://doi.org/10.3390/s21237901>

Academic Editors: Biswanath Samanta and Oscar Reinoso Garcia

Received: 22 October 2021

Accepted: 23 November 2021

Published: 26 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Synthetic data are one of the most promising areas of research in modern deep learning as this approach tries to solve the problem of insufficient training data [1]. Compared to collecting and manually labeling real-world images, generating synthetic images is much faster and cheaper. Furthermore, synthetic images can reduce inherent dataset bias in real training sets [2] (e.g., underrepresented viewpoints [3]) by balancing data distribution. Compared to manually labeled datasets, which have been shown to contain numerous errors [4,5], synthetic datasets have no label errors and pixel-perfect label accuracy and consistency (e.g., for bounding boxes or segmentation masks). Lastly, for industrial applications, synthetic images can be generated based on already available 3D CAD models. For those reasons, synthetic images offer a solution to the problem of limited labeled data in industrial deep learning.

Synthetic data for deep learning have been shown to work well on different tasks and domains. Mixing real and synthetic data can improve deep learning model's generalization, compared to using only one source of training data [1,6–8]. However, training with synthetic data for real-world applications suffers from a general machine learning problem called dataset shift, where training and test data come from different distributions [9]. More specifically, in computer vision the source domain consists of synthetic images and the target domain consists of real-world images. In order to overcome this so-called domain gap, different strategies for the generation of synthetic training images have been explored,

such as compositing real images with a cut-and-paste approach [10], rendering images either with very high randomization [11] or in a photorealistic way [12]. Each of these strategies usually involves randomizing multiple parameters in different ways. However, there is still no consensus on the best way to generate training images for object detection tasks. Mayer et al. [13] examined the question of what constitutes good synthetic training data for optical flow and disparity estimation but their findings might not transfer to the high-level computer vision domain of object detection. In an industrial application, we want to detect a texture-less turbine blade at a manual working station in a shopfloor. With this exemplary use case we study the domain gap between synthetic images based on a 3D model and real-world images from a camera for an industrial object detection task.

In an industrial environment the surrounding is not completely random. Lighting, background contents and object colors are typically much less versatile compared to commonly used large-scale datasets, such as PASCAL VOC [14] or COCO [15]. This is especially true if the camera in the deployment environment is stationary. Additionally, industrial objects are often texture-less and can have highly reflective materials (e.g., metallic objects). Thus, we can use this domain knowledge to model the environment accordingly in a rendering software.

Physically based rendering (PBR) attempts to minimize the domain gap between synthetic and real images by rendering photorealistic images. Pharr et al. [16] define PBR as a rendering technique that attempts to simulate reality by modeling the interaction of light and matter according to the principles of physics. The best way to achieve this is by using a path tracing render engine. A path tracer sends out rays of light for each pixel in the image that bounce off surfaces until they hit a light source [6].

In this work, we used PBR to generate synthetic training images for a deep neural network to solve an industrial object detection task. We compared different levels of variability and realism for different aspects in the image generation pipeline.

1. We systematically generated multiple sets of PBR images with different levels of realism, used them to train an object detection model and evaluated the training images' impact on average precision with real-world validation images.
2. Based on our results we provide guidelines for the generation of synthetic training images for industrial object detection tasks.
3. Our source code for generating training images with Blender is open source (<https://github.com/ignc-research/blender-gen>, accessed on 24 November 2021) and can be used for new industrial applications.

The paper is structured as follows: Section 2 outlines work related to the task of generating synthetic training images. Section 3 presents the methodology of this paper, in which the subsections give details on how to setup the rendering engine scene. In Section 4, we present our results where we compared different rendering approaches and evaluated the object detection performance on real-world test data. In Section 5, we discuss those results and their limitations. Finally, Section 6 summarizes our work and we give practical recommendations on how to create synthetic training images for new industrial object detection tasks.

2. Related Work

In the following section we briefly describe some of the different approaches that have been used to generate synthetic training images for deep learning models.

2.1. Cut-and-Paste

Georgakis et al. [17] generated training images by cropping real object instances with a segmentation mask and placing them on support surfaces in background images. Additionally, the scale of the object was adjusted to the depth of the selected position in the image and objects were blended in with the background. In a concurrent approach, Dwibedi et al. [10] also generated training images by cutting segmented object instances of household items, but then randomly pasting them onto background images. In contrast

to Georgakis et al., they suggest that for object detection models local patch-level realism of the bounding box is more important than the global scene layout (i.e., objects do not have to be placed physically correctly on surfaces and the scale can be chosen randomly). A direct comparison of the two approaches on the GMU dataset [18] supports the theory of patch-level realism [10]. Dvornik et al. [19] used [10] as a baseline and compared it to their context model. With their context model, objects were only placed in specific locations in the background image where the visual context of the surroundings are realistic (e.g., planes are placed in the sky and trains on train tracks). Compared to random object placements the context guidance achieved considerably better mean accuracy results on the VOC2012 dataset.

2.2. Domain Randomization

Tobin et al. [20] explored the concept of domain randomization (DR) in order to bridge the gap between a low-fidelity simulation environment and the real world. Their idea behind DR is that the real world can be seen as just another random instance of a simulation environment. High variability in rendered images is achieved by using random camera and object positions, changing the lighting and using non-realistic textures. Tremblay et al. [11] used DR to train an object detection model for cars. They rendered different 3D models of cars with random textures on top of random background images. Furthermore, object poses, camera viewpoints, lighting and flying distractor objects were randomized. Additionally, standard data augmentation techniques, such as flipping and adding Gaussian noise, were utilized.

In [21] the idea of structured domain randomization (SDR) was introduced for a car detection task. SDR uses DR on objects but it keeps the structure of real images by generating realistic global scene layouts. For example, this means that cars are placed on roads and pedestrians on sidewalks. They argue that by using SDR the neural network can learn the relationship between cars and roads.

In [22] Hinterstoisser et al. used OpenGL to render 3D CAD models with Phong lighting [23] on top of real background images. The background images were taken with a camera in the highly cluttered test setting without the objects that are to be detected. Each object was positioned at a random location in the image with a randomly sampled scale and orientation. Furthermore, they used data augmentation techniques such as swapping the background image channels, different light colors, Gaussian noise and blurring with a Gaussian kernel. This approach is similar to the cut-and-paste approaches but it uses rendered objects instead of cutting them out of real images. In a follow-up publication [24] they used a slightly different method where every pixel in the background is composed of randomly sampled 3D models. Furthermore, they propose a deterministic schedule to sample the foreground objects poses in order to balance out the training image distribution. Lastly, they allow background objects to occlude up to 30% of the foreground objects.

2.3. Physics-Based Rendering

Compared to DR, which randomizes the full content of the simulated environment, PBR uses rendering randomization. PBR tries to simulate reality as close as possible while also randomizing environment parameters, such as lighting and the virtual camera's position, to generate diverse training data [25]. Hodaň et al. [12] used the path tracing render engine Arnold [26] to generate highly photorealistic images in order to train a Faster-RCNN object detector. High realism was achieved by rendering 3D models of objects with realistic materials inside highly realistic 3D models of indoor scenes. The complex scenes were either purchased or created with the help of LIDAR scans, photogrammetry and artists using 3D modeling software. Furthermore, they used physics simulation to generate realistic object poses. When compared to the baseline of Hinterstoisser et al. [22], who rendered objects with OpenGL on top of images of the test scene, the realistic scenes achieved an improvement of up to 24% in mean average precision. With high quality render settings (average render time of 720 s per image) they got an improvement of 6% over low

quality settings (15 s) on the LineMod-Occluded [27,28] dataset but no improvement on the Rutgers APC [29] dataset. Thus, they conclude that low-quality settings are sufficient for scenes with simple materials and lighting. Furthermore, they evaluated the importance of context. When objects were placed in a scene that realistically modeled the test data setup, they achieved an improvement of up to 16% compared to a scene that was out-of-context.

Rudorfer et al. [30] also experimented with placing objects inside a 3D modeled scene. The scene was created with Blender and consisted of a very simple setup with objects placed on top of a black box using physics simulation; however, the results showed that it was not possible to train a single shot pose [31] network with this scene setup. In contrast, they achieved much better results by rendering the objects without physics simulation on top of random background images. This suggests that a diverse background is much more important than physics-based object placement and that modeling a 3D scene environment is only beneficial when it is implemented in a highly photorealistic way. Additionally, they found that using background images of the test scene was superior to using random images from the COCO dataset.

In [3], V-Ray for 3ds Max was used by Movshovitz-Attias et al. to render 3D CAD Models of cars in order to train a viewpoint estimation network. To obtain diverse images they randomly sampled the light sources position, energy and color temperature. Furthermore, they used random background images from the PASCAL dataset [14] and a variety of parameters for the virtual camera. In their evaluation they show that a complex modeling of material and random directional lighting outperforms simple material and ambient lighting. They point out that the best cost-to-benefit ratio can be achieved with a high number of synthetic images combined with a small number of real images.

Jabbar et al. [32] used the software Blender to create photorealistic images of transparent drinking glasses. They used high dynamic range images (HDRIs) as 360 degree background images, which also provide complex image-based lighting (IBL) [33], thus removing the need to manually setup realistic lighting in the scene. Their results showed a substantial improvement when the background was similar to the test data, compared to using completely random backgrounds.

Wong et al. [34] published a synthetic end-to-end pipeline for industrial applications. First, texturized 3D object models were created with photogrammetry software. Then, synthetic images were created with Blender by randomly sampling camera position, point light source number and intensity and random background images from the SUN dataset [35]. Their pipeline shows that synthetic images can be used for deep learning even when there is no accurate 3D model available.

2.4. Domain Adaptation

In addition to the aforementioned approaches, domain adaptation techniques can be used to further bridge the domain gap between synthetic and real images. Generative adversarial networks (GANs) [36] can be used to transform generated synthetic images closer to the target domain [37–40]. Alternatively, both source and target domain can be transformed into an intermediate domain, e.g., with the Laplacian Filter [40] or the Pencil Filter [41,42].

2.5. Summary

The cut-and-paste approach requires real-world data with segmentation masks. Furthermore, lighting, object texture and object viewpoint are fixed by the cut-out data, which thereby limits the images that can be generated. The approach of DR is popular in the self-driving cars literature, where it is unfeasible to model every possible outdoor object. DR has the advantage that it can be highly automated as no domain knowledge has to be manually incorporated. If it is not possible to bridge the domain gap with synthetic data, domain adaption techniques can be used to post-process the generated images.

We presume that for industrial use cases, where the environment is known a priori and 3D CAD models are readily available, domain knowledge can be utilized to generate better

training data than using full domain randomization. However, in some cases the level of photorealism is inversely correlated to the amount of image variability (e.g., background images and object textures). The related work section showed different approaches to generate synthetic training images, sometimes with contradicting results such as the theory of local patch-level realism [10] versus more realistic context models [19,21]. Table 1 compares selected PBR and DR approaches that were presented in this section and could be applied to an industrial object detection task. To summarize, there is no consensus on how to best generate synthetic images. Our work tackles this research gap by comparing a spectrum of different strategies that have been proposed in the literature. Our goal is to provide hands-on recommendations on how to generate synthetic training images for a specific object detection task based on a systematic evaluation of different approaches and parameters for PBR in an industrial use case.

Table 1. Comparison of state-of-the-art image generation with PBR and DR.

	Lighting	Background	Object Texture	Occlusion	Object Placement
Hodañ et al. [12]	Standard light sources (e.g., point lights) and Arnold Physical Sky [26]	3D scene models	From 3D model	Multiple 3D objects	Physics simulation
Rudorfer et al. [30]	White point lights	Application domain images or COCO	From 3D model	Multiple 3D objects	Random
Movshovitz-Attias et al. [3]	Directed light w/ different light temperatures	PASCAL	From 3D model	Rectangular patches	Random
Jabbar et al. [32]	IBL from HDRIs	360 ° HDRIs	Glass material	None	On a flat surface
Wong et al. [34]	White point lights	SUN	From 3D model	None	Random

Table 1. Cont.

	Lighting	Background	Object Texture	Occlusion	Object Placement
Tremblay et al. [11]	White point lights and planar light	Flickr 8K [43]	Random (Flickr 8K)	Multiple 3D geometric shapes	On a ground plane
Hinterstoisser et al. [22,24]	Random phong light with random light color	Application domain images or random 3D models	From 3D model	None or multiple 3D objects	Random

3. Method

We used the open-source 3D creation suite Blender to generate images and bounding box labels. Blender uses a path tracer rendering engine to generate physics-based renderings and can be fully automated using Python scripts. For those reasons, Blender is a popular tool amongst researchers for the automated generation of training images for deep learning (e.g., [30,32,34,44,45]). Our workflow and the scope of investigated aspects are depicted in Figure 1, given the constraint that the image generation pipeline can be easily adapted to new objects and industrial use cases. In this work, we compare different strategies on how to model lighting and the background in Blender. Furthermore, we investigate if different object textures or adding occluding foreground objects can improve the detection model's performance. Following the theory of patch-level realism [10], we do not model physically correct object placement in the global scene layout. Details on how we set up the Blender scene for image generation as well as the object detection model are described in the following subsections.

Figure 1. Our methodology for training an object detection model based on a 3D model as input. Different approaches for the green boxes are investigated in this paper.

3.1. 3D Object Model

Industrial objects are often texture-less and their visual appearance is thus defined by their shape, color, reflectance and the environment's lighting [46]. Transferring methods evaluated on existing public datasets to specific industrial scenarios can lead to quite different results [47]. Therefore, we used a texture-less turbine blade and created our own training and evaluation images at a manual working station in a shopfloor environment. We obtained a 3D model of the turbine blade from an industrial 3D scanner (see Figure 2) and imported the file to Blender where it was placed in the scene's origin. We assigned the model a physics-based material according to the bidirectional scattering distribution function (BSDF), which describes the scattering of light on surfaces [16]. We changed values for the material properties *Color*, *Roughness*, *Specular* and *Metallic* in Blender to simulate real-world appearance.

Figure 2. 3D model of a turbine blade, obtained from an industrial 3D scanner.

3.2. Positioning of Camera and 3D Object

The camera position in spherical coordinates, denoted by the radius $r \in \mathbb{R}^+$, inclination $\theta \in [0, \pi]$ and azimuth $\phi \in [0, 2\pi]$, is transformed with (1) into Cartesian coordinates (x_c, y_c, z_c) and then placed in the scene.

$$(x, y, z) = (r \cos \phi \sin \theta, r \sin \phi \sin \theta, r \cos \theta) \quad (1)$$

By sampling uniformly between minimum and maximum values, the camera is randomly positioned on a spherical shell around the origin of the Blender scene. The radius r controls the scale of the object in the rendered image. By limiting ϕ and θ certain viewpoints can be excluded (e.g., the view of the object from below). By uniformly sampling the object's position our 3D model of the turbine blade moves away from the image center. Furthermore, we constrain the camera to always look at an invisible object which is positioned at the scene's origin. We sample three rotation angles $\alpha_{1,2,3} \sim \mathcal{U}(0, 2\pi)$ and perform a XYZ Euler rotation on the invisible object in order to randomly rotate the constrained camera and thus create more diverse training data. The placement of the

constrained camera, 3D model and the invisible object is depicted in Figure 3. If more than 10% of the 3D model's bounding box is outside of the rendered image, the scene setup is automatically resampled.

Figure 3. The camera is constrained to look at an invisible object (X, Y, Z) at the scenes origin. By moving the 3D model and rotating the camera through the empty object we change the pose of our 3D model in the rendered image.

3.3. Modeling of Lighting

We compared two different approaches of modeling lighting: point lights and image-based lighting.

3.3.1. Point Lights

Point lights create omnidirectional light originating from one point. This creates illuminated areas on the 3D model. We create a randomly sampled number of point lights $n_{PL} \sim \mathcal{U}\{n_{min}, n_{max}\}$. Each point light's location is sampled according to (1) with the same parameters as the ones for the camera. Each light has a power of $E_{PL} \sim \mathcal{U}(E_{min}, E_{max})$. In a simple baseline we only use lights with white color. Additionally, in a more complex approach we randomly sample each light's color temperature from a realistic range consisting of six discrete values in addition to white light. As depicted in Figure 4, the color temperatures range from warm 4000 K to cool 9000 K (natural daylight is around 5000 K [48]). Modeling realistic lighting with point lights is not an easy task because the number of lights, their distance to the 3D model and their power are all interrelated hyperparameters.

Figure 4. Each point light color is randomly sampled from six discrete values with color temperatures ranging from warm 4000 K to cool 9000 K in addition to white light. Ref. [49] was used for color conversions.

3.3.2. Image-Based Lighting with HDRIs

HDRIs provide 360 degrees of image-based lighting, thus there is no need to manually model any additional light sources. Compared to point lights or other directed light sources, the easy setup of IBL is a major advantage. Furthermore, IBL also enables objects to have realistic reflections and translucency. We used 123 different indoor HDRIs in 4K resolution (we used all available indoor HDRIs from <https://polyhaven.com/hdris>, accessed on 6 September 2021) as background environment textures. Three examples are shown in Figure 5. We uniformly sample the HDRI light emission strength $E_{IBL} \sim \mathcal{U}(E_{min}, E_{max})$ to create diverse training data.

Figure 5. Compared to point lights, image-based lighting with an HDRI creates a more balanced ambient illumination. The images were rendered with three different HDRIs with $E_{IBL} = 1$.

3.4. Modeling of the Background

We compared three different approaches of generating the image background: random images from a large-scale dataset, 360 degree environment textures and taking pictures of the application domain.

3.4.1. Random Background

For each generated scene we randomly selected a background image from the COCO 2017 train dataset [15], which consists of more than 118,000 images. We rendered only the

3D objects in the scene and then composited this with the random background image. The background image was cropped to fit the rendered image size.

3.4.2. HDRIs

Similar to the random COCO background, in this approach we use the 123 indoor HDRIs in 4K resolution, which were also used for image-based lighting, as background images. HDRI environment maps provide a dynamic 360 degree background (i.e., the background is based on the scene's virtual camera angle). Furthermore, our selected indoor HDRIs have a more realistic indoor environment compared to the full COCO dataset, which also includes outdoor images.

3.4.3. Images of the Application Domain

Lastly, we took pictures from the application domain where our model will be deployed. We collected 43 images with different levels of illumination of the working area in which we want to detect the real turbine blade. In each image we changed the position of some elements, such as a mug or a box with tools. While these images provide a very realistic background compared to COCO images, they also strongly limit the variability of the background. As shown in Figure 6, the different approaches of modeling the background have different image variability (based on number of images and image content diversity) and realism (based on domain knowledge of the application domain).

Figure 6. The three different types of background images which were used. With these three choices we investigate the trade-off between image variability and level of realism.

3.5. Object Texture

Randomizing object textures is a feature that is heavily used in DR. As shown in Figure 7, we compare our simple model with a grey base color to random textures, realistic textures and real textures that we created ourselves. While the real textures have the highest amount of realism, random textures provide more variability in the training data. For random textures, we used the COCO 2017 train images as well as 220 different material textures from <https://polyhaven.com/textures>, accessed on 6 September 2021. For realistic textures, we manually selected 55 different textures from <https://www.textures.com> and <https://polyhaven.com/textures>, accessed on 6 September 2021, that provide a realistic, yet slightly different texture compared to the real object's texture. These include textures from greyish concrete, bare metal, plaster and rock. We also took 20 close-up pictures in slightly different lighting conditions with a smartphone camera from the turbine blade's surface and created our own textures out of them. For every rendered image, we sampled one image texture from the chosen pool of textures.

Figure 7. Examples of different textures that were used. We compare random COCO images, random materials, realistic materials, real material created from photographs of the turbine blade and a single base color against each other.

3.6. Adding Foreground Objects

We added a pool of distracting 3D object models, depicted in Figure 8, consisting of the YCB dataset's [50] tool items (licensed under CC BY 4.0): mug, power drill, wood block, Phillips screwdriver, flat screwdriver, hammer, scissors, large marker, adjustable wrench and medium clamp (We used the 64k laser scans from <http://ycb-benchmarks.s3-website-us-east-1.amazonaws.com>, accessed on 6 September 2021). We used those tools because they fit within our industrial context. Furthermore, we compared the YCB

tools to simple cubes as foreground objects. While cubes are less realistic, they offer perfectly flat surfaces, which enables better texture mapping. For every rendered image we sample $n_{FG} \sim \mathcal{U}\{n_{min}, n_{max}\}$ distracting foreground objects. Then, we randomly moved and rotated those objects within the scene. These distracting objects add the concept of occlusion when they are randomly sampled in front of the turbine blade. Furthermore, by adding additional 3D objects the object detection model cannot rely on artifacts that are introduced by the rendering engine compared to the composited background image.

In addition to the YCB dataset textures, we also randomized the distracting objects' textures. Following the methodology of Section 3.5, we explored random COCO images and random material textures.

Figure 8. Ten realistic YCB tool items [50] are used as additional foreground objects and compared to a simple cube.

3.7. Computation of Bounding Box Labels

After setting up the scene in Blender, we transformed all vertices of the turbine blade's 3D model from world space (x_i, y_i, z_i) to rendered image space (u_i, v_i) according to (2), which uses the virtual camera's projection matrix $\mathbf{P} \in \mathbb{R}^{3 \times 4}$ and a scaling factor s .

$$s \cdot \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \mathbf{P} \cdot \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \quad (2)$$

After transforming all vertices, we use the obtained minimum and maximum values $\{u_{min}, u_{max}, v_{min}, v_{max}\}$ to generate tight bounding boxes (x, y, w, h) around the rendered turbine blade according to (3) and (4) in the COCO data format.

$$(x, y) = (u_{min}, v_{min}) \quad (3)$$

$$(w, h) = (u_{max} - u_{min}, v_{max} - v_{min}) \quad (4)$$

Figure 9 shows the difference in the resulting 2D bounding box label when using (2)–(4) on all vertices of the 3D model versus using only the 3D bounding box coordinates provided by Blender. Though using all vertices is more computationally expensive, the resulting labels are much tighter than the ones from the 3D bounding box.

Figure 9. Comparison of bounding box computations. Using the eight 3D bounding box coordinates results in the blue label and using all mesh vertices results in the green label.

3.8. Object Detection Model and Training

For our object detection task we used a Faster R-CNN network [51], as shown in Figure 10. We used a pre-trained ResNet-50 backbone [52] to obtain convolutional feature maps. These feature maps are used by a region proposal network (RPN), which outputs regions of interest (RoI) as rectangular bounding boxes and classifies them as either being an object or not. Given the convolutional feature maps from the CNN backbone and the RoIs from the RPN, a feature vector is computed for each RoI. Then, the model outputs discrete probabilities for each object class as well as bounding boxes [53]. Each bounding box prediction (x, y, w, h) is defined by its upper-left corner position (x, y) in the image, its width w and its height h . While bigger or newer models might increase the final detection result, we chose to keep the model fixed following a data-centric AI approach and only change the generated input images.

Figure 10. Object detection model based on Faster R-CNN.

As evaluation metric for our object detection task we use average precision (AP), which approximates the area under the precision/recall curve. Precision and recall are defined by true positives tp , false positives fp and false negatives fn according to (5) and (6). A detection is considered a true positive if the Intersection over Union (IoU) of a predicted bounding box with the ground truth bounding box exceeds a given threshold [54]. With the PASCAL VOC metric this threshold is 50% ($AP^{0.5}$). For the COCO metric an average over 10 IoU thresholds is computed, ranging from 50% to 95% with a step size of 5% ($AP^{[0.5:0.95]}$).

We trained every model for a maximum of 25 epochs. Then, the model with the highest $AP^{[0.5:0.95]}$ was selected. More details on our choices for deep learning hyperparameters can be found in Table 2.

Table 2. Deep learning hyperparameters.

Hyperparameter	Value
Optimizer [55]	Stochastic gradient descent (SGD) with learning rate $\epsilon = 0.00001$, momentum $\mu = 0.9$, L^2 weight decay $\alpha = 0.0001$
Epochs	25
Training examples	5000
Batch size	8
Image size	640 pixel $\times$ 360 pixel

$$precision = \frac{tp}{tp + fp} \quad (5)$$

$$recall = \frac{tp}{tp + fn} \quad (6)$$

3.9. Validation Data

We recorded 650 validation images with a Microsoft Azure Kinect camera in 1080P resolution from the manual working station. In every image the turbine blade is visible and the pose of the turbine blade is different for all images. We allowed small forms of occlusion from a hand, fingers or from a vise. The validation images are as close as possible to the real industrial working conditions. We manually labeled the bounding box of the turbine blade for all images, which took about 6 s per image. By comparing validation error on real images for different image generation strategies, we can measure which strategy is best suited to close the domain gap between synthetic images and real-world images.

4. Experiments and Results

If not otherwise specified, the following results were created by rendering 5000 images and then training the object detection model for 25 epochs (always using the same random seed for image generation and model training). We used point lights for lighting, random COCO background images, a grey base color for the turbine blade's model and no additional foreground objects as an initial baseline. Starting with this baseline, we compare the different image generation approaches described in Sections 3.3–3.7.

On our computer with two Tesla M60 GPUs and an Intel Broadwell CPU, rendering took between 1.7 s and 5.6 s per image on average, depending on the scene configuration. Training Faster R-CNN with 5000 synthetic training images for 25 epochs took around 8 h on a single GPU.

4.1. Computation of the Bounding Box

First, we compare two different strategies for computing the bounding box. As shown in Figure 11, computing tight bounding boxes by transforming all mesh vertices to image space leads to a much better performance in $AP^{[0.5:0.95]}$ than using 3D bounding box corner coordinates. When transforming all mesh vertices with Equations (2)–(4), rendering an image and computing the label took 3.3 s. On contrast, when transforming only the 3D bounding box coordinates to image space the process took only 1.7 s on average. However, bounding box computation time can be reduced by downsampling the number of vertices of the 3D mesh.

Figure 11. Comparison of bounding box label computation.

4.2. Lighting

The results for different lighting models are shown in Table 3. Adding color to point lights by randomizing color temperature improved the performance slightly compared to white point lights. Image-based lighting with 123 indoor HDRIs achieved the best performance while at the same time requiring less parameter choices.

Table 3. Lighting results.

Lighting Model	n _{PL}	E	AP ^[0.5:0.95]	AP ^{0.5}
PL (white)	$\mathcal{U}\{1, 4\}$	$\mathcal{U}(20\text{ W}, 100\text{ W})$	0.623	0.917
PL (white)	$\mathcal{U}\{1, 5\}$	$\mathcal{U}(20\text{ W}, 100\text{ W})$	0.633	0.908
PL (white)	$\mathcal{U}\{1, 6\}$	$\mathcal{U}(20\text{ W}, 100\text{ W})$	0.632	0.919
PL (white)	$\mathcal{U}\{1, 6\}$	$\mathcal{U}(20\text{ W}, 70\text{ W})$	0.633	0.914
PL (white)	$\mathcal{U}\{1, 6\}$	$\mathcal{U}(20\text{ W}, 130\text{ W})$	0.631	0.928
PL (temperature)	$\mathcal{U}\{1, 6\}$	$\mathcal{U}(20\text{ W}, 130\text{ W})$	0.635	0.929
IBL with HDRIs	-	$\mathcal{U}(0.5, 6)$	0.640	0.926
IBL with HDRIs	-	$\mathcal{U}(0.5, 7)$	0.642	0.926
IBL with HDRIs	-	$\mathcal{U}(0.5, 8)$	0.641	0.931
IBL with HDRIs	-	$\mathcal{U}(1, 8)$	0.642	0.931

4.3. Background

Table 4 shows that 360 degree indoor HDRIs are not a good choice for background images. We believe this is due to the position and orientation of the virtual camera, which is often looking towards the indoor ceiling or floor of the scene. For this reason, HDRIs often do not provide rich background images.

Furthermore, the high variability in the large-scale COCO dataset outperformed the high realism of domain-specific background images of the manual working station. Contrary to our initial belief, mixing COCO and deployment background images did not result in an improvement over using only COCO images.

4.4. Object Texture

Results for changing the turbine blade's texture are shown in Table 5. Selecting textures with a realistic color palette achieved the best performance. Realistic material textures provide realism as well as more variability than the real material texture.

Table 4. Background results.

Background Model	AP ^[0.5:0.95]	AP ^{0.5}
COCO images	0.642	0.931
HDRI images	0.589	0.899
Deployment domain images	0.612	0.938
50% COCO and 50% deployment images	0.635	0.935
75% COCO and 25% deployment images	0.634	0.925
90% COCO and 10% deployment images	0.641	0.931

Random material textures performed only slightly worse; therefore, domain randomization of the object texture seems to be a viable alternative if no appropriate material textures are available. Projecting random COCO images onto the turbine blade's UV map resulted in unnatural and irregular textures and thus performed the worst.

Table 5. Object texture results.

Texture Model	AP ^[0.5:0.95]	AP ^{0.5}
Grey base color	0.642	0.931
Random COCO images	0.623	0.946
Random material texture	0.644	0.962
Realistic material texture	0.653	0.963
Real material texture	0.648	0.948

4.5. Foreground Objects

Table 6 shows the results of adding additional foreground objects. For the YCB tools, rendering up to three objects improves the detection performance slightly. Randomizing the YCB objects' textures performed worse than using the original textures. We believe this is due to the fact that the YCB tools already have complex textures, thus there is no benefit in randomizing them.

The cubes offer perfectly flat surface areas, which are ideal for mapping textures onto them. As a result, using cubes as simple geometric shapes with random textures resulted in a slightly better AP than the YCB tools and a significant increase compared to no foreground objects.

Table 6. Results for additional foreground objects.

Foreground Objects	n _{FG}	Texture	AP ^[0.5:0.95]	AP ^{0.5}
None	0	-	0.653	0.963
YCB tools	$\mathcal{U}\{0,1\}$	Original	0.657	0.963
YCB tools	$\mathcal{U}\{0,2\}$	Original	0.653	0.963
YCB tools	$\mathcal{U}\{0,3\}$	Original	0.660	0.972
YCB tools	$\mathcal{U}\{0,4\}$	Original	0.659	0.972
YCB tools	$\mathcal{U}\{0,3\}$	COCO	0.653	0.951
YCB tools	$\mathcal{U}\{0,3\}$	Random material	0.647	0.958
Cubes	$\mathcal{U}\{0,3\}$	COCO	0.666	0.987
Cubes	$\mathcal{U}\{0,3\}$	Random material	0.669	0.989

4.6. Number of Rendered Images

After investigating the image generation methodology in Blender, we investigated the number of rendered training images. Previous results used only 5000 rendered images and 25 epochs for training to reduce computation time while searching for optimal hyperparameters. Other than using more or less training data, we also changed the number of training epochs. All of the following models were trained for up to 24 h. Figure 12 shows the relationship between the number of rendered images for the training set and the average precision of the object detection model on validation images measured in $AP^{[0.5:0.95]}$. While the generation of synthetic data has the capability to create an unlimited amount of training data, the chart shows that a maximum average precision of $AP^{[0.5:0.95]} = 0.7$ is reached already with $n_{TI} = 5000$ training images. Adding more training data does not improve model performance after this point.

Figure 12. The impact of adding more synthetic training images on the object detection model for a training time of up to 24 h.

Qualitative object detection results on validation data are depicted in Figure 1.

4.7. Using Real Images

In order to compare our PBR-based approach to real images, we also trained the Faster-RCNN object detection model with a small number of real images. Thus, we captured and labeled $n_{TI} = 200$ images from the application domain in the same way as the validation data. Because it has been shown that training on synthetic data and then fine-tuning with real data in a two-step approach can achieve better performance than simply mixing synthetic and training datasets [7,8], we also used our PBR-model trained on $n_{TI} = 5000$ images as a pre-trained baseline and fine-tuned this with the same 200 real images.

As shown in Table 7, the models trained only on synthetic PBR images or real images achieve the same performance. Furthermore, the fine-tuned model has a substantially higher average precision than the other two models. The model pre-trained on PBR images acts as a strong base for further fine-tuning on real images.

Table 7. Using real training data.

Model	n _{TI}	AP ^[0.5:0.95]	AP ^{0.5}
Real training images	200	0.709	0.985
PBR training images	5000	0.704	0.989
Pre-trained on PBR and fine-tuned on real images	5000 and 200	0.785	1.00

4.8. Transfer to New Objects

Finally, after thorough investigation of the image generation methodology and hyperparameters, we created three new test datasets with 200 test images each according to Section 3.9 with occlusion and clutter. In addition to our previous turbine blade (TB 1) we added two new objects (TB 2 and TB 3). The two new turbine blades differ significantly in color and geometry from our previously used model, see Figure 13. For all three objects we performed a sequential ablation study on the new test data. Table 8 shows that our image generation methodology can be transferred to new objects. Applying previous results on new objects resulted in an increase in $AP^{[0.5:0.95]}$.

Figure 13. In addition to the previously studied TB 1, TB 2 and TB 3 are added as new test objects.**Table 8.** Sequential ablation study on test data with new objects ($AP^{[0.5:0.95]}$).

Object	Baseline ¹	IBL with HDRIs	Realistic Material Texture	Foreground Objects
TB 1	0.620	0.662	0.663	0.677
TB 2	0.481	0.568	0.580	0.629
TB 3	0.466	0.467	0.501	0.556
mean	0.522	0.566	0.581	0.621

¹ The baseline was trained on 5000 images with tight bounding box computation, white point lights, COCO background images, a single basecolor and no additional 3D objects.

4.9. Qualitative Results

Qualitative results of our final object detection models for TB 1, TB 2 and TB 3, trained only on PBR-images, are shown in Figure 2. Our deep learning model usually detects the turbine blade with very high confidence and outputs a tight bounding box. Rarely observed errors are mostly images with high occlusion and false positive detections. Examples of the rendered training images are shown in Figure 3.

5. Discussion

The results from Sections 4.2–4.5 can be arranged into two groups: object-related and non-object-related aspects. Background images and additional distracting foreground objects are unrelated to the object of interest. For both of these aspects the concept of domain randomization outperformed higher realism. Our results show that there is no need to use realistic image backgrounds or realistic distractor objects. On the other hand, lighting and object textures affect the visual appearance of the 3D model. For these aspects we found that realistic indoor lighting and realistic material textures performed the best. Although, random material textures resulted in the same $AP^{[0.5:0.95]}$ and a higher $AP^{0.5}$ than real material textures. This suggests that high variability is still an important aspect when trying to achieve high photorealism that should not be neglected.

However, our results on object texture and lighting are limited by the appearance of the turbine blade (TB 1) and the manual working station. As can be seen in the validation images from Figure 1, the turbine blade has a mostly homogenous grey color that is similar to other elements in the validation images and there was mostly artificial white light from above the table.

Even though we transferred our method to new objects (TB 2 and TB 3) and new test data in Section 4.8, our results are still limited by our specific use case. However, we provide the methodology and open-source tool to easily generate labeled PBR images for new objects and different industrial environments.

6. Conclusions

In this work we presented an image generation pipeline based on PBR that can generate synthetic training images for deep learning object detection models. With purely synthetic images as input data and thereby no manual labeling, we trained a Faster R-CNN model for texture-less turbine blades. We showed that the biggest improvements in average precision come from a tight bounding box label computation and optional fine-tuning on a small amount of real-world data. Furthermore, we evaluated different approaches regarding lighting, background images, object texture and additional foreground objects. Additionally, we transferred our methodology to new test data with two additional turbine blades and confirmed the positive effects of our image generation pipeline. Based on our results we propose the following guidelines for the generation of PBR training images for industrial objects.

First, we recommend image-based lighting by using HDRIs as environment textures. In addition to a slightly better average precision than point lights, IBL is much easier to setup with the only hyperparameter being the light's emission strength. Second, we recommend using background images from a large-scale dataset, such as COCO. We showed that random background images perform better than a small amount of realistic images from the application domain. Third, we recommend randomizing the 3D object's texture while at the same time keeping object appearance realistic. Lastly, we recommend using simple cubes with random material textures as additional distracting foreground objects. Based on our results, there is no need to use application-specific 3D foreground objects. Finally, we recommend rendering at least 5000 images per class as a starting point.

Our best image generation pipeline requires only the manual selection of realistic object textures. Background images, lighting and foreground objects are randomized from a pool of files that only need to be downloaded once. However, random object textures performed not much worse than realistic object textures and are therefore a viable alternative if realistic object textures are unavailable. With full domain randomization, industrial object detection models can be trained automatically based only on a 3D model and without the need of any domain knowledge.

For future research, we encourage others to try our open-source image generation pipeline in Blender (<https://github.com/ignc-research/blender-gen>, accessed on 22 November 2021) for new objects in different industrial environments as well as add further extensions to the image generation methodology. Additionally, alternative object

detection models (e.g., YOLO or transformer-based models) could be used and compared to Faster R-CNN. Furthermore, domain adaptation techniques could be applied to further decrease the domain gap. While our work focused on the task of object detection, we believe that the methodology can be transferred to similar high-level computer vision tasks, such as object pose estimation or object segmentation.

Author Contributions: Conceptualization, J.L.; methodology, L.E.; software, L.E.; data curation, L.E.; writing—original draft preparation, L.E.; writing—review and editing, J.L.; visualization, L.E.; supervision, J.L.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is part of the project MRO 2.0—Maintenance, Repair and Overhaul and was supported in part by the European Regional Development Fund (ERDF) under grant number ProFIT-10167454.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The turbine blade data are not publicly available due to protection of intellectual property.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Qualitative results of our object detection model for TB 1 on validation images are shown in Figure 1. Results of our object detection models for TB 1, TB 2 and TB 3 on new test images are shown in Figure 2. Figure 3 shows examples of rendered training images using our image generation methodology.

Figure 1. Object detection results on real validation data from our Faster R-CNN model trained purely on 5000 synthetic PBR images. Predictions are blue (including detection confidence) and manually created ground truth labels are green. The last row shows some errors.

Figure 2. Object detection results on real test data from our three Faster R-CNN models trained purely on 5000 synthetic PBR images each. Predictions are blue (including detection confidence) and manually created ground truth labels are green. The last column shows some errors.

Figure 3. Examples of our synthetic training images generated with PBR. For the three turbine blades we used COCO background images, image-based lighting from HDRIs, random realistic material textures and up to three cubes with random material textures as additional foreground objects.

References

- Nikolenko, S.I. Synthetic Data for Deep Learning. *arXiv* **2019**, arXiv:1909.11512v1.
- Torrallba, A.; Efros, A.A. Unbiased look at dataset bias. In Proceedings of the CVPR, Colorado Springs, CO, USA, 20–25 June 2011. [\[CrossRef\]](#)
- Movshovitz-Attias, Y.; Kanade, T.; Sheikh, Y. How Useful Is Photo-Realistic Rendering for Visual Learning? In *Lecture Notes in Computer Science*; Springer International Publishing: Cham, Switzerland, 2016; pp. 202–217. [\[CrossRef\]](#)
- Northcutt, C.G.; Jiang, L.; Chuang, I.L. Confident Learning: Estimating Uncertainty in Dataset Labels. *arXiv* **2021**, arXiv:1911.00068v4.
- Northcutt, C.G.; Athalye, A.; Mueller, J. Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. *arXiv* **2021**, arXiv:2103.14749v2.
- Schraml, D. Physically based synthetic image generation for machine learning: A review of pertinent literature. In *Photonics and Education in Measurement Science*; International Society for Optics and Photonics: Bellingham, WA, USA, 2019; Volume 11144. [\[CrossRef\]](#)
- Lambrech, J.; Kästner, L. Towards the Usage of Synthetic Data for Marker-Less Pose Estimation of Articulated Robots in RGB Images. In Proceedings of the 2019 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, 2–6 December 2019. [\[CrossRef\]](#)
- Nowruzi, F.E.; Kapoor, P.; Kolhatkar, D.; Hassanat, F.A.; Laganier, R.; Rebut, J. How much real data do we actually need: Analyzing object detection performance using synthetic and real data. *arXiv* **2019**, arXiv:1907.07061v1.
- Candela, J. *Dataset Shift in Machine Learning*; MIT Press: Cambridge, MA, USA; London, UK, 2009.
- Dwibedi, D.; Misra, I.; Hebert, M. Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017. [\[CrossRef\]](#)
- Tremblay, J.; Prakash, A.; Acuna, D.; Brophy, M.; Jampani, V.; Anil, C.; To, T.; Cameracci, E.; Boochoon, S.; Birchfield, S. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, USA, 18–22 June 2018. [\[CrossRef\]](#)
- Hodan, T.; Vineet, V.; Gal, R.; Shalev, E.; Hanzelka, J.; Connell, T.; Urbina, P.; Sinha, S.N.; Guenter, B. Photorealistic Image Synthesis for Object Instance Detection. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019. [\[CrossRef\]](#)
- Mayer, N.; Ilg, E.; Fischer, P.; Hazirbas, C.; Cremers, D.; Dosovitskiy, A.; Brox, T. What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation? *Int. J. Comput. Vis.* **2018**, *126*, 942–960. [\[CrossRef\]](#)
- Everingham, M.; Eslami, S.M.A.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes Challenge: A Retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [\[CrossRef\]](#)
- Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*; Springer International Publishing: Cham, Switzerland, 2014; pp. 740–755. [\[CrossRef\]](#)
- Pharr, M.; Jakob, W.; Humphreys, G. *Physically Based Rendering: From Theory to Implementation*, 3rd ed.; Morgan Kaufmann: Burlington, MA, USA, 2016.
- Georgakis, G.; Mousavian, A.; Berg, A.; Kosecka, J. Synthesizing Training Data for Object Detection in Indoor Scenes. In Proceedings of the Robotics: Science and Systems XIII, Robotics: Science and Systems Foundation, Cambridge, MA, USA, 12–16 July 2017. [\[CrossRef\]](#)
- Georgakis, G.; Reza, M.A.; Mousavian, A.; Le, P.H.; Kosecka, J. Multiview RGB-D Dataset for Object Instance Detection. In Proceedings of the IEEE 2016 Fourth International Conference on 3D Vision (3DV), Stanford, CA, USA, 25–28 October 2016. [\[CrossRef\]](#)
- Dvornik, N.; Mairal, J.; Schmid, C. Modeling Visual Context Is Key to Augmenting Object Detection Datasets. In *Computer Vision—ECCV 2018*; Springer International Publishing: Cham, Switzerland, 2018; pp. 375–391. [\[CrossRef\]](#)
- Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017. [\[CrossRef\]](#)
- Prakash, A.; Boochoon, S.; Brophy, M.; Acuna, D.; Cameracci, E.; State, G.; Shapira, O.; Birchfield, S. Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data. In Proceedings of the IEEE 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019. [\[CrossRef\]](#)
- Hinterstoisser, S.; Lepetit, V.; Wohlhart, P.; Konolige, K. On Pre-Trained Image Features and Synthetic Images for Deep Learning. In *Computer Vision—ECCV 2018 Workshops*; Springer International Publishing: Cham, Switzerland, 2017; pp. 682–697. [\[CrossRef\]](#)
- Phong, B.T. Illumination for Computer Generated Pictures. *Commun. ACM* **1975**, *18*, 311–317. [\[CrossRef\]](#)
- Hinterstoisser, S.; Pauly, O.; Heibel, H.; Marek, M.; Bokeloh, M. An Annotation Saved is an Annotation Earned: Using Fully Synthetic Training for Object Instance Detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea, 27–28 October 2019; pp. 2787–2796. [\[CrossRef\]](#)
- Tsirikoglou, A.; Eilertsen, G.; Unger, J. A Survey of Image Synthesis Methods for Visual Machine Learning. *Comput. Graph. Forum* **2020**, *39*, 426–451. [\[CrossRef\]](#)

26. Georgiev, I.; Ize, T.; Farnsworth, M.; Montoya-Vozmediano, R.; King, A.; Lommel, B.V.; Jimenez, A.; Anson, O.; Ogaki, S.; Johnston, E.; et al. Arnold: A Brute-Force Production Path Tracer. *ACM Trans. Graph.* **2018**, *37*, 1–12. [\[CrossRef\]](#)
27. Hinterstoisser, S.; Lepetit, V.; Ilic, S.; Holzer, S.; Bradski, G.; Konolige, K.; Navab, N. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Computer Vision – ACCV 2012*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 548–562. [\[CrossRef\]](#)
28. Brachmann, E.; Krull, A.; Michel, F.; Gumhold, S.; Shotton, J.; Rother, C. Learning 6D Object Pose Estimation Using 3D Object Coordinates. In *Computer Vision—ECCV 2014*; Springer International Publishing: Cham, Switzerland, 2014; pp. 536–551. [\[CrossRef\]](#)
29. Rennie, C.; Shome, R.; Bekris, K.E.; Souza, A.F.D. A Dataset for Improved RGBD-Based Object Detection and Pose Estimation for Warehouse Pick-and-Place. *IEEE Robot. Autom. Lett.* **2016**, *1*, 1179–1185. [\[CrossRef\]](#)
30. Rudorfer, M.; Neumann, L.; Kruger, J. Towards Learning 3d Object Detection and 6d Pose Estimation from Synthetic Data. In Proceedings of the 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 10–13 September 2019. [\[CrossRef\]](#)
31. Tekin, B.; Sinha, S.N.; Fua, P. Real-Time Seamless Single Shot 6D Object Pose Prediction. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018. [\[CrossRef\]](#)
32. Jabbar, A.; Farrawell, L.; Fountain, J.; Chalup, S.K. Training Deep Neural Networks for Detecting Drinking Glasses Using Synthetic Images. In *Neural Information Processing*; Springer International Publishing: Cham, Switzerland, 2017; pp. 354–363. [\[CrossRef\]](#)
33. Reinhard, E.; Heidrich, W.; Debevec, P.; Pattanaik, S.; Ward, G.; Myszkowski, K. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*; Morgan Kaufmann: Burlington, MA, USA, 2010.
34. Wong, M.Z.; Kunii, K.; Baylis, M.; Ong, W.H.; Kroupa, P.; Koller, S. Synthetic dataset generation for object-to-model deep learning in industrial applications. *PeerJ Comput. Sci.* **2019**, *5*, e222. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Xiao, J.; Hays, J.; Ehinger, K.A.; Oliva, A.; Torralba, A. SUN database: Large-scale scene recognition from abbey to zoo. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010. [\[CrossRef\]](#)
36. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2014; Volume 27.
37. Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; Webb, R. Learning From Simulated and Unsupervised Images Through Adversarial Training. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
38. Peng, X.; Saenko, K. Synthetic to Real Adaptation with Generative Correlation Alignment Networks. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018. [\[CrossRef\]](#)
39. Sankaranarayanan, S.; Balaji, Y.; Jain, A.; Lim, S.N.; Chellappa, R. Learning From Synthetic Data: Addressing Domain Shift for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
40. Rojtblerg, P.; Pollabauer, T.; Kuijper, A. Style-transfer GANs for bridging the domain gap in synthetic pose estimator training. In Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), Utrecht, The Netherlands, 14–18 December 2020. [\[CrossRef\]](#)
41. Su, Y.; Rambach, J.; Pagani, A.; Stricker, D. SynPo-Net—Accurate and Fast CNN-Based 6DoF Object Pose Estimation Using Synthetic Training. *Sensors* **2021**, *21*, 300. [\[CrossRef\]](#) [\[PubMed\]](#)
42. Rambach, J.; Deng, C.; Pagani, A.; Stricker, D. Learning 6DoF Object Poses from Synthetic Single Channel Images. In Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Munich, Germany, 16–20 October 2018. [\[CrossRef\]](#)
43. Hodosh, M.; Young, P.; Hockenmaier, J. Framing image description as a ranking task: Data, models and evaluation metrics. *J. Artif. Intell. Res.* **2013**, *47*, 853–899. [\[CrossRef\]](#)
44. Andulkar, M.; Hodapp, J.; Reichling, T.; Reichenbach, M.; Berger, U. Training CNNs from Synthetic Data for Part Handling in Industrial Environments. In Proceedings of the 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), Munich, Germany, 20–24 August 2018. [\[CrossRef\]](#)
45. Denninger, M.; Sundermeyer, M.; Winkelbauer, D.; Olefir, D.; Hodan, T.; Zidan, Y.; Elbadrawy, M.; Knauer, M.; Katam, H.; Lodhi, A. BlenderProc: Reducing the Reality Gap with Photorealistic Rendering. In Proceedings of the Robotics: Science and Systems (RSS), Virtual Event/Corvallis, OR, USA, 12–16 July 2020.
46. Hodan, T.; Haluza, P.; Obdrzalek, S.; Matas, J.; Lourakis, M.; Zabulis, X. T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017; pp. 880–888. [\[CrossRef\]](#)
47. Drost, B.; Ulrich, M.; Bergmann, P.; Härtinger, P.; Steger, C. Introducing MVTEC ITODD—A Dataset for 3D Object Recognition in Industry. In Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 22–29 October 2017; pp. 2200–2208. [\[CrossRef\]](#)
48. ISO 3664:2009. In *Graphic Technology and Photography—Viewing Conditions*; International Organization for Standardization: Geneva, Switzerland, 2009.

-
49. Charity, M. What Color Is a Blackbody?—Some Pixel Rgb Values. Available online: <http://www.vendian.org/mncharity/dir3/blackbody/> (accessed on 9 April 2019).
 50. Calli, B.; Singh, A.; Walsman, A.; Srinivasa, S.; Abbeel, P.; Dollar, A.M. The YCB object and Model set: Towards common benchmarks for manipulation research. In Proceedings of the 2015 International Conference on Advanced Robotics (ICAR), Istanbul, Turkey, 27–31 July 2015. [CrossRef]
 51. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2015; Volume 28, pp. 91–99.
 52. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
 53. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.
 54. Everingham, M.; Gool, L.V.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2009**, *88*, 303–338. [CrossRef]
 55. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA; London, UK, 2016.