IEEE International Conference on Robotics and Automation (ICRA), Paris, France, May 2020.

Stillleben: Realistic Scene Synthesis for Deep Learning in Robotics

Max Schwarz* and Sven Behnke

***Abstract*— Training data is the key ingredient for deep learning approaches, but difficult to obtain for the specialized domains often encountered in robotics. We describe a synthesis pipeline capable of producing training data for cluttered scene perception tasks such as semantic segmentation, object detection, and correspondence or pose estimation. Our approach arranges object meshes in physically realistic, dense scenes using physics simulation. The arranged scenes are rendered using high-quality rasterization with randomized appearance and material parameters. Noise and other transformations introduced by the camera sensors are simulated. Our pipeline can be run online during training of a deep neural network, yielding applications in life-long learning and in iterative render-and-compare approaches. We demonstrate the usability by learning semantic segmentation on the challenging YCB-Video dataset without actually using any training frames, where our method achieves performance comparable to a conventionally trained model. Additionally, we show successful application in a real-world regrasping system.**

I. Introduction

While the rise of deep learning for computer vision tasks has brought new capabilities to roboticists, such as robust scene segmentation, pose estimation, grasp planning, and many more, one of the key problems is that deep learning methods usually require large-scale training datasets. This is less of a problem for the computer vision community, where researchers can work on the available public datasets, but roboticists face a key restriction: Their methods usually need to be deployed in a particular domain, which is often not covered by the available large-scale datasets. Capturing a custom training dataset just for one specific purpose is often infeasible, because it involves careful planning, scene building, capturing, and usually manual annotation.

There are techniques for reducing the amount of required training data. Transfer learning, usually in the form of fine-tuning, where a network fully trained on a generic, large dataset is further trained on a smaller domain-specific dataset is the preferred method in these situations, as e.g. evident by the winning approaches at the Amazon Robotics Challenge (ARC) 2017 [1], [2]. Here, the dominant methods performed their fine-tuning on synthetic datasets generated from monocular object images. While picking novel items after roughly 30 min of capture/training time is an impressive feat, these methods are limited by the 2D composition of their synthetic scenes. The resulting arrangements are often not physically realistic and fail to model key effects, such as correct lighting, shadows, and contact interactions. Furthermore, the resulting images are only annotated with ground truth segmentation, prohibiting the training of more abstract tasks like pose or correspondence estimation.

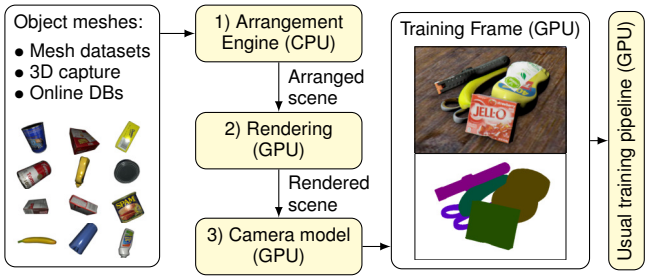


Fig. 1. Architecture of our online scene synthesis library. Meshes from various sources are arranged in realistic configurations by the Arrangement Engine. The render module generates high-quality renders, which are then imbued with typical noise and transformations caused by commodity cameras. Finally, a usual training pipeline can follow.

Still, bolstered by the success of synthetic data generation during the ARC, we want to ask the question: Are large-scale image datasets still necessary for robotic perception?

To address the problems of the mentioned 2D synthesis methods, we extend the idea to 3D. We propose a scene synthesis pipeline consisting of physically realistic placement, realistic 3D rendering, modeling of camera effects and automatic ground truth data generation for a variety of tasks. Our system focuses on highly cluttered arrangements of objects on a planar supporting surface, although it can be easily adapted to various support or container shapes. A physics engine is used to place the objects on the support surface. The scene is then rendered using standard GPU rasterization techniques.

In short, our contributions include:

1) An online, highly efficient scene synthesis pipeline with automatic ground truth annotation for training deep learning models,
2) application of said pipeline to the task of scene segmentation on the challenging YCB-Video dataset without actually using any training frames, where it reaches comparable performance to a conventionally-trained model, and
3) application in a real-world robotic system, where it is successfully used to train a combined segmentation and pose estimation network.

Our scene synthesis library named *stillleben*, after the German term for still live paintings, contains bindings to the popular PyTorch deep learning framework and is available as open source[1]. It will thus hopefully become a helpful tool for robotics researchers.

*All authors are with the Autonomous Intelligent Systems group of University of Bonn, Germany; schwarz@ais.uni-bonn.de

[1]https://github.com/AIS-Bonn/stillleben

(a) YCB-Video [5] (b) YCB-Video (syn) [5] (c) FAT [6]

Fig. 2. YCB-Video dataset and related synthetic datasets.

II. Related Work

One of the inspiring works for us is the introduction of *Domain Randomization* by Tobin *et al.* [3], who demonstrated learning of an object detection task on entirely synthetic data, with later execution on a real robotic system. The key insight by the authors is that the reality gap between synthetic and real data can be bridged by randomizing the parameters of the simulation generating the synthetic scenes. With enough "spread" of the parameters, situations close to the real domain will be generated and learned. In contrast to their method, which is limited to non-cluttered arrangements of simple shapes in a fixed 2D scene, our method offers a flexible way to simulate dense, cluttered arrangements in 3D, while modeling and randomizing more visual effects. Andrychowicz *et al.* [4] demonstrate applicability of the domain randomization idea to a manipulation task by learning to rotate a cube in-hand. Their system can successfully execute rotations to target faces in a real robotic setup despite having learned only in simulation.

Many recent works in pose estimation make usage of synthetic data for training their models. However, many treat synthetic data as an *augmentation* technique for real data. For example, Xiang *et al.* [5] render synthetic images in addition to their YCB-Video dataset for training their pose estimation method (see Fig. 2). Oberweger *et al.* [7] use a similar strategy, but use an intermediate feature mapping network to learn and abstract away domain differences. In both works, the use of synthetic data is not a key point and is thus not systematically analyzed.

Tremblay *et al.* [6] introduce a large-scale dataset called Falling Things (FAT), which is rendered from the YCB Object set, similar to our experiments in Section IV. However, their method is offline and not focused on dense, cluttered object arrangements. Furthermore, no comparison of a synthetically trained model against one trained with real data is attempted. In our experiments, we compare against a model trained on FAT. In further work, Tremblay *et al.* [8] present a pose estimation method especially suited for training on synthetic data. The authors demonstrate that the system can be successfully trained on a combination of simple domain-randomized images and the high-quality FAT images and outperforms PoseCNN [5], which was trained on the real YCB-Video Dataset. Whether the increase in performance is due to the training on synthetic data or due to algorithmic differences to PoseCNN remains unclear, though.

Zhang *et al.* [9] present a large-scale synthetic dataset for indoor scene segmentation. The renders are of very high quality and yield improvements in segmentation, normal estimation, and boundary detection when used for pretraining. In contrast to our work, the scenes were manually designed and annotated with physically realistic material properties. To achieve photorealism, the authors used a ray tracer for rendering, whereas our OpenGL-based renderer is fast enough for online usage.

Kar *et al.* [10] (published after submission of our work) learn a generative model for synthesizing annotated 3D scenes. Their approach automatically tunes many of the hyperparameters introduced in our method, but only optimizes the scene graph, excluding camera noise modeling and rendering options. Furthermore, the approach requires a comparison dataset large enough to estimate the scene distribution, though labels are not needed.

There are other works on online rendering in deep learning settings, usually for render-and-compare and mesh reconstruction applications. Kato *et al.* [11] reconstruct meshes from 2D images using a differentiable rendering module. Their approach, however, ignores texture. Li *et al.* [12] implement an iterative render-and-compare method for pose estimation. In contrast to our work, they focus on single objects. Furthermore, the rendering does not need to be particularly realistic, since a network trained with pairs of synthetic and real images computes the pose delta for the next iteration.

III. Method

Our method design is driven by several goals: To be usable in life-long learning scenarios (such as the one modeled in the ARC competition), it should be usable *online*, i.e. without a separate lengthy rendering process. This implies that the system needs to be *efficient*, especially since we do not want to take GPU power away from the actual training computations. It also needs to be *flexible*, so that users can quickly adapt it to their target domain.

An overview of the architecture is shown in Fig. 1.

A. Object Mesh Database

Input to our pipeline are object meshes. These can come from a variety of sources: Ideally, 3D scans of the target objects are available, as these are usually the most precise description of the object geometry and its appearance. Another possibility are CAD designs. There are also large-scale databases of 3D meshes which can be applicable, such as ShapeNet [13], although they often lack texture. Finally, there are online communities such as Sketchfab [2], which allow the retrieval (and purchase) of high-quality mesh models. For physics simulation in the next pipeline step, the meshes need to be annotated with inertial properties. If there is no further information, we assume a uniform density of $500\,\mathrm{kg\,m^{-3}}$ and compute mass and inertial properties from this.

[2]Sketchfab: `https://www.sketchfab.com`

Fig. 3. Scenes generated by our physics-based arrangement engine. Object meshes are taken from the YCB Object set. Background images are randomly selected from an Internet image search with keywords "table surface".

B. Arrangement Engine

The arrangement engine is responsible for generating physically realistic object configurations in the given scene. Since we focus on objects resting in cluttered configurations on a planar support surface, we choose a plane with random normal[3] n_p and support point $p = (0\ 0\ d)^T$ in the camera coordinate system. The chosen objects (in our experiments five at a time) are then inserted into the scene at random poses above the plane. Using the PhysX physics engine by NVIDIA[4], we then simulate the objects' behavior under gravity in direction $-n_p$ along with a weaker force of $1\,\mathrm{N}$ drawing the objects towards the support point p. The arrangement engine runs entirely on CPU to spare GPU power for rendering and the training process itself. To make online operation feasible, we reduce mesh complexity using Quadric Edge Simplification [14] to a target number of faces $F = 2000$. Since PhysX (like all major physics engines) can only simulate convex dynamic bodies, we then compute a convex hull for each mesh. This obviously excludes arrangements that make use of concavities, but initial experiments showed that the impact is negligible. To cover remaining configurations that are not captured by our arrangement engine, we randomly fall back to a simpler arrangement procedure that simply samples collision-free object poses (i.e. without simulating gravity). Resulting arrangements can be seen in Fig. 3.

C. Rendering

The rendering process is implemented as standard GPU rasterization using the Magnum OpenGL engine[5]. A single rendering pass suffices, as both color and semantic information such as class and instance segmentation and point coordinates can be written into the output by a custom fragment shader.

We implemented several extensions that model particular effects arising in densely cluttered scenes:

Physically-based Rendering (PBR) refers to the employment of physically-motivated bidirectional reflectance distribution functions (BRDF). The usual choice of the Cook-Torrance BRDF [15] has two material properties, usually called metalness and roughness. Since we assume no prior knowledge, we set these parameters randomly for each object on each render.

Image-based Lighting (IBL) allows us to place the object into complex lighting situations without explicitly modeling all light sources. This yields diffuse and specular light responses on the object surfaces, which would be difficult to generate explicitly.

Ambient Occlusion (SSAO) is an approximation for the dark shadows caused by close arrangements of objects. Since the light causing these shadows is ambient, i.e. caused by multiple light bounces of environment and object surfaces, traditional shadow mapping techniques cannot model this effect within the rasterization framework. Screen Space Ambient Occlusion approximates it by sampling possible occluders in screen space.

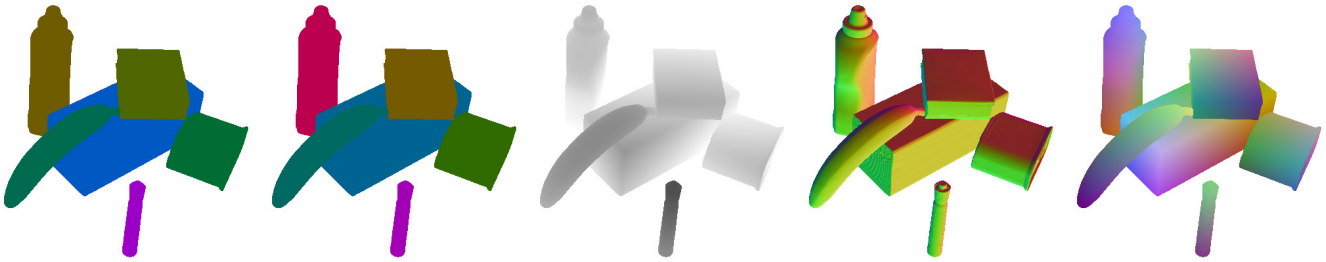
For more information about these well-known techniques, we refer to the book by Fernando [16].

We further developed a texture modification method driven by the fact that objects are often slightly modified from their original appearance, e.g. by placing a bar code sticker or product label on them. We model these effects by randomly projecting an image randomly selected from an Internet

[3]In case n_p is pointing away from the camera, it is flipped.
[4]PhysX: `https://developer.nvidia.com/physx-sdk`
[5]Magnum: `https://magnum.graphics`

(a) real (b) Phong shading (c) IBL + PBR (d) ... + SSAO (e) ... + Stickers

Fig. 4. Rendering extensions. For comparison, a real training image from the YCB-Video dataset is shown in (a). The other images show the effect of the discussed rendering extensions on a synthetic scene rendered with the same poses. Note that material settings for PBR (d) are randomized—which is why the mustard bottle looks metallic.



(a) class segmentation (b) instance segmentation (c) depth (d) normals (e) object-centric coordinates

Fig. 5. Output channels. The library generates a number of different channels usable for training of various perception tasks. The scene is the same as shown in Fig. 4.

search using the keyword "product label" onto the object surface. We call this extension **Sticker Projection (SP)**.

Figure 4 shows the effect of the individual rendering extensions. The contribution of each extension is analyzed in Section IV.

Finally, the background can be customized as well. We render the scene on top of background images, which can be sampled from image datasets or provided by the user, for example when training for a well-known structured background environment in industry applications. Furthermore, the supporting surface can also be textured (see Fig. 3).

The rendering output contains (see Fig. 5):

- Pixel-wise color (the image),
- pixel-wise class and instance segmentation for training semantic segmentation,
- pixel-wise depth and normals for training RGB-D models,
- pixel-wise coordinates in each object's coordinate frame for correspondence training as in [17], and
- each object's pose.

We note that the renderer is equipped with approximative differentiation, allowing backpropagation of image-space gradients to object pose gradients. This functionality has been described by Periyasamy *et al.* [18] in detail.

D. Camera Model

In robotic applications, sensors are often low-cost commodity types, resulting in imperfect captures of the real scene. Additionally, lighting conditions are often difficult, yielding even higher noise levels. Any robust perception model will need to deal with these effects. In the standard case of training on a larger dataset, the noise statistics can be learned from the dataset. In our case, we model camera effects with broadly randomized parameters to obtain a model robust to the perturbations.

We follow the work of Carlson *et al.* [19], who propose a comprehensive camera model including chromatic aberration, blur, exposure, noise, and color temperature. The operations were implemented in PyTorch using CUDA, so that rendered images do not have to be copied back to CPU.

IV. Experiments

Our experiments are carried out on the YCB-Video dataset [5]. This dataset is intended for evaluating 6D pose estimation and scene segmentation methods and contains video sequences captured by a hand-held camera of static object arrangements. The cluttered arrangements and bad lighting conditions make it highly challenging. One highly interesting property is that the objects are drawn from the YCB Object and Model set [20], for which high-quality 3D meshes are available. We sample random background images from ObjectNet3D [21] and supporting plane textures from a top-50 Internet image search with keywords "table surface".

We perform all our training experiments using the lightweight RefineNet architecture [22] on the task of semantic segmentation. Contrary to more high-level tasks, semantic segmentation is straightforward to evaluate due to its pixel-wise nature and is, in many current pipelines, a prerequisite for pose estimation in cluttered scenes.

We train all networks using the Adam optimizer with a learning rate of $1e-5$ and parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ for 450k frames. The networks are evaluated using the mean Intersection-over-Union (IoU) score on the YCB-Video test set (keyframes).

A. Timings

We perform our timing tests on a compute server with 2x Intel Xeon Gold 6248 CPUs running at 2.5 GHz to 3.9 GHz. For rendering and training, an NVIDIA TITAN RTX GPU is used. Running stand-alone, our library generates 640×480 annotated training images with 30 fps when using eight arrangement engine processes in parallel. When training a RefineNet network using the generated frames, the entire pipeline achieves 10 fps using a batch size of four. In comparison, a real dataset read from disk yields 13 fps. If the small drop in performance needs to be avoided, a second GPU could be used for rendering, hiding the rendering cost by rendering in parallel to training.

B. Semantic Segmentation

Models trained with our synthesized data obtain roughly 85 % performance compared to a baseline trained on the real dataset (see Table I). We anticipated that we would not be able to outperform the real baseline, since there are many complex effects our arrangement engine and renderer cannot capture. Additionally, some of the objects have radical differences to their object meshes, some of which our sticker projection module cannot produce (see Fig. 6). To investigate

TABLE I

SEGMENTATION RESULTS AND ABLATION STUDY ON YCB-VIDEO

Training data source	Mean IoU	Relative
Real + stillleben	0.800	1.053
Real dataset	**0.760**	**1.000**
Real dataset (50%)	0.697	0.917
Real dataset (25%)	0.597	0.786
with real poses	0.691	0.909
stillleben (ours)	**0.656**	**0.863**
w/o stickers	0.635	0.836
w/o SSAO	0.632	0.832
w/o PBR + IBL	0.648	0.853
w/o cam model	0.128	0.168
Falling Things [6]	0.632	0.832
YCB-Video synthetic [5]	0.300	0.395

Fig. 6. Deviations of the real objects from their meshes: The top row shows crops of YCB-Video test frames, the bottom row their corresponding renders. The pitcher has a sticker on it, the cleaner bottle is wrapped with tape, and the wood block seems to have entirely different texture and surface finish. Furthermore, the mesh textures exhibit artifacts from the scanning process, such as the vertical reflection stripes on the cup.

the remaining gap, we also performed a training run with real poses, i.e. object configurations taken from the YCB-Video training set, but rendered using our pipeline. As this model achieves 91 % compared to the baseline, we conclude that there are both arrangement and visual effects our pipeline cannot yet capture.

When the size of the real YCB-Video training dataset is artificially reduced, baseline performance drops below our model trained with synthetic data (see Table I). As the YCB-Video dataset is quite large (265 GiB), we conclude that our synthesis pipeline can replace capture of a large-scale dataset while attaining comparable performance.

To judge the importance of the implemented rendering extensions, we performed an ablation study (see Table I). While the camera model with its blurring and introduction of noise is apparently crucial for bridging the gap between rendered and real images (we hypothesize this step hides obvious rendering artifacts which would otherwise be learned), the other techniques yield more modest improvements.

Finally, we compare the usability of our generated data

Fig. 7. Collection of 24 driller meshes for category-level generalization experiment. Meshes obtained from `https://sketchfab.com`.

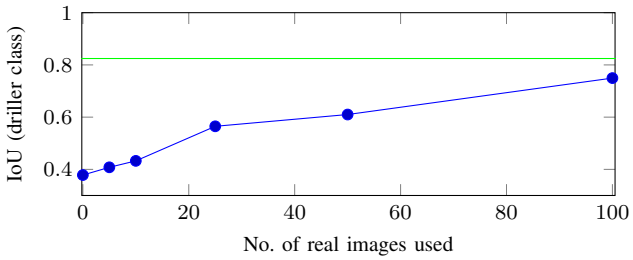


Fig. 8. Category-level generalization and improvement using few real images. We show the segmentation performance (specifically the driller class) vs. the number of real images from YCB-Video that were used during training. The green line shows the IoU score of the model trained on the full YCB-Video training set.

against two offline-generated datasets using the YCB-Video objects. The original synthetic images distributed with the YCB-Video dataset [5] yield very suboptimal performance—but they were never intended for standalone usage without the real dataset. We outperform the FAT dataset [6] by a small margin, which we think is due to our specialization on dense, cluttered arrangements, whereas FAT contains more spread-out arrangements with less occlusions.

C. Category-Level Generalization

The YCB-Video Dataset recommends itself to these experiments, since it contains high-quality meshes corresponding to the exact instances used in the dataset. We maintain that this is a valid use case: Commercial 3D scanners can generate high-quality 3D models from real objects in minutes, which can then be used to train perception systems using our pipeline. However, we are also interested in the case where meshes of the exact instance are not available. To investigate the usability of our generated data to train class-level perception that generalizes to unseen instances, we perform an additional experiment on YCB-Video. Here, we select a particular object, the driller, since meshes of drillers are readily available. We withhold the original mesh from our training pipeline and replace it with a collection of 25 driller meshes collected from online databases such as

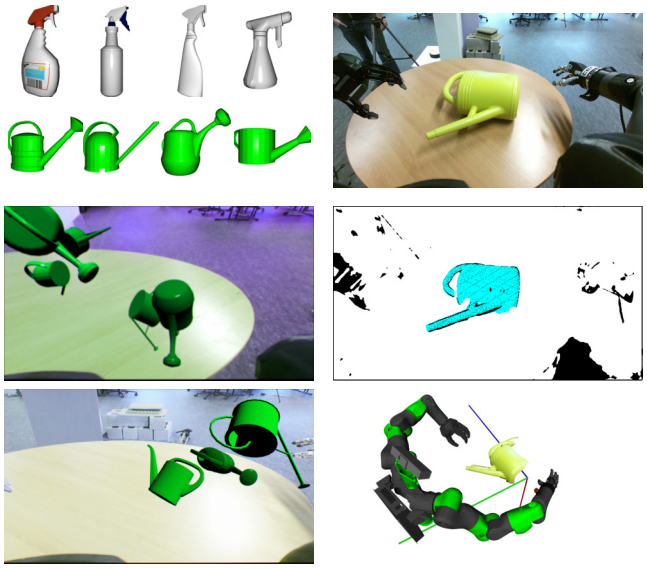


Fig. 9. Regrasping application. Left column: Mesh database and generated watering can training scenes. Right column: Real input image, semantic segmentation (black: positives, cyan: selected region with valid depth), estimated 6D pose. Note that the segmentation has false positives on the robot arms, since they were not part of the training. These were filtered out using kinematic information. Left part taken from [23].

Sketchfab (see Fig. 7).

We show the resulting performance on the YCB-Video test set (focused on the driller class) in Fig. 8. The resulting IoU score is quite low, probably unusable in real-world applications. We conclude that our mesh collection is insufficient to allow the network to generalize to the test instance, maybe due to bad mesh quality or too low variety, allowing the network to learn the different instances by heart. In this situation, the user may augment the synthetic training data by a few manually annotated real images, which will improve accuracy to a usable level (see Fig. 8).

D. Application in a real-world robotic system

While we were not able to test the system using real YCB Objects, it was successfully used for training a CNN-based semantic segmentation and pose estimation in a functional regrasping pipeline by Pavlichenko *et al.* [23]. While [23] contains the system-level description and details about the manipulation planning aspects, we describe the scene synthesis step here.

The target objects investigated in this application were watering cans and spray bottles. Since a 3D scanner was not available, we retrieved 3D models from Sketchfab and other online databases (see Fig. 9). Nearly all meshes were textureless, so we assigned random uniform colors for the spray bottles and uniform green color to the watering cans.

As background scenes, real images captured by the robot, observing an empty scene under different lighting conditions were used. Since the scenes were not expected to be cluttered, only the basic arrangement engine mode was used, where objects are placed randomly in a collision-free manner.

Segmentation was performed using the RefineNet architecture (as before), which was extended to also densely estimate the direction to the object center and the object's rotation as a Quaternion. Combined, this can be used to estimate the full 6D pose [5]. The trained model was used successfully in 53 manipulation trials, out of which the object was segmented with a success rate of 100%. The subsequent highly complex manipulation task was completed with a success rate of 65% [23]. In this setting, which is less complicated than the highly cluttered YCB-Video scenes, our pipeline resulted in robust category-level segmentation and pose estimation without any data capture or manual annotation.

V. Discussion & Conclusion

We have presented a pipeline for generating synthetic scenes of densely cluttered objects from object meshes. Our generated data can be used to train deep segmentation models to comparable performance on the challenging YCB-Video dataset. Additionally, we demonstrated a robotic application including segmentation and pose estimation. We conclude that in these situations, the time- and labor-consuming task of capturing and annotating real datasets can be minimized or skipped altogether in favor of synthetically generated scenes. The pipeline is fast enough for online usage, which opens up new and exciting research opportunities, both in online learning systems that can quickly adapt to changing objects and environments and in iterative systems that can create and compare internal representations to their sensor input by rendering.

We explicitly note that the pipeline is easy to adapt to different object sets and environments—and can thus hopefully lower the barrier to training robust perception systems for custom robotic applications.

Limitations include the dependence on high-quality mesh input, as demonstrated by our generalization experiments. While remaining gaps between training set and real objects can be bridged by the means of additionally captured real images, a purely synthetic solution would be preferable. Furthermore, our arrangement procedure leads to entirely random configurations, whereas humans often place objects in functional ways. To capture this bias, more complex arrangement options could be investigated.

Acknowledgment

This research has been supported by an Amazon Research Award.

References

[1] D. Morrison, A. W. Tow, M McTaggart, R Smith, N Kelly-Boxall, S Wade-McCue, J Erskine, R Grinover, A Gurman, T Hunn, *et al.*, "Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 7757–7764.

[2] M. Schwarz, C. Lenz, G. M. García, S. Koo, A. S. Periyasamy, M. Schreiber, and S. Behnke, "Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 3347–3354.

[3] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 23–30.

[4] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[5] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Robotics: Science and Systems (RSS)*, 2018.

[6] J. Tremblay, T. To, and S. Birchfield, "Falling things: A synthetic dataset for 3d object detection and pose estimation," in *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[7] M. Oberweger, M. Rad, and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3D object pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 119–134.

[8] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Conference on Robot Learning (CoRL)*, 2018.

[9] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser, "Physically-based rendering for indoor scene understanding using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5287–5295.

[10] A. Kar, A. Prakash, M.-Y. Liu, E. Cameracci, J. Yuan, M. Rusiniak, D. Acuna, A. Torralba, and S. Fidler, "Meta-sim: Learning to generate synthetic datasets," in *ICCV*, 2019.

[11] H. Kato, Y. Ushiku, and T. Harada, "Neural 3D mesh renderer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3907–3916.

[12] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep iterative matching for 6D pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[13] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3D model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[14] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 1997, pp. 209–216.

[15] R. L. Cook and K. E. Torrance, "A reflectance model for computer graphics," *ACM Transactions on Graphics (TOG)*, vol. 1, no. 1, pp. 7–24, 1982.

[16] R. Fernando *et al.*, *GPU gems: programming techniques, tips, and tricks for real-time graphics*. Addison-Wesley Reading, 2004, vol. 590.

[17] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 420–427, 2016.

[18] A. S. Periyasamy, M. Schwarz, and S. Behnke, "Refining 6D object pose predictions using abstract render-and-compare," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2019.

[19] A. Carlson, K. A. Skinner, R. Vasudevan, and M. Johnson-Roberson, "Modeling camera effects to improve visual learning from synthetic data," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[20] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols," *IEEE Robotics and Automation Magazine*, pp. 36–52, 2015.

[21] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "ObjectNet3D: A large scale database for 3d object recognition," in *European Conference on Computer Vision*, Springer, 2016, pp. 160–176.

[22] V. Nekrasov, C. Shen, and I. Reid, "Light-weight refinenet for real-time semantic segmentation," in *British Machine Vision Conference (BMVC)*, 2018.

[23] D. Pavlichenko, D. Rodriguez, C. Lenz, M. Schwarz, and S. Behnke, "Autonomous bimanual functional regrasping of novel object class instances," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2019.