

PHOTOREALISTIC IMAGE SYNTHESIS FOR OBJECT INSTANCE DETECTION

Tomáš Hodaň^{1,2} Vibhav Vineet² Ran Gal² Emanuel Shalev² Jon Hanzelka²
Treb Connell² Pedro Urbina² Sudipta N. Sinha² Brian Guenter²

¹FEE, Czech Technical University in Prague ²Microsoft Research

ABSTRACT

We present an approach to synthesize highly photorealistic images of 3D object models, which we use to train a convolutional neural network for detecting the objects in real images. The proposed approach has three key ingredients: (1) 3D object models are rendered in 3D models of complete scenes with realistic materials and lighting, (2) plausible geometric configuration of objects and cameras in a scene is generated using physics simulation, and (3) high photorealism of the synthesized images is achieved by physically based rendering. When trained on images synthesized by the proposed approach, the Faster R-CNN object detector [1] achieves a 24% absolute improvement of mAP@.75IoU on Rutgers APC [2] and 11% on LineMod-Occluded [3] datasets, compared to a baseline where the training images are synthesized by rendering object models on top of random photographs. This work is a step towards being able to effectively train object detectors without capturing or annotating any real images. A dataset of 600K synthetic images with ground truth annotations for various computer vision tasks will be released on the project website: thodan.github.io/objectsynth.

1. INTRODUCTION

Object instance detection is a computer vision task which involves recognizing specific objects in an image and estimating their 2D bounding boxes. Convolutional neural networks (CNN's) have become the standard approach for tackling this task. However, training CNN models requires large amounts of real annotated images which are expensive to acquire.

Computer graphics has been used to synthesize training images for various computer vision tasks. This approach scales well as only minimal human effort, which may include 3D modeling, is required. Nevertheless, despite training CNN's on massive datasets of diverse synthetic images, a large drop of performance has been observed when models trained only on synthetic images were tested on real images [4, 5, 6]. The domain gap between the synthetic and real images can be reduced by domain adaptation techniques that aim to learn domain invariant representations or to transfer trained models from one domain to another [7]. A different line of work, presumably complementary to domain adaptation, has recently tried to reduce the domain gap by synthe-



Photorealistic images synthesized by the proposed approach



Real images from LineMod [14] and Rutgers APC [2] datasets

Fig. 1. Faster R-CNN object detector [1] achieves 11–24% higher mAP@.75IoU on real test images when trained on photorealistic synthetic images than when trained on images of objects rendered on top of random photographs.

sizing training images with a higher degree of visual realism. The use of physically based rendering has been considered with this motivation and shown promising results [8, 9].

Physically based rendering techniques, e.g. Arnold [10], accurately simulate the flow of light energy in the scene by ray tracing. This naturally accounts for complex illumination effects such as scattering, refraction and reflection, including diffuse and specular interreflection between the objects and the scene and between the objects themselves. The rendered images are very realistic and often difficult to differentiate from real photographs [11]. Rendering techniques based on rasterization, e.g. OpenGL [12], can approximate the complex effects in an ad hoc way through custom shaders, but the approximations cause physically incorrect artifacts that are difficult to eliminate [13]. Physically based rendering has been historically noticeably slower than rasterization, however, the recently introduced Nvidia RTX ray tracing GPU promises a substantial reduction of the rendering time.

In this work, we investigate the use of highly photorealistic synthetic images for training Faster R-CNN, a CNN-based object detector [1]. To synthesize the images, we present an approach with three key ingredients. First, 3D models of objects are not rendered in isolation but inside 3D models of complete scenes. For this purpose, we have created models of six indoor scenes with realistic materials and lighting. Second, plausible geometric configuration of objects and cameras in a scene is generated using physics simulation. Finally, a high degree of visual realism is achieved by physically based rendering (see Fig. 1 and the supplementary material).

The experiments show that Faster R-CNN trained on the photorealistic synthetic images achieves a 24% absolute improvement of mAP@.75IoU on real test images from Rutgers APC [2], and 11% on real test images from LineMod-Occluded [3, 14]. The improvement is relative to a baseline where the training images are synthesized by rendering object models on top of random photographs – similar images are commonly used for training methods for tasks such as object instance detection [15], object instance segmentation [16], and 6D object pose estimation [17, 18].

2. RELATED WORK

Synthetic images have been used for benchmarking and training models for various computer vision tasks. Here we review approaches to generate synthetic images and approaches to reduce the gap between the synthetic and real domains.

Rendering Objects. Su et al. [5] synthesized images of 3D object models for viewpoint estimation, Hinterstoisser et al. [16] for object instance detection and segmentation, and Dosovitskiy et al. [19] for optical flow estimation. They used a fixed OpenGL pipeline and pasted the rendered pixels over randomly selected real photographs. Rad et al. [18], Tekin et al. [20] and Dwibedi et al. [15] similarly pasted segments of objects from real images on other real images for object detection and pose estimation. Dvornik et al. [21] showed the importance of selecting suitable background images. While these approaches are easy to implement, the resulting images are not realistic, objects often have inconsistent shading with respect to the background scene, interreflections and shadows are missing and the object pose and context are usually not natural. Attias et al. [22] rendered photorealistic images of 3D car models placed within 3D scene models and showed the benefit over naive rendering methods, whereas Tremblay et al. [23] rendered objects in physically plausible poses in diverse scenes, but did not use physically based rendering.

Rendering Scenes. Another line of work explored rendering of complete scenes and generating corresponding ground truth maps. Richter et al. [24, 4] leveraged existing commercial game engines to acquire ground truth for several tasks. However, such game engines cannot be customized to insert new 3D object models. Synthia [25] and Virtual KITTI [26]

datasets were generated using virtual cities modeled from scratch. Handa et al. [27] and Zhang [9] modeled 3D scenes for semantic scene understanding. Finally, SceneNet [28] was created by synthesizing RGB-D video frames from simulated cameras moving realistically within static scenes.

Domain Adaptation. Popular approaches to bridge the gap between the synthetic and real domains include re-training the model in the real domain, learning domain invariant features, or learning a mapping between the two domains [29, 7]. In contrast, domain randomization methods reduce the gap by randomizing the rendering parameters and have been used in object localization and pose estimation [29, 30, 31].

PBR based approaches. Physically based rendering (PBR) techniques can synthesize images with a high degree of visual realism which promises to reduce the domain gap. Li and Snavely [8] used PBR images to train models for intrinsic image decomposition and Zhang et al. [9] for semantic segmentation, normal estimation and boundary detection. However, they focus on scene understanding not object understanding tasks. Wood et al. [32] used PBR images of eyes for training gaze estimation models. Other ways to generate photorealistic images have been also proposed [33, 34].

3. PROPOSED APPROACH

To achieve a high degree of visual realism in computer generated imagery, one needs to focus on (1) modeling the scene to a high level of detail in terms of geometry, textures and materials, and (2) simulating the lighting, including soft shadows, reflections, refractions and indirect light bounces [10]. This section describes the proposed approach to synthesize highly photorealistic images of objects in indoor scenes, which includes modeling of the objects and the scenes (Fig. 2), arranging the object models in the scene models, generating camera poses, and rendering images of the object arrangements.

3.1. Scene and Object Modeling

3D Object Models. We worked with 3D models of 15 objects from LineMod (LM) [14] and 14 objects from Rutgers APC (RU-APC) [2]. We used the refined models from BOP [35], provided as colored meshes with surface normals, and manually assigned them material properties. A Lambertian material was used for the RU-APC models as the objects are made mostly of cardboard. The LM models were assigned material properties which match their appearance in real images. The *specular*, *metallic* and *roughness* parameters of the Arnold renderer [10] were used to control the material properties.

3D Scene Models. The object models were arranged and rendered within 3D models of six furnished scenes. Scenes 1–5 represent work and household environments and include fine details and typical objects, e.g. the kitchen scene (Scene 5)

contains dishes in a sink or a bowl of cherries. Scene 6 contains a shelf from the Amazon Picking Challenge 2015 [36].

The scene models were created using standard 3D tools, primarily Autodesk Maya. Scenes 1 and 2 are reconstructions of real-world environments obtained using LiDAR and photogrammetry 3D scans which served as a guide for an artist. Materials were recreated using photographic reference, PBR material scanning [37], and color swatch samples [38]. Scenes 3–5 were purchased online [39], their geometry and materials were refined, and clutter and chaos was added to mimic a real environment. A 3D geometry model of the shelf in Scene 6 was provided in the Amazon Picking Challenge 2015 [36]. Reference imagery of the shelf was used to create textures and materials that match its appearance.

Exterior light was modeled with Arnold Physical Sky [10] which can accurately depict atmospheric effects and time-of-day variation. Interior lights were modeled with standard light sources such as area and point lights.

3.2. Scene and Object Composition

Stages for Objects. In each scene, we manually selected multiple stages to arrange the objects on. A stage is defined by a polygon and is typically located on tables, chairs and other places with distinct structure, texture or illumination. Placing objects on such locations maximizes the diversity of the rendered images. One stage per shelf bin was added in Scene 6.

Object Arrangements. An arrangement of a set of objects was generated in two steps: (1) poses of the 3D object models were instantiated above one of the stages, and (2) physically plausible arrangements were reached using physics simulation where the objects fell on the stages under gravity and underwent mutual collisions. The poses were initialized using FLARE [40], a rule-based system for generation of object layouts for AR applications. The initial height above the stage was randomly sampled from 5 to 50cm. For LM objects, we staged one instance per object model and initialized it with the canonical orientation, i.e. the cup was up-right, the cat was standing on her legs, etc. For RU-APC objects, we staged up to five instances per object model and initialized their orientation randomly. Physics was simulated using NVIDIA PhysX.

Camera Positioning. Multiple cameras were positioned around each object arrangement. Instead of fitting all the objects within the camera frustum, we point the camera at a randomly selected object. This allows for a better control of scale of the rendered objects. The azimuth and elevation of the camera was generated randomly and the distance of the camera from the focused object was sampled from a specified range. Before rendering the RGB image, which is computationally expensive, we first rendered a mask of the focused object and a mask of its visible part. We then calculated the visible fraction of the focused object and rendered the full RGB image only if the object was at least 30% visible.

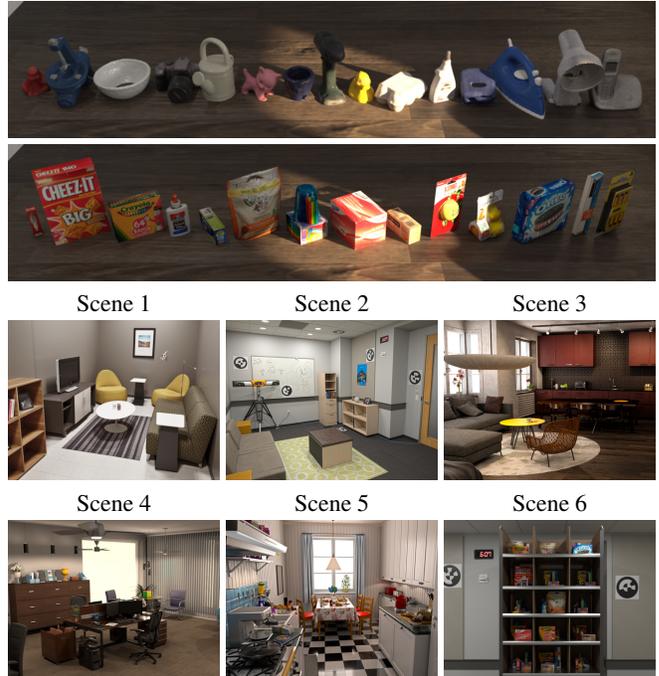


Fig. 2. 3D object models from LineMod [14] (first row) and Rutgers APC [2] (second row) were rendered in six scenes.

3.3. Physically Based Rendering (PBR)

Three images were rendered from each camera using the Arnold physically based renderer [10] at *low*, *medium* and *high* quality settings – the mapping between the quality settings and Arnold parameters can be found in the supplementary material. Rendering was done on 16-core Intel Xeon 2.3GHz processors with 112GB RAM. GPUs were not used. The average rendering time was 15s in low, 120s in medium, and 720s in high quality settings. We used a CPU cluster with 400 nodes which allowed us to render 2.3M images in low, 288K in medium, or 48K in high quality within a day.

We rendered 1.9M object instances in 1K arrangements in the six scenes, seen from 200K cameras. With the three quality settings, we obtained a total of 600K VGA resolution images. LM objects were rendered in Scenes 1–5 and RU-APC objects in Scenes 3 and 6. Each object instance is annotated with a 2D bounding box, a segmentation mask and a 6D pose.

4. EXPERIMENTS

The experiments evaluate the effectiveness of PBR images for training the Faster R-CNN object detector [1]. Specifically, the experiments focus on three aspects: (1) importance of the PBR images over the commonly used images of objects rendered on top of random photographs, (2) importance of the high PBR quality, and (3) importance of scene context.

Datasets. The experiments were conducted on two datasets, LineMod-Occluded (LM-O) [3, 14] and Rutgers APC (RU-APC) [2]. We used their reduced versions from BOP [35]. The datasets include 3D object models and real test RGB-D images of VGA resolution (only RGB channels were used). The images are annotated with ground-truth 6D object poses from which we calculated 2D bounding boxes used for evaluation of the 2D object detection task. LM-O contains 200 images with ground truth annotations for 8 LM objects captured with various levels of occlusion. RU-APC contains 14 object models and 1380 images which show the objects in a cluttered warehouse shelf. Example test images are in Fig. 1.

Baseline Training Images (BL). We followed the synthetic data generation pipeline presented in [16] and used OpenGL to render 3D object models on top of randomly selected real photographs pulled from NYU Depth Dataset V2 [41]. For a more direct comparison, the BL images were rendered from the same cameras as the PBR images, i.e. the objects appear in the same poses in both types of images. Generation of one BL image took 3s on average. Examples are in the supplement.

Object Instance Detection. We experimented with two underlying network architectures of Faster R-CNN: ResNet-101 [42] and Inception-ResNet-v2 [43]. The networks were pre-trained on Microsoft COCO [44] and fine-tuned on synthetic images for 100K iterations. The learning rate was set to 0.001 and multiplied by 0.96 every 1K iterations. To virtually increase diversity of the training set, the images were augmented by randomly adjusting brightness, contrast, hue, and saturation, and by applying random Gaussian noise and blur. Although the presented results were obtained with this data augmentation, we found its effect negligible. Implementation from Tensorflow Object Detection API [45] was used.

Evaluation Metric. The performance was measured by $mAP@.75IoU$, i.e. the mean average precision with a strict IoU threshold of 0.75 [44]. For each test image, detections of object classes annotated in the image were considered.

4.1. Importance of PBR Images for Training

On RU-APC, Faster R-CNN with Inception-ResNet-v2 trained on high quality PBR images achieves a significant 24% absolute improvement of $mAP@.75IoU$ over the same model trained on BL images (Tab. 1, PBR-h vs. BL). It is noteworthy that PBR images yield almost 35% or higher absolute improvement on five object classes, and overall achieve a better performance on 12 out of 14 object classes (see the supplement for the per-class scores). This is achieved when the objects are rendered in Scene 6. When the objects are rendered in Scene 3 (PBR-ho vs. BL), we still observe a large improvement of 11% (scene context is discussed in Sec. 4.3). Improvements, although not so dramatic, can be observed also with ResNet-101. On LM-O, PBR images win by almost 11%, with a large improvement on 7 out of 8 object classes.

Dataset	Architecture	PBR-h	PBR-l	PBR-ho	BL
LM-O	Inc.-ResNet-v2	55.9	49.8	–	44.7
	ResNet-101	49.9	44.6	–	45.1
RU-APC	Inc.-ResNet-v2	71.9	72.9	58.7	48.0
	ResNet-101	68.4	65.1	51.6	52.7

Table 1. Performance ($mAP@.75IoU$) of Faster R-CNN trained on high and low quality PBR images (PBR-h, PBR-l), high quality PBR images of out-of-context objects (PBR-ho), and images of objects on top of random photographs (BL).

4.2. Importance of PBR Quality

On LM-O, we observe that Faster R-CNN with Inception-ResNet-v2 trained on high quality PBR images achieves an improvement of almost 6% over low quality PBR images (Tab. 1, PBR-h vs. PBR-l). This suggests that a higher PBR quality helps. We do not observe a similar improvement on RU-APC when training on PBR images rendered in Scene 6. The illumination in this scene is simpler, there is no incoming outdoor light and the materials are mainly Lambertian. There are therefore no complex reflections and the low quality PBR images from this scene are cleaner than, e.g., when rendered in Scenes 3–5. This suggests that the low quality is sufficient for scenes with simpler illumination and materials. Example images of the two qualities are in the supplement.

4.3. Importance of Scene Context

Finally, we analyze the importance of accurately modeling the scene context. We rendered RU-APC objects in two setups: 1) *in-context* in Scene 6, and 2) *out-of-context* in Scene 3 (examples are in the supplement). Following the taxonomy of contextual information from [46], the in-context setup faithfully model the gist, geometric, semantic, and illumination contextual aspects of the test scene. The out-of-context setup exhibit discrepancies in all of these aspects. Training images of in-context objects yield an absolute improvement of 13% with Inception-ResNet-v2 and 16% with ResNet-101 over the images of out-of-context objects. This shows the benefit of accurately modeling context of the test scene.

5. CONCLUSION

We have proposed an approach to synthesize highly photorealistic images of 3D object models and demonstrated their benefit for training the Faster R-CNN object detector. In the future, we will explore the use of photorealistic rendering for training models for other vision tasks. A dataset of 600K photorealistic images will be released on the project website.

We acknowledge K. Bekris and C. Mitash for help with the RU-APC dataset. This work was supported by V3C – Visual Computing Competence Center (program TE01020415 funded by Technology Agency of the Czech Republic), Software Competence Center Hagenberg, and Research Center for Informatics (project CZ.02.1.01/0.0/0.0/16.019/0000765 funded by OP VVV).

6. REFERENCES

- [1] Ren, et al., “Faster R-CNN: towards real-time object detection with region proposal networks,” *TPAMI*, 2017. 1, 2, 3
- [2] Rennie, et al., “A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place,” *Robotics and Automation Letters*, 2016. 1, 2, 3, 4
- [3] Brachmann, et al., “Learning 6D object pose estimation using 3D object coordinates,” in *ECCV*, 2014. 1, 2, 4
- [4] Richter, et al., “Playing for data: Ground truth from computer games,” in *ECCV*, 2016. 1, 2
- [5] Su, et al., “Render for CNN: viewpoint estimation in images using CNNs trained with rendered 3D model views,” in *ICCV*, 2015. 1, 2
- [6] Rozantsev, et al., “Beyond sharing weights for deep domain adaptation,” *TPAMI*, 2018. 1
- [7] Csurka, “A comprehensive survey on domain adaptation for visual applications,” in *Domain Adaptation in Computer Vision Applications*. 2017. 1, 2
- [8] Li and Snavely, “CGIntrinsics: Better intrinsic image decomposition through physically-based rendering,” in *ECCV*, 2018. 1, 2
- [9] Zhang, et al., “Physically-based rendering for indoor scene understanding using convolutional neural networks,” in *CVPR*, 2017. 1, 2
- [10] Georgiev, et al., “Arnold: A brute-force production path tracer,” *TOG*, 2018. 1, 2, 3
- [11] Pharr, et al., *Physically based rendering: From theory to implementation*, Morgan Kaufmann, 2016. 1
- [12] Shreiner, *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*, Pearson Education, 2009. 1
- [13] Marschner and Shirley, *Fundamentals of computer graphics*, CRC Press, 2015. 1
- [14] Hinterstoisser, et al., “Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes,” in *ACCV*, 2012. 1, 2, 3, 4
- [15] Dwibedi, et al., “Cut, paste and learn: Surprisingly easy synthesis for instance detection,” in *ICCV*, 2017. 2
- [16] Hinterstoisser, et al., “On pre-trained image features and synthetic images for deep learning,” in *ECCVW*, 2018. 2, 4
- [17] Kehl, et al., “SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again,” in *ICCV*, 2017. 2
- [18] Rad and Lepetit, “BB8: a scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth,” in *ICCV*, 2017. 2
- [19] Dosovitskiy, et al., “Flownet: Learning optical flow with convolutional networks,” in *ICCV*, 2015. 2
- [20] Tekin, et al., “Real-time seamless single shot 6d object pose prediction,” in *CVPR*, 2018. 2
- [21] Dvornik, et al., “Modeling visual context is key to augmenting object detection datasets,” in *ECCV*, 2018. 2
- [22] Movshovitz-Attias, et al., “How useful is photo-realistic rendering for visual learning?,” in *ECCV*, 2016. 2
- [23] Tremblay, et al., “Deep object pose estimation for semantic robotic grasping of household objects,” in *CoRL*, 2018. 2
- [24] Richter, et al., “Playing for benchmarks,” in *ICCV*, 2017. 2
- [25] Ros, et al., “The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes,” in *CVPR*, 2016. 2
- [26] Gaidon, et al., “Virtual worlds as proxy for multi-object tracking analysis,” in *CVPR*, 2016. 2
- [27] Handa, et al., “Understanding real world indoor scenes with synthetic data,” in *CVPR*, 2016. 2
- [28] McCormac, et al., “SceneNet RGB-D: Can 5M synthetic images beat generic imagenet pre-training on indoor segmentation?,” in *ICCV*, 2017. 2
- [29] Tobin, et al., “Domain randomization for transferring deep neural networks from simulation to the real world,” in *IROS*, 2017. 2
- [30] Tremblay, et al., “Training deep networks with synthetic data: Bridging the reality gap by domain randomization,” *arXiv preprint arXiv:1804.06516*, 2018. 2
- [31] Sundermeyer, et al., “Implicit 3D orientation learning for 6D object detection from RGB images,” in *ECCV*, 2018. 2
- [32] Wood, et al., “Rendering of eyes for eye-shape registration and gaze estimation,” in *ICCV*, 2015. 2
- [33] Wood, et al., “Learning an appearance-based gaze estimator from one million synthesised images,” in *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, 2016. 2
- [34] Shrivastava, et al., “Learning from simulated and unsupervised images through adversarial training,” in *CVPR*, 2017. 2
- [35] Hodan, et al., “BOP: Benchmark for 6D object pose estimation,” in *ECCV*, 2018. 2, 4
- [36] Yu, et al., “A summary of team MIT’s approach to the Amazon Picking Challenge 2015,” *arXiv preprint arXiv:1604.03639*, 2016. 3
- [37] “Mura,” www.muravision.com. 3
- [38] “NIX Color Sensor,” www.nixsensor.com. 3
- [39] “Evermotion,” www.evermotion.org. 3
- [40] Gal, et al., “FLARE: fast layout for augmented reality applications,” in *ISMAR 2014*, 2014. 3
- [41] Silberman, et al., “Indoor segmentation and support inference from rgb-d images,” in *ECCV*, 2012. 4
- [42] He, et al., “Deep residual learning for image recognition,” in *CVPR*, 2016. 4
- [43] Szegedy, et al., “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *AAAI*, 2017. 4
- [44] Lin, et al., “Microsoft COCO: Common objects in context,” in *ECCV*, 2014. 4
- [45] Huang, et al., “Speed/accuracy trade-offs for modern convolutional object detectors,” in *CVPR*, 2017. 4
- [46] Divvala, et al., “An empirical study of context in object detection,” in *CVPR*, 2009. 4

PHOTOREALISTIC IMAGE SYNTHESIS FOR OBJECT INSTANCE DETECTION SUPPLEMENTARY MATERIAL

Tomáš Hodaň^{1,2} Vibhav Vineet² Ran Gal² Emanuel Shalev² Jon Hanzelka²
Treb Connell² Pedro Urbina² Sudipta N. Sinha² Brian Guenter²

¹FEE, Czech Technical University in Prague ²Microsoft Research

This supplement provides examples of the baseline training images in Fig. 1, a visualization of the object pose generation process in Fig. 2, images of out-of-context RU-APC objects in Fig. 3, examples of high quality PBR images in Fig. 4, a comparison of low/high PBR quality in Fig. 5, example results of the Faster R-CNN object detector [1] trained on high quality PBR images in Fig. 6, per-class detection scores in Tab. 2 and Tab. 3, and a description of the PBR quality settings below.

PBR Quality Settings. Tab. 1 shows the mapping between the used quality settings and the Arnold parameters [2]. Increasing the number of rays traced per image pixel (AA) reduces aliasing artifacts caused by insufficient sampling of geometry and noise caused by insufficient sampling of illumination. Increasing the number of diffuse rays traced when a ray hits a diffuse surface (D rays) and the number of rays traced when the ray hits a specular surface (S rays) reduces illumination noise. Increasing the number of diffuse and specular reflections (D depth and S depth) improves the accuracy of the illumination integral by gathering more of the light energy bounced around in the scene. This is especially important in the case when photons must bounce multiple times from the light source to reach an object visible to the camera.

1. REFERENCES

- [1] Ren, et al., “Faster R-CNN: towards real-time object detection with region proposal networks,” *TPAMI*, 2017. 1, 3
- [2] Georgiev, et al., “Arnold: A brute-force production path tracer,” *TOG*, 2018. 1
- [3] Gal, et al., “FLARE: fast layout for augmented reality applications,” in *ISMAR 2014*, 2014. 1
- [4] Hinterstoisser, et al., “Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes,” in *ACCV*, 2012. 2
- [5] Rennie, et al., “A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place,” *Robotics and Automation Letters*, 2016. 2, 3
- [6] Brachmann, et al., “Learning 6D object pose estimation using 3D object coordinates,” in *ECCV*, 2014. 3
- [7] Hodan, et al., “BOP: Benchmark for 6D object pose estimation,” in *ECCV*, 2018. 4



Fig. 1. Examples of baseline training images generated by rendering 3D object models on top of random photographs.



Fig. 2. Initial object poses generated using FLARE [3] (left), and final poses calculated by NVIDIA PhysX (right).



Fig. 3. Training images of out-of-context RU-APC objects.

Setting	AA	D rays	S rays	D depth	S depth	Max depth
low	1	1	1	1	1	2
medium	9	36	36	3	2	3
high	25	225	100	3	3	4

Table 1. Arnold parameters [2] for different quality settings.

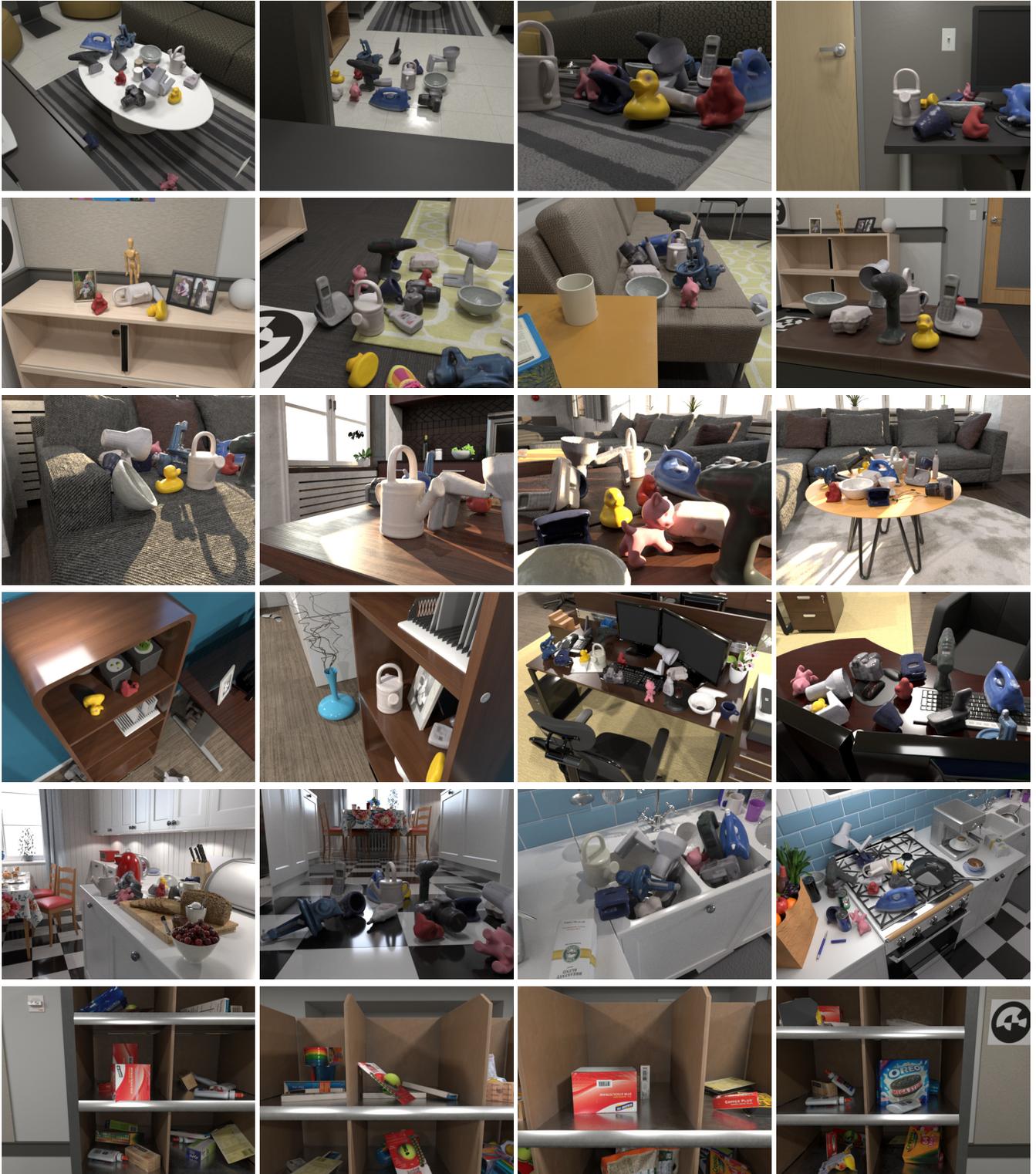


Fig. 4. Examples of high quality PBR images of objects from the LineMod dataset [4] in Scenes 1–5 (top five rows), and images of objects from the Rutgers APC dataset [5] in Scene 6 (bottom row). The images were automatically annotated with 2D bounding boxes, masks and 6D poses of visible object instances.

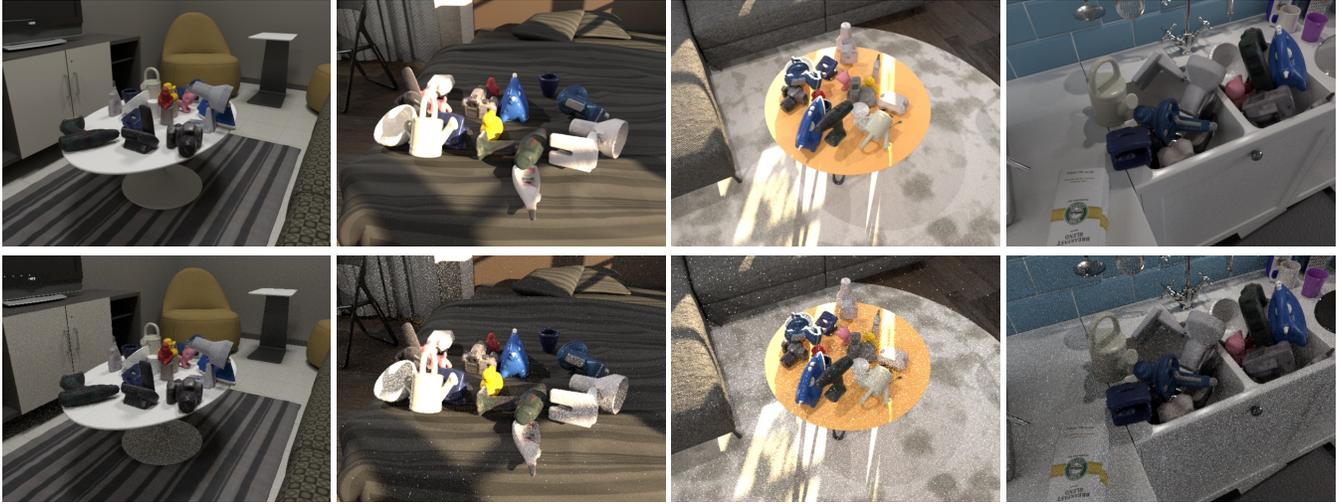


Fig. 5. The same images rendered in high (top) and low (bottom) PBR quality.

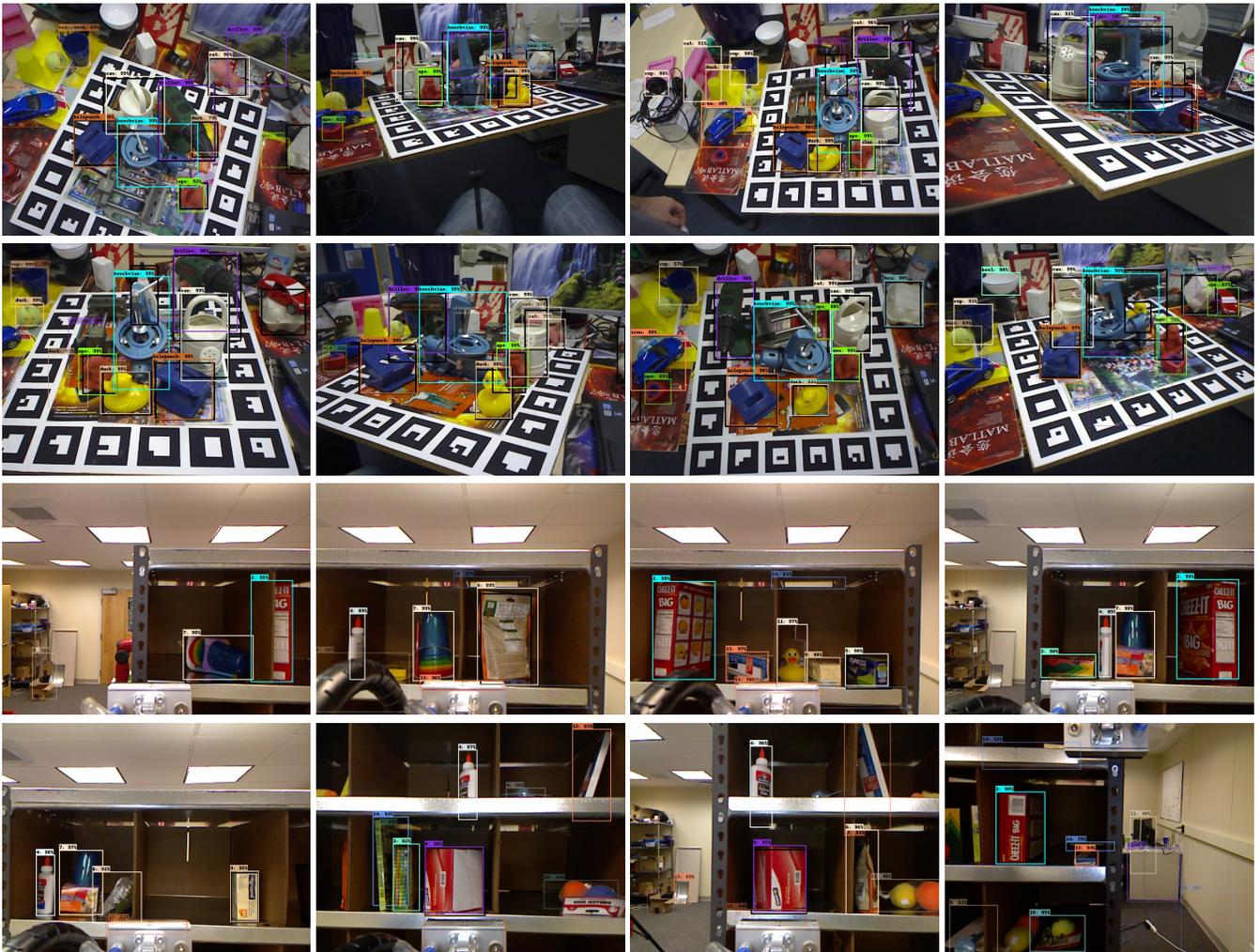


Fig. 6. Example results of the Faster R-CNN object detector [1] trained on high quality PBR images and evaluated on real test images from the LineMod-Occluded dataset [6] (top two rows) and the Rutgers APC dataset [5] (bottom two rows).

Data/Obj. ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	mAP
Inception-ResNet-v2															
PBR-h	57.1	93.3	88.0	61.2	80.4	62.5	99.0	98.1	73.2	44.8	65.4	70.9	86.8	26.4	71.9
PBR-l	57.6	96.3	84.6	62.2	81.3	60.5	98.7	98.6	73.1	44.8	79.3	67.2	90.5	25.6	72.9
PBR-ho	46.2	61.9	56.0	55.8	54.4	69.8	89.0	89.3	81.6	21.5	72.7	58.3	43.3	21.7	58.7
BL	33.5	47.5	71.5	32.7	42.4	13.5	44.9	73.0	57.4	44.5	47.6	35.8	87.6	40.6	48.0
ResNet-101															
PBR-h	30.6	93.7	91.6	68.2	72.1	56.7	93.4	93.6	75.2	42.6	84.5	60.9	73.2	21.4	68.4
PBR-l	26.8	87.6	87.3	64.0	79.8	27.8	95.2	90.4	66.2	37.5	83.1	61.4	79.3	25.3	65.1
PBR-ho	35.2	64.4	58.4	52.9	46.7	53.0	71.5	73.8	69.3	32.2	66.2	51.8	28.3	19.3	51.6
BL	29.1	38.5	82.0	59.2	52.4	59.1	79.5	75.0	36.4	36.8	75.1	50.6	48.5	14.8	52.7

Table 2. Object detection scores on RU-APC: Per-class average precision (AP@.75IoU) and the mean average precision (mAP@.75IoU) of Faster R-CNN trained on (i) high/low quality PBR images of in-context objects rendered in Scene 6 (PBR-h, PBR-l), (ii) high quality PBR images of out-of-context objects rendered in Scene 3 (PBR-ho), and (iii) images of objects rendered on top of random photographs (BL). The object identifiers follow the BOP convention [7].

Data/Obj. ID	1	5	6	8	9	10	11	12	mAP
Inception-ResNet-v2									
PBR-h	60.3	44.5	56.7	53.4	81.8	48.6	9.6	92.3	55.9
PBR-l	57.3	35.8	53.3	52.6	77.8	23.8	3.1	94.5	49.8
BL	30.7	45.4	42.5	32.4	77.1	33.4	19.6	76.7	44.7
ResNet-101									
PBR-h	46.3	40.3	48.5	58.0	76.4	39.5	4.7	85.5	49.9
PBR-l	44.1	26.6	41.6	53.7	73.7	24.5	1.1	91.6	44.6
BL	35.5	45.3	37.1	44.6	75.0	33.6	12.7	76.8	45.1

Table 3. Object detection scores on LM-O: Per-class average precision (AP@.75IoU) and the mean average precision (mAP@.75IoU) of Faster R-CNN trained on (i) high/low PBR images of objects rendered in Scenes 1–5 (PBR-h, PBR-l), and (ii) images of objects rendered on top of random photographs (BL). The object identifiers follow the BOP convention [7].