

Sim2Real image translation to improve a synthetic dataset for a bin picking task

Diana Duplevska*, Maksims Ivanovs[†], Janis Arents[‡] and Roberts Kadikis[§]

Institute of Electronics and Computer Science (EDI)

Riga, LV-1006, Latvia

Email: *diana.duplevska@edi.lv, [†]maksims.ivanovs@edi.lv, [‡]janis.arents@edi.lv, [§]roberts.kadikis@edi.lv

Abstract—The use of synthetic data is a promising solution to the problem of the availability of real data needed for the development of robotic systems. However, the precision of the systems trained on synthetic data tends to decrease when they are deployed in the real-life scenarios, which happens due to the disparities between the artificial data and the real world. Therefore, efficient methods for Sim2Real translation are much needed for further progress in robotics. In our study, we use Generative Adversarial Networks (GANs) to generate more photorealistic data from the synthetic data created for training deep neural networks to handle a bin picking task. As a result, object detectors trained on the images improved with the GANs with the optimal approach to image translation demonstrate better performance than their counterparts trained on the original synthetic data.

Index Terms—Sim2Real, image translation, synthetic data, CycleGAN, bin picking

I. INTRODUCTION

Robotic systems in modern industries face rising demands on speed, precision, size, adaptability, interoperability, and cognitive features. Complying with such demands requires the ability to swiftly improve and integrate various custom-made modules of these systems [1], [2]. Today’s AI-based perception modules require a large amount of training data; moreover, changes in the environment of deployed systems may lead to a domain shift, when the data used for training is no longer a good representation of the real task the robot needs to perform. A promising solution is the use of synthetic data and simulations, which can facilitate the development of smart robotic systems in various ways and at different stages [3]. Acceleration of the design cycle, generation of large amounts of the data at a low cost, and enabling safe and fully controlled testing environments are only some of the opportunities provided by the synthetic data [4]. While the generation of synthetic data is a promising direction of research and an enabling technology for future robots, several unresolved challenges currently reduce the precision of the robotic systems when deployed in real-life scenarios. In particular, the differences between a simulation and the real world typically are the main cause for the decrease in precision: artificial lighting, sensor data recreation, virtual representations of objects, physics simulations and other countless

aspects of a real environment that cannot yet be fully recreated in virtual worlds are all contributing to the issue of transferring learned models from simulation to reality.

Data collection and processing plays an important role in machine learning tasks including smart robot control, as nowadays machine learning-based methods are widely used to solve challenging tasks with the use of robots [5]. According to [8], on average, more than 80% of time spent on AI projects is dedicated the collection and processing of the data. For robots, the data is an important element of their design, as it makes it possible to anticipate events and prepare for them in advance, therefore coping with harsh, dynamic conditions and unforeseen situations. In essence, the data supports the perception part that includes acquiring the information about an unstructured environment, analysing this information, extracting features, interpreting them, etc. This information further on is taken into account in adaptive path planning and motion control steps to cope with highly unstructured environments [3] [7].

On the one hand, coping with unforeseen situations requires special algorithms that are able to generalise over a variety of scenarios. On the other hand, the data is the key factor, and the success rate depends on whether the robot has had a similar experience in the training process. The training set, however, typically consists of annotated datasets; manual labelling of such datasets is time-consuming manual labour, and thus some corner cases might be missing from the dataset due to a complicated training data acquisition process. Data synthesis, however, facilitates this process and simplifies the use of modern computer vision methods in industry.

Image synthesis or rendering is the process of generating digital images from virtual scenes. The photorealism of rendered images, videos, and computer games keeps increasing, and the tools for creating virtual environments with included physics simulation also become more user-friendly and affordable: for example, such tools as Blender, Unity, and Unreal Engine can be used free of charge. Therefore, AI and computer vision research community increasingly use such tools to generate data or train systems in virtual environments directly; however, the precision decrease can be observed when models are trained on purely synthetic data when compared to real data [9]. In smart robotic systems, this concern can be highly connected to an issue that is commonly referred to as “reality gap” [10]: even though reliable operation can be achieved

This research has received funding from the ECSEL Joint Undertaking under grant agreement 101007311 (IMOCO4.E). The Joint Undertaking receives support from the European Union’s Horizon 2020 research and innovation programme.

in simulations, trained models can perform unreliably when transferred to a real environment. As real-world maintenance times can be very short, a more reliable sim-to-real translation is required.

In the present paper, we are concerned with improving the photorealism of synthetic images by using Generative Adversarial Networks [17] for image-to-image translation in order to improve the precision of the object detection task. The respective experiments performed in our study article are a first step in the overall development of a bin-picking pipeline, as their goal is to detect objects in 2D images that are the most promising to end up in a successful grasp. The proposed approach can be also utilised for further steps such as instance segmentation and grasp pose estimation in 3D world. Furthermore, the generated data contains sufficient 3D information to be used by different approaches, for example, to directly perform 6D object pose estimation [11]. However, a detailed analysis of how our proposed approach contributes to these steps is beyond the scope of this paper; therefore, we envisage these as future work. Taking into account the above-mentioned considerations, the rest of the paper is structured as follows. In Section II, we describe related work; Section III is concerned with the description of the data used in the present study; in Section IV, we describe the improvements of CycleGAN [TODO: add citation]; in Section V, we discuss the visual quality of resulting CycleGAN images; Section VI describes the methodology of object detection experiments; Section VII is concerned with the results; finally, Section VIII offers conclusions and directions for future work.

II. RELATED WORK

Reality gap is a topical problem for machine learning. To reduce or almost completely bridge the gap, one can apply methods of adapting the synthetic and real environments by transferring the appearance of one environment to another. Several domain transfer approaches have already proven themselves, namely, pixel-level [12] [13] and feature-level [14]. Feature-level domain adaptation focuses on learning domain-invariant features between source and target domains or by learning domain-invariant features which are represented by a convolutional neural network (CNN) [15] [16]. Pixel-level domain adaptation focuses on image stylizing - images from a source domain are made to look like images from a target domain. This domain adaptation is commonly based on image-conditioned generative adversarial networks (GANs) [17]. These methods can be used to solve the simulation-to-reality domain shift problem for robotic manipulation. To address the reality gap in robotics, some projects use machine learning-based domain adaptation [18] or randomization of simulated environments [19]. With domain randomization, a deep neural network is trained with randomized simulation parameters and scene configurations, which produces differences in visual appearance.

However, these approaches may change the image, including removing the information that is necessary for a given task. This is especially critical for tasks with robot manipulation or

object detection, because objects or their features may be completely or partially erased. For this reason, it is very common to combine domain adaptation with additional techniques such as a semantic map of the simulated image [20], which saves the semantics of the task. Also, additional loss can be introduced. Reinforcement learning task loss [21] enforces consistency of task policy Q-values between the original and transferred images to preserve information important to a given task. RL-CycleGAN is trained jointly with the RL model and requires task-specific real-world episodes. Another approach is to add an object detector with perception consistency loss, which penalizes the generator for discrepancies in object detection between translations [22]. RetinaGAN works for supervised and imitation learning, and it uses object detection as an independent task for object-level visual domain differences.

In the present work, we used pixel-level domain adaptation, which is based on GAN. We transferred the domain from synthetic domain to real (Sim2Real) and checked the translation using the object detector; our objective was that the objects should be preserved and remain recognizable as much as possible.

III. DATA

We used several datasets in the present study, which are described in the following.

A. Synthetic dataset

For the experiments in this work, a synthetic dataset as described in [6] was generated, consisting of 8800 high-resolution photo-realistic scenes as depicted in Fig. 1. For each scene, an initially empty box was filled with randomly dropped bottles by using Blender physics simulation engine, which allowed to realistically generate random configurations of the bottles in the container. After filling the box, the light intensity of four different light sources was varied and the scene itself was rendered from 16 different angles. For increased realism, a Blender shader nodes were used as well as realistic textures, reflections and indirect light bounces. Also for each scene an annotation file was generated that includes every single object in the scene, its rotation, coordinates, and the visibility percentage.



Fig. 1: Generated image

B. Real-world dataset

Real-world data were acquired by randomly distributing bottles in the container. For each of the acquired image, the positions of bottles were altered by emptying and refilling the box. The camera exposure time and intensity of lighting were systematically modified to acquire high diversity of different lighting conditions in the real-world dataset. In total, 2200 real images were acquired and manually labelled, out of which 1760 images were used for training Sim2Real and 440 images for testing.

C. CycleGAN datasets

We used CycleGAN to translate synthetic bottle images into the more realistic ones (Sim2Real transfer), thus creating several new datasets. We used previously described synthetic and real-world image datasets to train the CycleGAN translator. Original real-world photos of the size 528x342 pixels, whereas synthetic images are 1024x768; as the CycleGAN requires identical size images for training, the preprocessing of the images includes resizing of the images from both datasets to 256x256 resolution. The reduction of the resolution was also necessary due to the limited GPU resources available for training CycleGAN. The next important aspect was the disparity of the data: the real-world photos were taken directly from above the box, whereas the synthetic ones were also taken from above but at a slight angle. As the CycleGAN doesn't need paired data, it was not necessary to further process the data e.g. by segmenting or labelling it. Therefore, the only further change to the data was that we cropped the background on the synthetic data, because after resizing to 256x256 pixels it was necessary to make sure that the objects would be clearly visible. From the synthetic data set containing 8800 images, we took for training only synthetic data with good lighting to make all the features of the objects clearly visible to the neural network. Due to that, each dataset for training CycleGAN included 1760 images.



(a) Real image (b) Synthetic image

Fig. 2: Examples of images from real (a) and synthetic (b) dataset

IV. CYCLEGAN IMPROVEMENTS

To solve the “reality gap” problem, we used a Generative Adversarial Network (GAN) [17] to generate realistic data from our synthetic data by image-to-image translation. We chose to apply the Cycle-Consistent Adversarial Network

(CycleGAN) in our task because this approach is meant to find a mapping from domain X to domain Y without the images being paired. This means that the data does not have to be completely identical, making it easier to create datasets.

CycleGAN uses two mapping functions $G : X \rightarrow Y$ to translate images in domain X to domain Y and inverse mapping $F : Y \rightarrow X$ to translate images in domain Y to domain X . There are additionally two discriminator functions D_X and D_Y that are used to discriminate whether an image is in a respective domain. In our case, the X corresponds to synthetic data, and the Y corresponds to the real data.

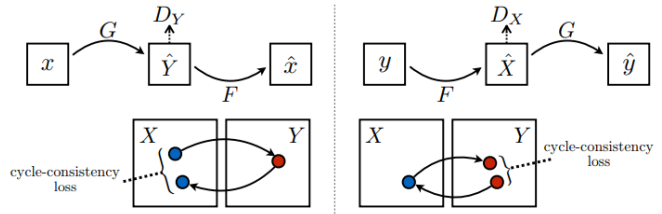


Fig. 3: CycleGAN data generation algorithm [29]

Each mapping function and its associated discriminator function has a generative adversarial loss. And an inverse mapping introduces a cycle consistency loss to push $F(G(X)) \approx X$ and otherwise $G(F(Y)) \approx Y$.

As a baseline, we used the CycleGAN code from Tensorflow [26] and trained it on our datasets. After initial tests with the baseline code, we made several improvements. The training process was the same for all tests: we ran each CycleGAN model for 20 epochs with the Adam optimizer using an initial learning rate of 0.0002, $\beta = 0.5$ and $\lambda = 10$. The weights were initialized with a Gaussian distribution with a mean 0 and a standard deviation of 0.02. For every epoch the dataset was shuffled, and the buffer size was set to 1000.

1) *Baseline - CycleGAN by TensorFlow:* For the initial training, we used a regular CycleGAN available in TensorFlow. The main difference between the original CycleGAN and the TensorFlow implementation of it was that the original CycleGAN paper uses a modified ResNet-based generator [29]. The TensorFlow implementation that we used is based on a modified U-Net generator for simplicity. The U-Net can be described as a convolutional autoencoder with skip connections. The encoder downscales the image using convolutional layers, and the decoder equivalently upscales the latent space back to the original dimensions. Every transposed convolutional layer in the decoder has a so-called skip connection. Skip connections are a way to help bypass the vanishing gradient problem by concatenating the output of a layer to multiple layers instead of only one.

We trained CycleGAN as implemented in TensorFlow to transfer the style of the real photos to synthetic images of bottles. As shown in Fig. 5, the resulting neural network correctly translates and draws the shape of the box. However, the bottles that are in the shadow on the synthetic image are partially erased after the transfer. In the top part of the resulting

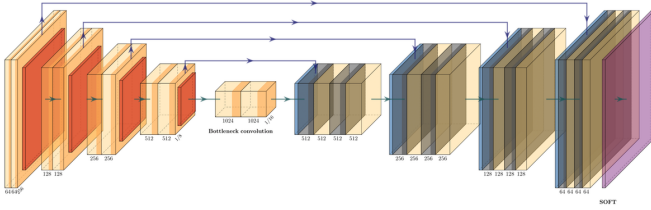


Fig. 4: U-Net network architecture

image, the neural network draws non-existent bottles. This is a problem for computer vision tasks and for this reason we could not use these images for testing with the object detector.



Fig. 5: Image before CycleGAN (input synthetic image) and after (generated image)

2) *Better CycleGAN*: To improve the quality of the generated images, we added several data preprocessing functions. In addition to image resizing, jittering, mirroring and data normalization, we added colour data augmentation: random contrast, brightness, hue and saturation. This was done to reduce overfitting. We also added a central crop while maintaining the image size of 256x256, which allowed the neural network to focus on the bottles rather than the background and box.

In order to let the generator train longer, we added Gaussian noise to the discriminator input, so it was more difficult for the discriminator to evaluate images. This method helps to avoid early overfitting of the discriminator. The overfitted discriminator evaluates even high-quality generated images as fake, which causes imbalance in the neural network.

3) *Checkerboard artifacts*: The results of improved CycleGAN can be used to train the object detector. However, there are still some observable image defects, eliminating which could improve the quality of the generated images. The most noticeable problem appears to be a small pixel grid in the image, which is known as checkerboard artifacts [24]. This effect occurs because of the transposed convolution [25] operations in a decoder. Therefore, to solve this problem, it was necessary to replace the transposed convolution operation with another: for example, resize the images and then add a common convolutional layer. We replaced the transposed convolution with a TensorFlow function `tf.keras.layers.UpSampling2D` to upscale the images; by using this resizing with convolution, we removed the checkerboard artifacts, yet the images became blurred. This is a more critical problem for computer vision

tasks, because the edges of objects are blurred, and their position is not clear. For this reason, we used the resized convolution on all layers except the last one, where we retained the transposed convolution. This method made the resulting images less blurry, and the checkerboard grid was hardly noticeable.

V. VISUAL QUALITY OF RESULTING CYCLEGAN IMAGES

We used Frechet Inception Distance (FID score) to evaluate image quality by comparing the original data and fake images. It is a performance metric that calculates the distance between the feature vectors of real images and the feature vectors of fake images that are generated by GAN generator; lower FID score means smaller difference between real and fake images. We used a TensorFlow implementation of the FID score to evaluate images [23]. We compared CycleGAN generated output images with real-world photos and with original synthetic images. This allowed us to find out how similar the fake images are to their original synthetic images and how much to the real-world photos (i.e., the target domain). As the result, synthetic images after Sim2Real transfer by CycleGAN should look more realistic. All obtained FID scores are shown in a table I.

We used 5000 images to measure the FID score. This amount of the data was obtained using augmentation (random horizontal and vertical flip, random rotation, random hue). We compared images that were generated by CycleGAN with original datasets (synthetic images and real photos), which were also used to train the neural network. "Baseline" dataset was created by original TensorFlow example of CycleGAN. "Augmented" dataset is the baseline code with data preprocessing. Then we added a noise layer in discriminator and got "Augmented noise" dataset. Next dataset, "Resized convolution", is acquired by replacing the generator's transposed convolution layers with upsampling and convolution layers. "Resized transpose" dataset has resized convolution on all layers except the last one. Both worst FID results are with "baseline" dataset. When comparing fake real images and real photos (FID A) the "resized transpose" dataset had the best result. However, when comparing the generated images with the original synthetic images (FID B), the "resized convolution" dataset had a better score.

Dataset	Baseline	Augmented	Augmented noise	Resized convolution.	Resized transpose
Images					
Fid A	230.28	171.41	137.27	141.03	112.26
Fid B	264.81	169.73	152.29	122.86	127.25

TABLE I: Evaluation for synthetic \rightarrow real. FID A evaluates distance between generated real images (fake real photos) and real photos, FID B is between generated real images and synthetic images.

For further testing, we selected two different types of CycleGAN (“Augmented noise” and “Resized transpose”) to compare the quality of the neural network on large images. These two types of CycleGANs have different types of image resizing in CNN part, as a result they affect the image in different ways. For example, on small input data image defects are invisible to the eye and FID score, but on large images there may be defects and artefacts that affect the image semantics. We applied these neural networks to original synthetic images with a resolution of 1024x768 and bright lighting. Thus, we got two more data sets: “Augmented noise 1024x768” and “Resized transpose 1024x768”. We cannot compare these data sets with real photos due to the different sizes and different semantics of the photos, so we only compared them with the original synthetic image to show how much the image quality has changed after cycleGAN.



Dataset	Augmented noise (1024x768)	Resized transpose (1024x768)
Images		
FID	58.46	69.24

TABLE II: Evaluation for synthetic \rightarrow real with resolution 1024x768

As it follows from the low FID, the quality of large images after Sim2Real transfer was better than of those from a 256x256 resolution dataset. On images of size 1024x768, the pixel grid was less noticeable and had a smaller impact on the FID score than blurred objects without pixel grid. As a result, the dataset with checkerboard artifacts showed the best results. However, these results are obtained by comparing only large images with excellent lighting, yet the neural network will be applied to synthetic images with different brightness parameters. These datasets will be used in the future for the object detector.

Images from next dataset had various lighting conditions, from very bright to dark, and the light source position was also changed. When applying Augmented noise CycleGAN to these images, the semantics do not change, and the only shortcoming is that there some artifacts that appear in the background. Howeverk, when Resized transpose CycleGAN is used on images with medium brightness, there appear artifacts in the form of pixels around objects (bottles and the box), which can interfere with the semantics of the image such as the shape of the objects. The dataset in question received a relatively high FID score, which allows us to conclude that the image quality in this particular case is worse, and the dataset might pose problems and result in decreased precision for the object detection task.




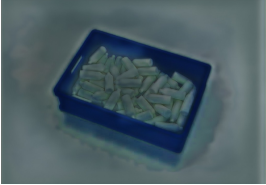
Dataset	Augmented noise (1024x768)	Resized transpose (1024x768)
Bright images		
Dark images		
FID	80.11	124.83

TABLE III: Evaluation for synthetic \rightarrow real with resolution 1024x768 and different brightness levels.

VI. OBJECT DETECTION EXPERIMENTS

We conducted object detection experiments using YOLOv5 object detector implemented in the Ultralytics library [27]. Experiments were conducted on each dataset with YOLO5 Small, Medium, and Extra Large (XLarge) models pretrained on COCO dataset [28]. As it follows from the name of the models, they differ in size, which is 14, 41, and 166 MB, respectively. During the training, 90 percent of the images in a respective dataset were used for training, and 10 percent of the images were used for validation. The training parameters of the models were as follows: image size 640x640 pixels, batch size 16, learning rate 0.01, momentum 0.937, weight decay 0.0005. Each model was trained for 300 epochs with early stopping after 100 epochs, which occurred if there was no validation loss improvement. After training, the checkpoint with the best performance on the validation set was tested on the set of real-world images consisting of 300 test images.

VII. RESULTS

We report our results in IV using the standard metrics for the task of object detection, namely, precision, recall, and mean average precision (mAP) for bounding boxes. mAP is calculated for an intersection over union (IoU) threshold of 0.5 and also averaged for $\text{IoU} \in [0.5 : 0.05 : 0.95]$.

As it can be seen, the models trained on the Augmented noise dataset consistently outperform both the models trained on the Original synthetic dataset and the models trained on the Resized transpose dataset in terms of precision, recall, and mAP for IoU with threshold of 0.5 and mAP averaged for $\text{IoU} \in [0.5 : 0.05 : 0.95]$ metrics. Furthermore, the performance of the models trained on the Resized transpose dataset is consistently worse than that of the models trained on the Original synthetic dataset with the sole exception being the better precision of the XLarge model. Interestingly enough, larger models do not show better performance than their smaller counterparts, which may indicate that their size was too large for a comparatively small size of the training datasets.

Model	Dataset	Precision	Recall	mAP (threshold 0.5)	mAP (avg for IoU ∈ [0.5 : 0.05 : 0.95])
YOLOv5 Small	Original	0.689	0.894	0.742	0.447
	synthetic	0.615	0.838	0.632	0.245
	Resized transpose Augmented noise	0.731	0.905	0.777	0.48
YOLOv5 Medium	Original	0.693	0.802	0.72	0.42
	synthetic	0.581	0.782	0.589	0.204
	Resized transpose Augmented noise	0.712	0.91	0.753	0.455
YOLOv5 XLarge	Original	0.718	0.785	0.75	0.413
	synthetic	0.683	0.867	0.715	0.339
	Resized transpose Augmented noise	0.721	0.872	0.761	0.461

TABLE IV: Results of the object detection experiments.

VIII. CONCLUSIONS

In this study, we had the goal of improving Sim2Real translation methods by increasing the photorealism of synthetic images of plastic bottles in a box designated for a bin picking task. Initially we employed the original implementation of CycleGAN in TensorFlow for translating images; subsequently, we employed the improved versions of CycleGAN, which we designated as “Resized transpose” and “Augmented noise” CycleGANs. To objectively assess the results of the Sim2Real translation, we train YOLOv5 object detectors of three various sizes on these datasets and test them on the real-world images. While the object detectors trained on the Resized transpose dataset perform worse than those trained on the original synthetic data, their counterparts trained on the Augmented noise dataset outperform the models trained on the original synthetic data across all the metrics of interest, namely, precision, recall, and mAP. These results demonstrate that CycleGAN can be successfully used for Sim2Real translation for the datasets for a bin picking task. Even though the performed object detection experiments are only one step of the bin-picking pipeline, the proposed approach is applicable for the proceeding steps of the bin-picking pipeline which is envisaged as our future work. Furthermore, we also intend to further improve the photorealism of the translated images.

ACKNOWLEDGEMENTS

We would like to thank Andis Bizuns for his kind help with organising the data.

REFERENCES

- [1] Čech, M., Beltman, A. J., Ozols, K. (2019, September). I-mech-smart system integration for mechatronic applications. In 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 843-850). IEEE.
- [2] Čech, M., Beltman, A. J., Ozols, K. (2021). Pushing Mechatronic Applications to the Limits via Smart Motion Control. Applied Sciences, 11(18), 8337.
- [3] Arents, J. and Greitans, M., 2022. Smart Industrial Robot Control Trends, Challenges and Opportunities within Manufacturing. Applied Sciences, 12(2), p.937.

- [4] Choi, H.; Crump, C.; Duriez, C.; Elmquist, A.; Hager, G.; Han, D.; Hearl, F.; Hodgins, J.; Jain, A.; Leve, F.; et al. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proc. Natl. Acad. Sci. USA* **2021**, *118*, e1907856118. doi:<https://doi.org/10.1073/pnas.1907856118>.
- [5] P. Racinskis, J. Arents, and M. Greitans, “A motion capture and imitation learning based approach to Robot Control,” *Applied Sciences*, vol. 12, no. 14, p. 7186, 2022.
- [6] Arents, J., Lesser, B., Bizuns, A., Kadikis, R., Buls, E., Greitans, M. (2022). Synthetic Data of Randomly Piled, Similar Objects for Deep Learning-Based Object Detection. In: *Image Analysis and Processing – ICIAP 2022. Lecture Notes in Computer Science*, vol 13232. Springer, Cham. https://doi.org/10.1007/978-3-031-06430-2_59
- [7] Torres, P.; Arents, J.; Marques, H.; Marques, P. Bin-Picking Solution for Randomly Placed Automotive Connectors Based on Machine Learning Techniques. *Electronics* **2022**, *11*, 476. <https://doi.org/10.3390/electronics11030476>
- [8] Cognilytica. Data Engineering, Preparation, and Labeling for AI **2019**.
- [9] Anderson, Jason. “Methods and Applications of Synthetic Data Generation.” (2021).
- [10] Jakobi, N.; Husb, P.; Harvey, I. Noise and The Reality Gap: The Use of Simulation in Evolutionary Robotics. In *Proceedings of the European Conference on Artificial Life, Lausanne, Switzerland, 13–17 September 1999*. doi:https://doi.org/10.1007/3-540-59496-5_337.
- [11] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, “A survey on learning-based robotic grasping,” *Current Robotics Reports*, vol. 1, no. 4, pp. 239–249, 2020.
- [12] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. “Unsupervised pixel-level domain adaptation with generative adversarial neural networks,”(2017) doi: <https://doi.org/10.48550/arXiv.1612.05424>.
- [13] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. “Learning from simulated and unsupervised images through adversarial training,”(2017) doi: <https://doi.org/10.48550/arXiv.1612.07828>.
- [14] B. Sun, J. Feng, and K. Saenko. “Return of frustratingly easy domain adaptation,” doi: <https://doi.org/10.48550/arXiv.1511.05547>.
- [15] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky.(2016). “Domain-adversarial training of neural networks,” doi: <https://doi.org/10.48550/arXiv.1505.07818>.
- [16] Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan.(2016). K.“Domain separation networks,” doi: <https://doi.org/10.48550/arXiv.1608.06019>.
- [17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. (2014). “Generative adversarial nets,” vol. 27, pp. 2672–2680.
- [18] Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Pieter Abbeel, Sergey Levine, Kate Saenko, Trevor Darrell. (2015). “Adapting Deep Visuomotor Representations with Weak Pairwise Constraints” doi: <https://doi.org/10.48550/arXiv.1511.07111>.
- [19] Fereshteh Sadeghi, Sergey Levine. (2016). “CAD2RL: Real Single-Image Flight without a Single Real Image” doi: <https://doi.org/10.48550/arXiv.1611.04201>.
- [20] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, Vincent Vanhoucke. (2017). “Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping” doi: <https://doi.org/10.48550/arXiv.1709.07857>.
- [21] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, Mohi Khansari. (2020). “RL-CycleGAN: Reinforcement Learning Aware Simulation-To-Real” doi: <https://doi.org/10.48550/arXiv.2006.09001>.
- [22] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, Yunfei Bai. (2020). “RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer” doi: <https://doi.org/10.48550/arXiv.2011.03148>.
- [23] Heusel et al. [Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. and Hochreiter, S. (2017). “Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*” (pp. 6626–6637).
- [24] Odena, et al. (2016). “Deconvolution and Checkerboard Artifacts”, *Distill*. doi:<http://doi.org/10.23915/distill.00003>.
- [25] Wenzhe Shi, Jose Caballero, Lucas Theis, Ferenc Huszar, Andrew Aitken, Alykhan Tejani, Johannes Totz, Christian Ledig, Zehan Wang. (2016). “Is the deconvolution layer the same as a convolutional layer?” doi:<https://doi.org/10.48550/arxiv.1609.07009>.

- [26] CycleGAN Tensorflow. Link: <https://www.tensorflow.org/tutorials/generative/cyclegan>
- [27] YOLOv5 implementation in Ultralytics library. Link: <https://github.com/ultralytics/yolov5>
- [28] Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft coco: Common objects in context." In European conference on computer vision, pp. 740-755. Springer, Cham, 2014.
- [29] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks" doi: <https://doi.org/10.48550/arXiv.1703.10593>.