Closing the Reality Gap

Nawal Hafez
Chair for Data Processing, Technical University of Munich
nawal.hafez@tum.de

***Abstract*—Large annotated datasets are a necessity nowadays but also very expensive to generate in real settings. Simulations are unable to represent the real world which is why different methods are utilized to account for these deficiencies. We choose three methods, photo-realistic rendering, domain randomization and adaptation, to inspect the effects on machine learning methods such as instance segmentation and pose estimation methods.**

***Keywords*—Synthetic Data Generation, Rendering, Object Detection.**

I. Introduction

Object detection and pose estimation are relevant in various fields nowadays, such as augmented reality, assembly lines, surveillance applications, driver assistance systems and robotics. Learning based methods have been shown effective at generalization [1]. Some of these methods require big training sets for the training process.

One drawback when it comes to training is the data generation. Often big datasets are required that are versatile and different enough to enable generalization, such as ImageNet [2], Pascal VOC [3] and LINEMOD Dataset [4]. These datasets are very hard to create, especially if not only a bounding box of an object is required but a pixel-wise segmentation or a 6D-pose. Generating large amounts of data synthetically is beneficial because the annotation can be automated and created scenes with different lighting and other conditions can be created. However, most synthetic data is too simplistic to achieve good results.

There are different methods to bridge the gap to real data, such as photo-realistic rendering, domain adaptation and randomization. Photo-realistic rendering tries to capture the physics of the world and should hence be able to render realistic images. Domain adaptation uses a generative adversarial neural network that learns to transform images created in a simulation to a more realistic version of it while domain randomization applies different types of noises or objects to an image to emulate the variability of the real world.

Throughout this work we will use these approaches to train neural networks for object detection, segmentation and pose estimation. We test the performance of the training on real data and compare different methods of data generation and their effect on the results. We also combine approaches to create the training datasets.

Methods for generating realistic synthetic images are presented in section II. We evaluate these methods based on existing object detection pipelines in section III. The main implementation details feature in section IV. Experiments are explained in V.

II. Related Work

A. Object Recognition and Segmentation

Learning-based approaches for object detection and grasping have become very successful and in a lot of cases eclipsed more traditional approaches in efficiency and accuracy rates [5]. Some of the most popular object detection methods include YOLO [6] which provides bounding boxes of objects in an image as well as Mask R-CNN [7], an instance segmentation method.
Mask R-CNN is the model we train for evaluating the data generation methods. In contrast to older instance segmentation methods, such as DeepMask [8], the segmentation and classification are processed in parallel which is faster and more flexible. Therefore, Mask R-CNN is an *instance-first* strategy and not a *segmentation-first* one. Mask R-CNN is an extension of Fast-RCNN. It has the same first stage as Fast-RCNN, namely a Region Proposal Network (RPN). The second stage is used to predict classes and boxes. Mask R-CNN has a third output, a mask, for each region of interest (RoI). [7]

These approaches require large annotated datasets for training. Generating all this data in real settings is usually time consuming and not always possible. Therefore, synthetic data is used and different methods are applied to close the gap to real data. The main approaches are: **Domain Randomization, Domain Adaptation**, and **Photo-realistic Rendering**.

B. Domain Randomization (DR)

The idea is to make the network to learn the most important feature that enables the object detection because the training data is versatile enough that it can generalize to the real world. Aspects that can be randomized include [9]–[11] :

- changing backgrounds, thereby creating new scenes
- adding a random amount of different geometric shapes to the scene
- sampling poses of the targeted object
- varying the object's texture
- different lighting scenarios
- type of noise and its amount added to the image

When training for grasping some other randomizations are added, such as random forces, or time [12].

C. Domain Adaptation (DA)

In DA images are transformed from one domain to another. DA is based on *Generative Adverserial Networks*. These are networks that are split into two parts, hence the term adverserial. One part of the network, the generator, tries to imitate as well as possible and the other part, the discriminator,

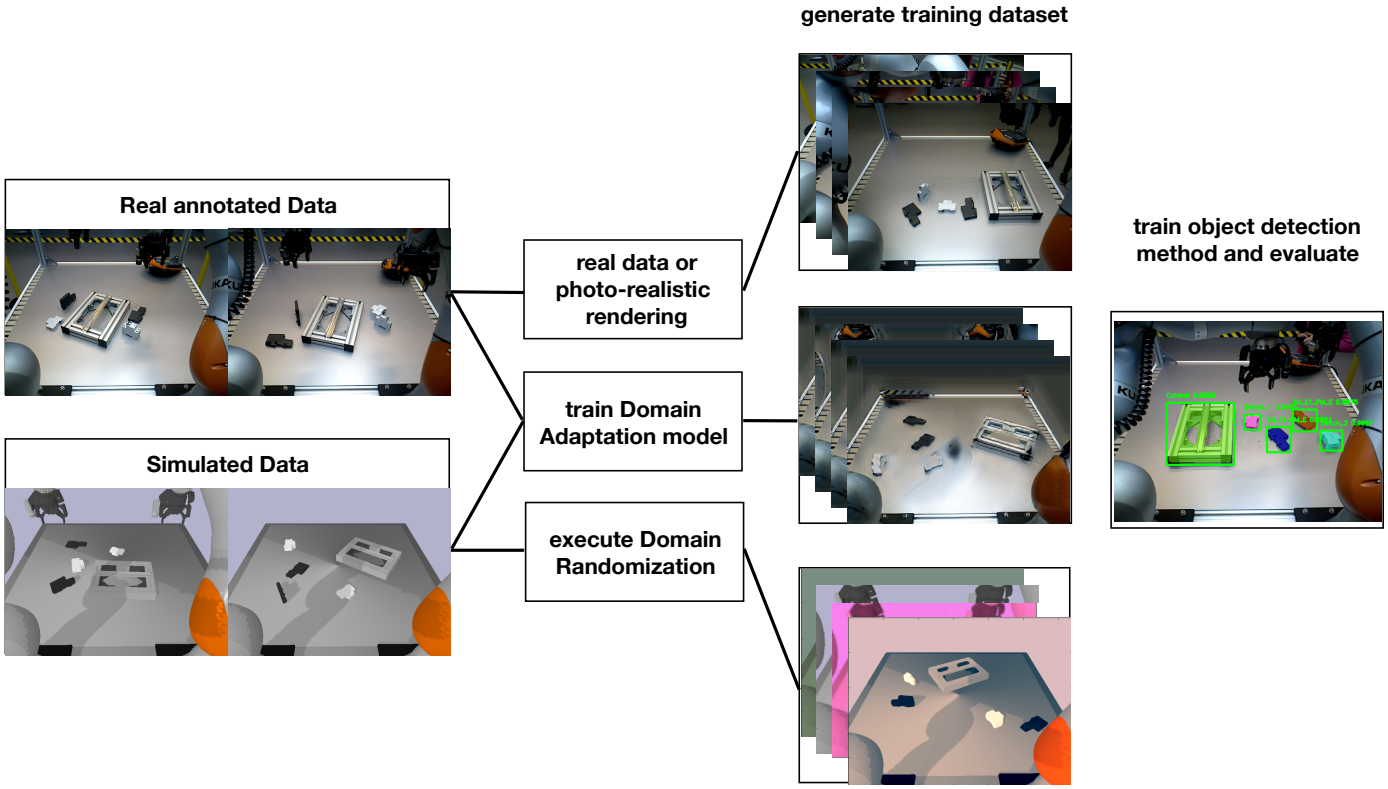


Fig. 1: Our Approach relies on generating data using three methods. We use these datasets to train a deep learning method and compare the performance on a real dataset.

tries to determine if it is a fake or not. [13]
DA relies on GANs to recreate the target domain as well as possible. The fact that the models are independent of a specific task and can be used for any domain adaptation is beneficial. The target domain either has paired or unpaired images [14]. Older methods [15] use feature vectors as input instead of synthetic images in the case of GANs. CycleGAN utilizes unpaired images with the idea of minimizing adversarial loss. The most important difference is the cycle consistency which should ensure that if the image is transferable from domain A to B then domain A can be reconstructed again [16]. It's been shown that Conditional Adversarial Network achieve the best results [17], [18]. The difference in conditional and regular GANs is the fact that the network does not only learn the mapping between two domains but the loss function that trains the mapping as well. In case of pix2pix, which is an exemplary approach that uses conditional GANs, the loss is conditioned on the input image. That means the generator and discriminator both observe the input image [17].
One issue with training conditional GANs is the huge amounts of labeled data needed. Recent research has shown that using self- and semi-supervised learning can lead to using only 20% of the data to achieve the same results [19].

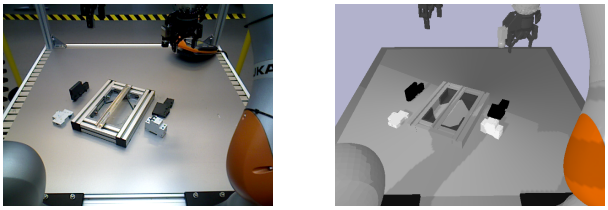
D. Photo Realistic Rendering

Opensource physics simulators or common renderers fail to recreate the complexity of the real world. Creating a realistic simulation is time consuming and not realistic if you wish to recreate many different scenes. To counter that high quality renderers can be used, according to [11], to at least take accurate pictures of the object, such as Mitsuba[1]. A realistic simulation pipeline that is also often used is Blender [2]. Photo-realistic rendering has been shown to be effective in simple environments and when in combination with real data [20]. It is however always imperfect. In general it has mixed results [21]–[24]. Some researchers suggest ensuring that the physics constraints are met when they place objects in a simulated scene as well [9].

III. Methodology

The main idea is to generate data using the three different approaches: **Photo-realistic Rendering**, **Domain Randomization**, **Domain Adaptation**. After generating the data we train object recognition methods to evaluate the effectiveness

[1] http://www.mitsuba-renderer.org
[2] https://www.blender.org



(a) Real image scene (b) Simulated scene

Fig. 2: Real annotated data

of each approach to get the best results. We also combine the methods to see if it can improve performance. An overview of the approach is visualized in Fig. 1.

A. Data Generation

1) Photo-realistic Rendering: It is possible to use any of the methods proposed in section II. However, creating a simulation or using Mitsuba is very time consuming which is why we use annotated real images that should present the optimal case of accurately presenting the real world. The drawback of this method is the limited amount of annotated data available.

Using more realistic CAD models can be helpful in improving rendered images. Scanned objects with the EinScan-Pro Scanner[3] are used to render scenes and images of the real scene are used as background. We replace our rendered object in the scene, see Fig. 7b and thereby do not need to annotate any real data.

2) Domain Randomization: We render the scene using a simple renderer. The images are then augmented. The most relevant augmentations are:

- gaussian blur
- additive gaussian noise
- median blur
- adding/removing pixels
- average blur
- sharpening kernel
- contrast normalization
- brightness manipulation

The augmentation used on every image is added randomly during or before training. That depends on the object recognition methods. Some methods already support augmentation during training. We also change the background for a subset of the data. The new background images are from the VOC2012 dataset [25]. For the simulation we use the scanned models, see Fig. 7a, as well as simple models without texture, see Fig. 2b.

3) Domain Adaptation: We use two DA methods to train models that transform the simulated images into the target domain, pix2pix and CycleGAN. Both methods are trained using a real annotated dataset. After training the model we generate images by transforming real images into the target domain and use these images to train the instance segmentation method.

4) Combining Methods: Combining these data generation methods is another relevant aspect for this work. Datasets are generated containing two methods each and the evaluation is done accordingly..

[3]https://www.einscan.com/handheld-scanner/einscan-pro/

B. Evaluation

For the evaluation we use a subset of a real annotated dataset that was not used at any stage of the training. Confusion matrices as well as mask overlapping errors and precision-recall metrics are the main analysis tools for the effectiveness of each approach. We set a threshold for the minimum overlap and calculate a confusion matrix to determine the success of training with these methods.

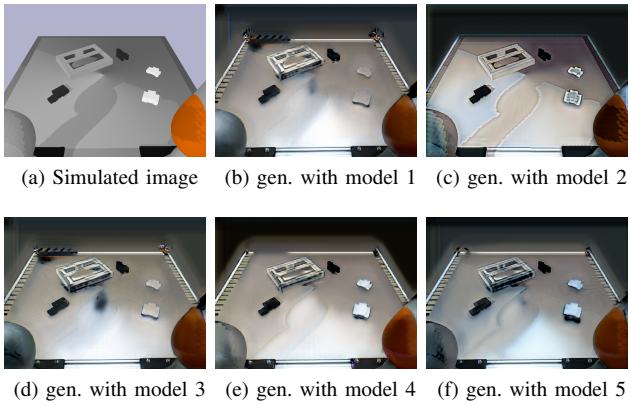
IV. Implementation Details

A. Training DA

130 annotated images are available. 104 of these images are used to train and validate pix2pix and CycleGAN. We use a Pytorch implementation for both methods[4].

To train pix2pix we require a real image and the ground truth poses of the objects to be able to render the objects as in the scene. Our objects include a rail, a black object, the TM and two kinds of switches. Fig. 2 contains an example of a real world image and the simulated image. We use these pairs to train pix2pix. Even if CycleGAN does not require pairs we use the pairs because they are already available to train the model.

After training pix2pix and Cycle_gan models we choose the best five models based on the loss recorded and use these to generate datasets to train the Mask R-CNN[5] models. We choose the 5 with the smallest losses for each model and generate new datasets. We generate simulated datasets with 400 random images each and then use the model to transfer it to the real domain. Examples of inferred images for the CycleGAN and pix2pix model are visible in Fig. 3 and Fig. 4.



(a) Simulated image (b) gen. with model 1 (c) gen. with model 2

(d) gen. with model 3 (e) gen. with model 4 (f) gen. with model 5

Fig. 3: A simulated image and then its transformation into the target domain using 5 different CycleGAN models

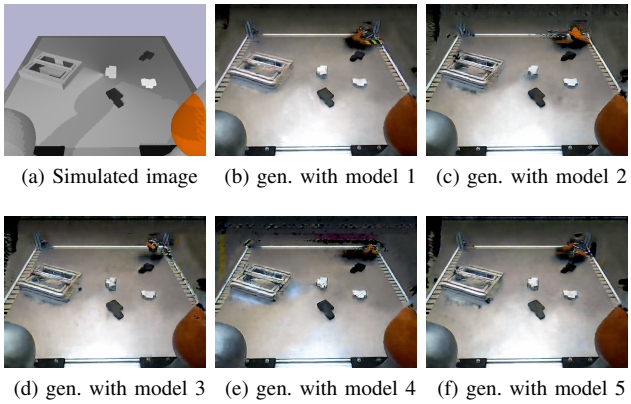
B. Simulation Details

We use pybullet to create the simulated scenes [6]. Each image contains a rail instance and an instance of each switch as well

[4]https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

[5]https://github.com/matterport/Mask_RCNN

[6]https://github.com/bulletphysics/bullet3



(a) Simulated image (b) gen. with model 1 (c) gen. with model 2

(d) gen. with model 3 (e) gen. with model 4 (f) gen. with model 5

Fig. 4: A simulated image and then its transformation into the target domain using 5 different Pix2pix models

as the black TM. For DA we use these randomly generated images to infer new images from the new domain. For DA we augment the images using the imgaug library [7].

V. Experiments

An overview of all the experiments is listed in TABLE I. It includes the information about the method used to generate the data, as well as the precision, recall recorded on the test data. In the case of multiple methods for one experiments it is also clear how the data is divided.

Exp. #	methods	#images	$precision[\%]$	$recall[\%]$
1	CycleGAN 320	400	77.6	83.9
2	CycleGAN 200	400	63.0	70.2
3	CycleGAN 340	400	70.2	79.8
4	CycleGAN 260	400	62.0	79.0
5	pix2pix 300	400	87.2	76.6
6	pix2pix 180	400	86.8	79.8
7	pix2pix 260	400	75.4	76.6
8	real images	105	98.2	91.1
9	scanned models with real cell backgrounds	300	93.5	81.5
10	DR	400	84.7	84.7
11	DR with scanned CAD models	300	74.7	87.9
12	real + simulated data without DR	105 + 200	100.0	94.4
13	real images + simulated data with DR	105 + 200	100.0	95.2
14	real images + CycleGAN	105 + 200	100.0	90.3
15	DR + CycleGAN	200 + 200	82.8	85.5

TABLE I: Overview of experiments

A. Domain Adaptation Experiments

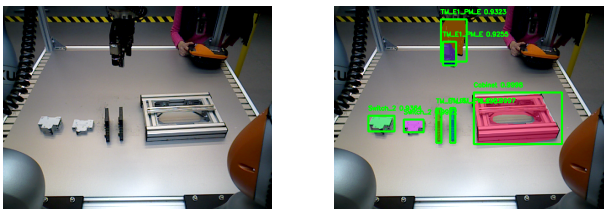
For our DA experiments we train a CycleGAN and pix2pix models. These are the experiments 1 through 7. We train Mask R-CNN with four CycleGAN models. Looking at the results it is clear that the precision is better than recall for each experiment. We will look at the first experiment closely with the help of the confusion matrix displayed in TABLE II.

[7]https://github.com/aleju/imgaug

The confusion matrices for experiment 2-4 are similar. It is clear that the model struggles in detecting the switches which is why it has a high precision but lower recall. Due to the rail's unique shape and size there does not seem to be any issues with the detection. The TM has a very high recall but low precision which is due to the fact that a lot of background objects are predicted as TMs. An image from the validation dataset, Fig. 5a, is provided to the trained network and the segmented output is visualized in Fig. 5b. You can see when the model falsely detects the camera on the robot as a TM. This happens often and can be explained by the fact that the model does not transform the robot hands precisely.

	gt_rail	gt_Switch	gt_background	gt_TM
predicted_rail	24	0	3	0
predicted_switch	0	37	7	3
predicted_background	2	10	0	4
predicted_TM	0	1	16	43

TABLE II: CycleGAN epoch 320 confusion matrix



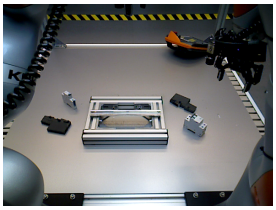
(a) Input test image (b) Segmented Image

Fig. 5: Model was trained with data generated by a CycleGAN model

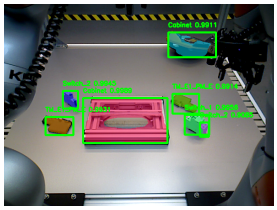
As for the pix2pix experiments one thing that stands out is that pix2pix has better precision than CycleGAN models. Based on the example in Fig. 6 pix2pix struggles with the controller that is sometimes in the background because it is not in the simulation. Therefore, it is not surprising that even with the high precision the mistakes in detection are related to that controller. The controller is often mistaken as TM or a cabinet.
Potential ways to improve the training would be to include random backgrounds in training and see if that helps improve precision. It could be also beneficial to train CycleGAN or pix2pix with more data. That could help improve the quality of the domain adaptation.

B. Domain Randomization Experiments

For our DR experiments use the imgaug library to augment the images during training. We use pybullet to render images. In experiment 11 we use untextured models and add a color for the whole mesh in the URDF file. In this case we achieve the same recall as the best DA model and a far better precision. The main struggle in this case is detecting switches, see TABLE III. It is clear that the depiction of the switches in

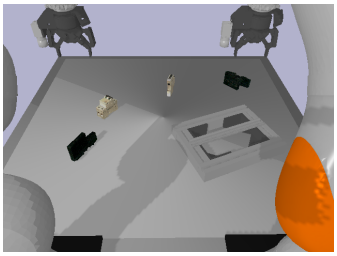


(a) Input test image

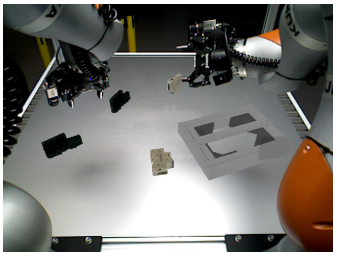


(b) Segmented Image

Fig. 6: Model was trained with data generated by a pix2pix model



(a) Scanned models placed in the simulation environment



(b) Scanned models with random real background of the scene

simulation does not allow for easy detection in real settings. The false detection of TMs is usually the camera attached to the robot hand. To improve training we can potentially use scanned models to see if that improves performance. Another possibility is changing trying to model the camera more realistically than currently in the simulation.

In experiment 12 we use the textured scanned models. Images with textured models are in Fig. 7a. We could not scan the rail easily because it has a reflective surface which is problematic with the scanner we used. It is clear that the recall for switches improves tremendously. However, the precision for TMs decreases. That can be explained by looking at simulated images. When we add the TM texture, it is possible that the network does not need to learn the exact shape of the object because the robot hands are lighter in the simulation. Therefore, anything with similar size like parts of the robot hands get recognized as the TM. When we do not use scanned models, the robot hands and the TM have the same colour which would explain why it forces the network to learn the shape of the TM better.

	gt_rail	gt_Switch	gt_background	gt_TM
predicted_rail	24	0	4	0
predicted_switch	0	33	1	0
predicted_background	2	13	0	2
predicted_TM	0	2	12	48

TABLE III: confusion matrix for training done with DR

	gt_rail	gt_Switch	gt_background	gt_TM
predicted_rail	23	0	1	0
predicted_switch	0	41	5	0
predicted_background	3	5	0	5
predicted_TM	0	2	29	45

TABLE IV: confusion matrix for training done with DR + scanned models

C. Photo-Realistic Rendering Experiments

For this part two experiments are relevant, 8 and 9. Even though we use as little as 105 images we achieve the best precision and recall in comparison to any data generation relying on one method for data generation. Looking closer at the results, it is clear that this model detects and TMs and rails perfectly but the only mistakes are because of the switches.

As for our second experiment to try and recreate realistic images by using scanned models with realistic backgrounds, see Fig. 7b and experiment 9. In this case we improve the precision in comparison to any stand alone method but the recall isn't as high as using real images.

D. Combinations

There are a few things we notice when combining multiple data generation methods. Combining methods improves precision drastically. It is also notable that when we combine real images with simulated images augmenting the data we achieve the best results (exp. 12 and 13). The difference in performance when we augment images or not would indicate that realistic images contain enough differences to make the augmentation not essential. The confusion matrix for experiment 13 looks as follows, TABLE V.

It is also clear that combining DR and real data is the only experiment that does improve recall and precision in comparison to just training with real images.

	gt_rail	gt_Switch	gt_background	gt_TM
predicted_rail	26	0	0	0
predicted_switch	0	44	0	0
predicted_background	1	4	0	1
predicted_TM	0	0	0	49

TABLE V: confusion matrix for training done with DR and real data

VI. Discussions

In conclusion, with this amount of real data it does not make sense to use DA. Training an instance segmentation method through DA generated data performed the worst out of the three possibilities even when combined with the other methods. For future work it could be interesting to train pix2pix to do the segmentation directly instead of using it as a way to generate data for training.

The performance of DR with a very simple simulation is better than DA and does not require any real annotated data. It achieves a recall of 85 %. In case real annotated data is

available it makes sense to combine it with a DR method because it achieves the best performance.

VII. Conclusion

Using learning based methods for object recognition is a widely researched topic. However, it is associated with generating large amounts of data. The best results are typically achieved with real data. Annotating real data is time-consuming and complicated. We inspect different methods to generate data and then train an instance segmentation method. These methods include photo-realistic rendering, domain adaptation and domain randomization. We also combine these methods.

In conclusion, we recommend using DR methods in combination with small amounts of real data to achieve the best results. For very small amounts of data the DA methods we used did not perform well. Using the same data to train directly did achieve better results.

References

[1] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes," in *Asian Conference on Computer Vision (ACCV)*, 2012, pp. 548–562.

[2] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[3] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision (IJCV)*, pp. 98–136, 2015.

[4] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, e. K. M. Navab, Nassir", Y. Matsushita, J. M. Rehg, and Z. Hu, "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes," in *Asian Conference on Computer Vision (ACCV)*, 2013, pp. 548–562.

[5] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes," in *International Conference on Computer Vision (ICCV)*, 2011, pp. 858–865.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[8] P. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to Segment Object Candidates," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1990–1998.

[9] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," *arXiv preprint*, 2018.

[10] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 969–977.

[11] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[12] 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," in *arXiv preprint*, 2018.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[14] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 95–104.

[15] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from Simulated and Unsupervised Images through Adversarial Training," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2107–2116.

[16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2223–2232.

[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1125–1134.

[18] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis with Auxiliary Classifier GANs," in *International Conference on Machine Learning (ICML)*, 2017, pp. 2642–2651.

[19] M. Lučić, M. Tschannen, M. Ritter, X. Zhai, O. Bachem, and S. Gelly, "High-Fidelity Image Generation With Fewer Labels," in *International Conference on Machine Learning (ICML)*, 2019, pp. 4183–4192.

[20] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D Orientation Learning for 6D Object Detection from RGB Images," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 699–715.

[21] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh, "How useful is photo-realistic rendering for visual learning?" in *European Conference on Computer Vision (ECCV)*, 2016, pp. 202–217.

[22] C. Mitash, K. E. Bekris, and A. Boularias, "A Self-supervised Learning System for Object Detection Using Physics Simulation and Multi-view Pose Estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 545–551.

[23] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for Data: Ground Truth from Computer Games," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 102–118.

[24] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2686–2694.

[25] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision (IJCV)*, pp. 303–338, 2010.

[26] B. Tekin, S. N. Sinha, and P. Fua, "Real-Time Seamless Single Shot 6D Object Pose Prediction," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 292–301, 2018.