Conditional patch-based domain randomization

Ani, Mohammad; Basevi, Hector; Leonardis, Ales

DOI:
10.1109/IROS47612.2022.9981381

Document Version
Peer reviewed version

Citation for published version (Harvard):
Ani, M, Basevi, H & Leonardis, A 2022, Conditional patch-based domain randomization: improving texture domain randomization using natural image patches. in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Kyoto, Japan, pp. 1979-1985, 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, 23/10/22. https://doi.org/10.1109/IROS47612.2022.9981381

Conditional Patch-Based Domain Randomization: Improving Texture Domain Randomization Using Natural Image Patches

Mohammad Ani[1], Hector Basevi[2], Aleš Leonardis[2]

***Abstract*— Using Domain Randomized synthetic data for training deep learning systems is a promising approach for addressing the data and the labeling requirements for supervised techniques to bridge the gap between simulation and the real world. We propose a novel approach for generating and applying class-specific Domain Randomization textures by using randomly cropped image patches from real-world data. In evaluation against the current Domain Randomization texture application techniques, our approach outperforms the highest performing technique by 4.94 AP and 6.71 AP when solving object detection and semantic segmentation tasks on the YCB-M [1] real-world robotics dataset. Our approach is a fast and inexpensive way of generating Domain Randomized textures while avoiding the need to handcraft texture distributions currently being used.**

I. INTRODUCTION

Data-driven deep learning approaches have been highly beneficial for a wide range of robotics tasks, particularly when using computer vision systems as part of the pipeline [2], [3], [4]. Typically, such systems rely on access to vast quantities of annotated data to train a model capable of understanding a scene [5], [6]. A significant hurdle with training such systems is access to high-quality labeled data which describes where and what each object of interest is for a scene. Synthetic data is increasingly being used as an alternative to gathering and annotating real-world data to train deep learning-based vision systems. In particular, Domain Randomization (DR) is a popular approach that has been shown to bridge the gap between simulation and the real world in various robotics tasks using vision [7], [8]. The approach randomizes simulator parameters, such as the texture of an object of interest, their positions, or backgrounds, to generate highly varied scenes to train vision systems capable of transfer to the real world. In a typical DR approach, we must manually define distributions for simulator parameters to sample values from. For example, to generate textures, one would manually define distributions for simulator parameters to sample the colors or patterns to generate textures to apply to objects of interest. Suppose we wanted to generate DR textures for a cereal box. The most common approach is to sample a random color, sometimes patterns, and apply that to the object [7], [2], [4]. This approach does not account for visually relevant features that may be beneficial during the training process, such as text, barcodes, or nutritional information on the cereal box.

[1] This work was done while at the School of Computer Science, University of Birmingham, Birmingham, UK, mxa563@alumni.bham.ac.uk.
[2] School of Computer Science, University of Birmingham, Birmingham, UK {h.r.a.basevi, a.leonardis}@bham.ac.uk.

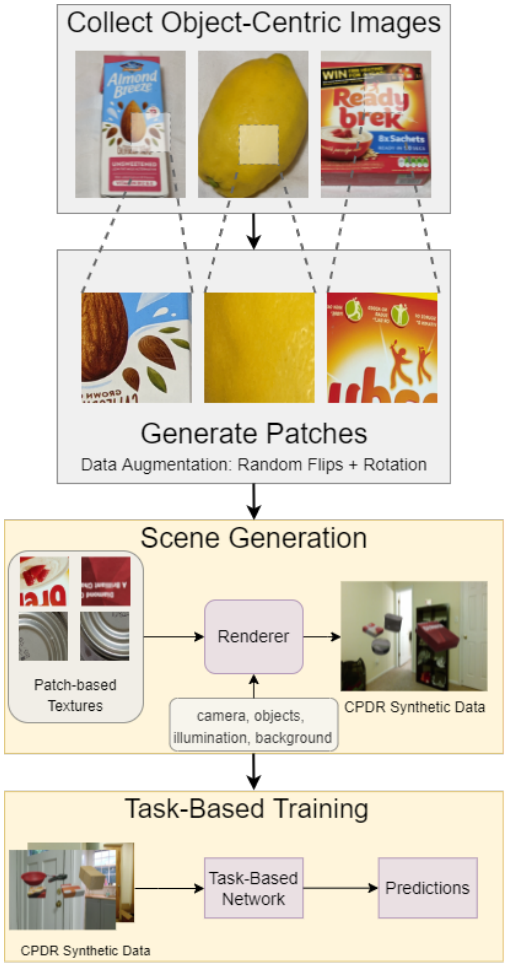


Fig. 1. We present an approach for synthesizing Domain Randomized synthetic images using textures based on real-world image patches for solving detection and segmentation tasks. Natural images provide complex features that are more difficult to attain with artificially created ones, while conditionally applying textures based on the objects of interest adds visually relevant information.

The appearance of manufactured objects, such as the presence or absence of text elements and choice of colors, is partially determined by object function and construction. We take advantage of the presence of these characteristics in natural images in a way that existing DR techniques do not.

Previous work has shown that the application of DR textures in object localization and detection does impact task-based performance [9], [10]. However, there is a wide range of texture synthesis techniques currently explored in the DR literature. These types of textures may include using a single

shade of color (flat RGB), a color gradient (gradient RGB), patterns such as checkerboard, zig zag, or striped, or adding additional noise patterns [11], [7], [2], [4], [9]. It is unclear which of these DR texture generation approaches would be the most suitable for a particular task.

This paper introduces *Conditional Patch-Based Domain Randomization (CPDR)*, an approach for synthesizing and applying more visually relevant textures for solving vision and robotics tasks using DR synthetic data as shown in Fig. 1. We do so by taking real-world images of objects and randomly sampling from those images, patches to use as the textures in the DR process as shown in Fig. 2. For example, we would apply cropped patches from random real-world images of boxes instead of using flat RGB as the textures for a cereal box when generating the DR synthetic data. This strategy enables us to apply textures that contain more visually relevant features from natural images such as patterns or text that often represent objects of that class. Assuming we know the types of objects, but not their exact appearance, we may employ CPDR for generating more applicable DR textures for solving a task, and outperform the existing DR systems used in the current literature. In scenarios where we do not know the types of objects, we find that unconditional application of patch-based textures from real-world images still outperforms the existing DR techniques. Our contributions are as follows:

- We present a novel approach for generating DR textures by sampling random patches from natural images containing real-world objects.
- We propose another method for generating DR textures by conditionally sampling image patches from real-world images during the DR process.
- We show that conditional application of patch-based textures similar to objects in the target dataset further improves performance compared to the unconditional application of randomly sampled patches.
- We show that our proposed approach using conditional class-specific patch-based textures outperforms the highest performing DR system through experimentation by 4.94 average precision (AP) and 6.71 AP in object detection and semantic segmentation tasks, respectively. We also outperform the existing methods by 2.85 AP on the detection task, and 4.30 AP on the segmentation task using patch-based textures applied unconditionally.

II. RELATED WORK

A. Synthetic Image Generation

Existing literature widely explored synthesizing images, particularly using generative adversarial networks (GANs) [12]. More recently, researchers have made progress towards generating higher quality and higher resolution images using improvements to generative models [13], [14], [15]. The methods have shown significant improvements in image quality, particularly at higher resolutions. Generally, these approaches to synthesizing synthetic images rely on systems trained on a significant amount of real-world data, such as the case with Brock et al. [14], which required training a GAN with 300M real-world images labeled with 18K categories.

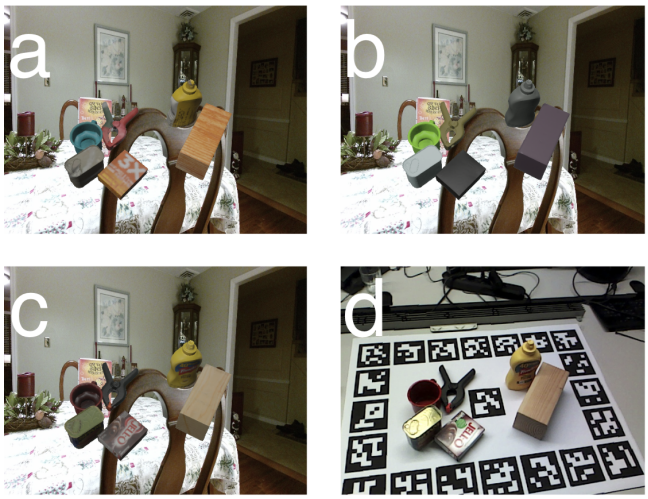


Fig. 2. Synthetic images were generated using poses from YCB-M [1] and textures synthesized using CDPR (a), flat RGB (b), and the true object texture (c). The CPDR approach applies textures visually more similar to object classes in the target dataset. The real-world image with the same poses is shown in d.

While the above recent approaches in image generation techniques result in high-fidelity synthetic images, it is more challenging to generate a specific scene explicitly. For example, we could not generate more complex scenes such as a tabletop scene containing precisely n amount of objects from a specific camera angle and under specific illumination conditions as we would using a traditional renderer. Furthermore, we would not have access to labeled data that describes the scene using this particular approach to generating synthetic data.

B. Domain Randomization

Commonly, DR approaches use a variety of textures to generate DR images for solving vision and robotics tasks. These textures are programmatically defined as a distribution of colors or patterns that we can sample our textures from. For example, while most works use simple textures such as shades or gradients of a color, some works also incorporate more complex patterned textures such as checkerboard, striped, or zig zag patterns [11], [16], [2], [7], [17], [10], [9] which recent work has shown to be more beneficial in increasing task-based performance [9].

While existing approaches have used real-world images as part of the randomization process [11], [3], [18], these are typically used to randomize the backgrounds, such as the work by Tremblay et al. [11]. Approaches may also use real-world images to supplement DR data, as is the case with Bousmalis et al. [3], where they used 9.4 million unlabeled real-world images to develop a generative model to improve grasping performance. We propose to instead use real-world images as sources of image patches, which can be used as

object textures. We hypothesize this procedure would improve task-based performance due to complex features with natural images that are more difficult to attain with artificially created ones. Such features include patterns, text, or barcodes from boxes, metallic surfaces for cans, or wooden grains for a wooden block.

III. Method

This section details the proposed method for generating textures from real-world image patches. We also describe the real-world datasets used to generate these patches, the datasets to generate the DR synthetic data used to train the deep learning models, and the real-world test set which would be used to evaluate the performance of the trained models.

A. Approach Overview

The proposed approach is a simple, fast, and effective method of producing complex patterned textures via random sampling of image patches from real-world images. Fig. 1 shows an overview of the approach:

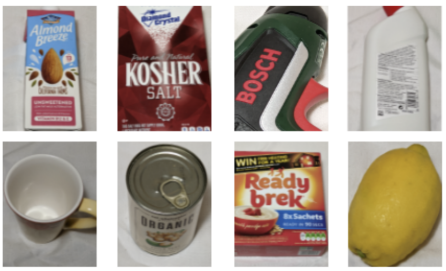
1) Collect images: Start by collecting a set of real-world object-centric images. In our scenario, we assume we have access to images with object-centric viewpoints, which occupy most of the image frame.
2) Generate patches: Given a set of real-world images, uniformly sample image patches of the desired size.
3) Data augmentation: Perform data augmentation on the previously generated image patches. We perform random rotations and random flips to further increase image patches diversity.
4) Scene generation: Apply the previously generated set of patches to the synthetic object meshes, and use additional parameters such as poses, backgrounds, and illumination to generate DR synthetic data using a renderer.

B. Synthetic Training Datasets

1) Image Collection: Our approach uses patches generated from a collection of real-world images that would be similar to the target dataset. For example, using household items when solving object detection on a kitchen countertop. In our experiments, we use two distinct real-world image datasets. The first dataset is a set of images containing household objects primarily used for training object detection systems [19]. Fig. 3 shows the collection of household objects in the dataset on the top, which contains 166 RGB images with 13 different object classes. Each image is of size 3264x1836 and contains annotations for 2D bounding boxes for each of the objects visible in a scene. The textures generated from this set of images are referred to as *Patches A*.

The second image dataset is a custom object-centric household dataset, which serves as a basis for our experiments' conditional application of textures. This dataset would allow us to investigate the scenario where we know the class of objects that would appear in the target dataset. This dataset

Patches A



Patches B

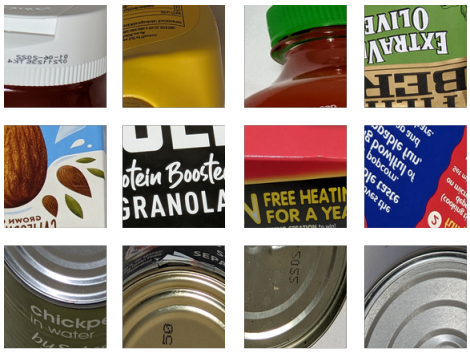
Fig. 3. All objects used for the *Patches A* [19] dataset on the top, and subset of objects used for *Patches B* on the bottom

contains additional household object classes that are more similar to the target objects, assuming we have access to the categories in the target dataset. The bottom of Fig. 3 shows several samples of the original images from this dataset, which contains 274 images with 7 object classes. As seen in Fig. 3, we position the objects over a white background around the center of the camera frame, ensuring each object occupies the majority of the image. Each image is of size 3024x4032 and contains class labels. The 7 object classes are: 'bottles', 'boxes', 'cans', 'fruits', 'pens', 'dishware', and 'tools'. To supplement this dataset, a collection of 15 real-world images of wood grain is added from Image*After [20]. These categories were selected such that they would be similar to the classes in the target dataset. The textures generated from this dataset will be referred to as *Patches B*.

2) Generating Patches: We uniformly sample across a set of given images to generate patches of size nxn. In the case of patches for *Patches A*, we use the provided 2D bounding boxes to ensure that each patch generated is contained around the object and not the background. A sample of the image patches of size 128x128 from this dataset is in Fig. 4.

Patches B did not require bounding box positions before cropping, as the dataset was collected to ensure the majority of the object is visible in the frame. Therefore, we uniformly sample the random image patches across the set of images for *Patches B*. Fig. 4 shows a sample of the textures of size 128x128, after being resized from the crops of size 512x512, generated using this method. The approach to generating

Patches A



Patches B

Fig. 4. Sample patches used for *Patches A* [19] dataset on the top, and sample conditional patches for *Patches B* on the bottom.

patches outlined above ensures we capture visually relevant patches for objects in a scene. The necessity of first cropping at 512x512 was due to the object occupying the majority of a high-resolution image. The initial crops at 512x512 ensure the visual features we desire are visible.

3) DR Textures: To compare the performance of our method against existing DR texture generation approaches, we also generate the types of DR textures used within the existing literature [11], [16], [2], [7], [17], [10], [9]. We reproduce the different textures currently used in the literature such as selecting a random RGB value (flat RGB), two random RGB values for a gradient (gradient RGB), number, size, and colors for the checkerboard patterns, striped patterns, and zig zag patterns. As is commonly practiced within the DR literature, the textures are created by sampling uniformly. Fig. 5 shows several samples from these textures.

4) Generating DR Synthetic Scenes: To evaluate the influence of DR texture selection, we apply each of the 14 available texture methods we are using to synthetic object meshes to render a synthetic scene. We fix object poses, illumination, backgrounds, and camera positions while generating a dataset per texture method we are investigating. To do this, we replicated scenes from the YCB-M [1] robotics benchmark dataset, which contains a subset of 20 objects from the YCB object dataset [21] in tabletop scenarios. The YCB-M dataset contains images of the same scene configurations from multiple cameras. We use the scenes

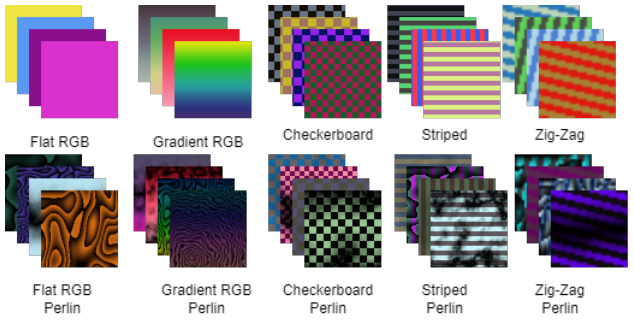


Fig. 5. Sample DR textures that have been reproduced from the existing literature.

from the viewpoint of an Intel RealSense R200 camera from the YCB-M dataset, as the images from this viewpoint and camera resulted in one of the highest performances when the authors evaluated the data on a pose estimation task compared to the other RGB images from the other cameras.

We use the annotations provided by the YCB-M dataset to reproduce the scenes, to generate a total of 14 synthetic datasets. Ten of the datasets use the current DR texture application techniques in the existing literature, three are using our approach, and the last dataset is the synthetic real textured dataset, which contains the original textures of the objects as provided by YCB [21]. The synthetic real textured dataset would give us an understanding of performance when the textures of the objects are known. Fig. 2 shows image samples from three synthetic datasets where we fix object poses, illumination, backgrounds, and camera positions across the datasets. We manually estimate the positions of the lights for each scene in the YCB-M dataset, as lighting information was not provided in the annotations. The backgrounds used to generate the DR synthetic scenes are backgrounds from the Active-Vision real-world dataset [22], as this has shown to be beneficial in aiding transfer from synthetic to real-world domains [23], [24], [11]. To ensure fairness across the different DR datasets, we used the same backgrounds for each scene within a particular dataset, such that the only difference between each image is the texture applied. For example, the backgrounds for the scenes generated using flat RGB would be the same backgrounds for the scenes generated using CPDR. To render the DR synthetic scenes, we built upon the tools by To et al. [25], which is an Unreal Engine [26] program to create DR images and annotations. We generate 3935 synthetic images for each of the DR textures applied for the training sets.

5) Real-world Test Dataset: The real-world test dataset consists of scenes from the YCB-M robotics benchmark dataset [1]. We use the RGB images from the Intel RealSense R200 camera from the YCB-M dataset, for the reasons previously mentioned above. We test on an 837 image subset from the YCB-M dataset, taken from 5 held-out scene configurations. We do so to ensure the training and test scene contents are disjoint, and that the test scenes contain all object classes present in the training set to evaluate the model's performance for solving the object detection and

semantic segmentation tasks. Samples of the training sets and test sets are shown in Fig. 6.

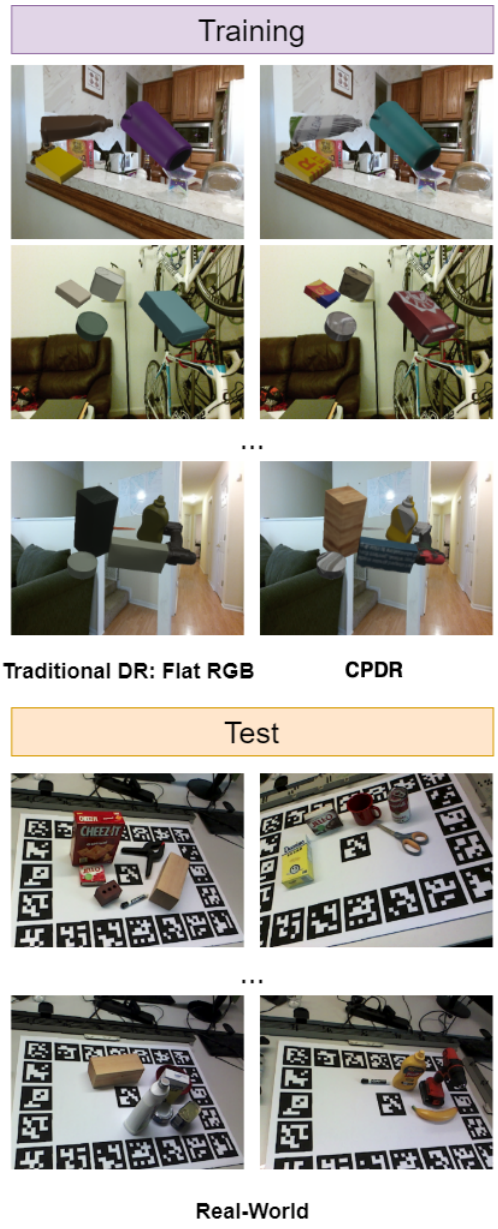


Fig. 6. Samples from the training datasets and test dataset.

C. Training Implementation

We train a widely adopted network architecture for solving object detection and semantic segmentation tasks [27]. We use Mask-RCNN with a ResNet-50 backbone, where features are extracted after the fourth convolutional block. Ensuring a fair comparison across previous experiments, we use the same hyperparameters across all experiments, which uses images of size 640x480, a batch size of 4, and an RTX Titan GPU. The models were trained using the SGD optimizer, with a base learning rate of 0.00025, a linear warmup factor of 0.001, weight decay of 0.0001, and momentum 0.9. All networks backbones were initialized using pre-trained MSCOCO weights [28] and fined-tuned on the DR datasets for 25k iterations. We adopt the framework provided by Wu et al. [29] to train our models.

IV. Experiments

We compare our approach to generating DR textures to the existing DR texture methods in object detection and segmentation tasks. The goals of our experiments are twofold: to evaluate if applying non-class-specific (unconditional) patch-based textures from natural images can improve task-based performance over the existing DR texture methods and if the conditional application of these textures is necessary. For completeness, we also include the results of training the models on the equivalent real-world images, and with a synthetic dataset containing the original object textures provided by the scans from YCB [21]. This gives us an understanding of performance when training on real-world images from the same domain, and performance when the synthetic objects' textures are known.

A. Evaluation Metrics

We evaluate the performance of the detection and segmentation tasks using the network architecture and setup described in Section III-C on standard COCO metrics, reporting average precision (AP) over IoU. Specifically, we use the primary challenge COCO metric where AP are averaged over 10 IoU thresholds between $[0.5 : 0.95]$ in increments of 0.05 [28]. We also report the conventional requirement for AP at IoU 0.5 (AP_{50}), based on the PASCAL Visual Object Classes (VOC) challenge [30], which defines a correct prediction when the IoU between a prediction and ground truth exceeds 0.5 (50%). We also report IoU at 0.75 (AP_{75}), which is considered a strict threshold [28].

B. Unconditional Real-World Image Patches

We apply patches generated from *Patches A* and *B* unconditionally, meaning the textures applied to an object do not correspond to a particular category. This method of texture application helps us investigate the application of textures that are randomized without enforcing class-specific information. For example, we can place patches from boxes on cans. Tables I and II show the results for the object detection and segmentation tasks, in the rows labeled 'Unconditional'.

We achieve an improvement of 0.98 AP and 2.85 AP over the highest performing DR system in the detection task when using the unconditional *Patches A*, and unconditional *Patches B*, respectively. We show an even larger improvement over the most widely used DR system (flat RGB) by 4.17 AP and 6.04 AP when using the unconditional *Patches A* and unconditional *Patches B*, respectively.

We observe a similar set of results when solving the segmentation task, outperforming the current literature's best DR system by 1.60 AP and 4.30 AP using the unconditional *Patches A* and unconditional *Patches B*, respectively.

One possible explanation for the difference in performance between *Patches A* and *B* is the types of objects that are

present in the two datasets. For example, *Patches B* contains fruits, wooden grains, tools, and dishware, which is not present in the *Patches A* dataset. Similar objects from these categories do appear in the real-world test set, and could explain the increase in performance, as they contain more visually relevant features.

TABLE I

RESULTS ON 837 REAL-WORLD IMAGE SUBSET OF 5 SCENES FROM THE YCB-M DATASET [1] FOR THE OBJECT DETECTION TASK, SORTED BY LOWEST TO HIGHEST AP SCORES.

Dataset	AP	AP50	AP75
Flat RGB Perlin	11.96	25.45	10.30
Zig Zag Perlin	12.20	28.68	8.03
Gradient RGB Perlin	12.22	25.63	9.87
Checkerboard Perlin	12.47	27.50	9.63
Flat RGB	13.16	27.44	11.36
Striped	13.44	28.05	11.81
Striped Perlin	13.51	30.05	10.75
Gradient RGB	14.05	27.75	12.75
Checkerboard	14.74	31.73	12.62
Zig Zag	16.35	34.34	13.94
Unconditional Patches A	17.33	36.48	14.42
Unconditional Patches B	19.20	41.58	15.18
Conditional Patches B	**21.29**	**42.42**	**19.11**
Synthetic Real Textured	29.81	58.45	26.94
Real World	39.40	72.90	39.07

TABLE II

RESULTS ON 837 REAL-WORLD IMAGE SUBSET OF 5 SCENES FROM THE YCB-M DATASET [1] FOR THE OBJECT SEGMENTATION TASK, SORTED BY LOWEST TO HIGHEST AP SCORES.

Dataset	AP	AP50	AP75
Flat RGB Perlin	8.11	17.09	6.79
Gradient RGB Perlin	8.71	18.98	7.66
Flat RGB	8.95	18.16	7.95
Striped Perlin	9.20	19.53	8.40
Zig Zag Perlin	9.26	19.62	8.28
Checkerboard Perlin	9.58	19.66	9.42
Striped	9.69	19.18	9.67
Gradient RGB	10.12	19.92	8.48
Checkerboard	11.18	23.04	10.49
Zig Zag	12.77	25.54	11.88
Unconditional Patches A	14.37	27.32	13.39
Unconditional Patches B	17.07	32.38	15.67
Conditional Patches B	**19.48**	**35.87**	**18.58**
Synthetic Real Textured	26.53	46.07	25.50
Real World	37.20	63.17	36.72

Here, we showed that our approach using natural image patches as the textures during the DR process outperforms the existing DR systems in both the detection and segmentation tasks. Despite applying textures from one type of object onto a different type of object, using patches from real-world images as the textures within the DR process can be a quick and easy way to integrate into existing systems to increase performance. This approach would not require devising complex artificial texture distributions to sample DR textures, as is currently the case with commonly used DR techniques, making it an appealing alternative. Although we achieved the highest performance when using *Patches B*, *Patches A* also outperformed the existing DR systems. This result indicates that we are still able to achieve an improvement over the existing techniques, without the need of collating custom image datasets.

Following this set of unconditional application of patches experiments, we now investigate the effectiveness of applying the patches conditionally. Meaning, applying patches from one object onto a similar type of object.

C. Conditional Real-World Image Patches

We have previously seen that using patches from real-world images as the textures when performing DR increases task-based performance, outperforming current implementations. This experiment examines the addition of class-specific information as part of the randomization process. For example, we would only be using patches of boxes on synthetic boxes or using patches of cans on synthetic cans. To investigate using class-specific information, we use textures generated from the *Patches B* dataset. Note that the *Patches B* dataset was gathered such that it contains similar object categories to the target dataset, assuming we have access to this prior information. While there is no direct mapping from each class in *Patches B* to the target real-world data from the YCB-M dataset, the classes are similar. For example, the patches used for the Cheez-it box, sugar box, pudding box, and gelatin box corresponding to objects in the YCB-M dataset, would be sampled from the 'box' category. Fig. 4 shows a subset of the different available classes in the *Patches B* dataset.

Referring back to Tables I and II we see the results when applying the image patches generated from *Patches B* conditionally, which inject additional visually relevant features into the training set. Conditional application of *Patches B* outperforms the existing DR methods by a greater margin than the unconditional application in detection and segmentation tasks, as shown in the row labeled 'Conditional'. When applying *Patches B* conditionally, we outperform the best DR system by 4.94 AP and 6.71 AP for the detection and segmentation tasks, respectively. Furthermore, when comparing *Patches B* applied conditionally and unconditionally, we see a performance increase from 19.20 AP to 21.29 AP for the detection task and 17.07 AP to 19.48 AP for the segmentation task. This result suggests that additional class-specific information when performing DR is beneficial.

V. CONCLUSION

In this paper, we presented a novel approach for generating textures for DR using textures generated from real-world image patches for solving object detection and segmentation tasks. We show that applying these textures unconditionally, meaning non-class specific textures, outperforms all existing DR texture randomization techniques that we evaluated on real-world data. We also show that conditionally applying the textures further increases performance compared to unconditional application. Functionally, the method is a fast, simple, and high-performing solution to generating DR textures, eliminating the need to devise suitable artificial texture distributions to sample DR textures as is the typical approach in the current literature.

While our approach outperforms the existing methods, we have utilized object-centric images to generate the patch-based textures, which are similar to our target dataset. In future work, we plan to explore generating patch-based textures from dissimilar sources. Furthermore, an area for improvement is to incorporate consistency between object poses and image backgrounds, allowing the placement of objects more naturally in a scene.

VI. Acknowledgement

We acknowledge MoD/Dstl and EPSRC for providing the grant to support the UK academics involvement in a Department of Defense funded MURI project through EPSRC grant EP/N019415/1. The research in this paper was supported in part by the Engineering and Physical Sciences Research Council (grant number EP/S032487/1).

References

[1] T. Grenzdörffer, M. Günther, and J. Hertzberg, "Ycb-m: A multi-camera rgb-d dataset for object recognition and 6dof pose estimation," *International Conference on Robotics and Automation (ICRA)*, pp. 3650–3656, 2020.

[2] S. James, A. J. Davison, and E. Johns, "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task," *Conference on Robot Learning (CoRL)*, 2017.

[3] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," *International Conference on Robotics and Automation (ICRA)*, pp. 4243–4250, 2018.

[4] 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 *Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Autonomous Driving*, 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] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] 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," *2017 International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

[8] F. Sadeghi and S. Levine, "CAD2RL: real single-image flight without a single real image," in *Robotics: Science and Systems XIII*, N. M. Amato, S. S. Srinivasa, N. Ayanian, and S. Kuindersma, Eds., 2017.

[9] M. Ani, H. Basevi, and A. Leonardis, "Quantifying the use of domain randomization," in *International Conference on Pattern Recognition (ICPR)*, 2021, pp. 6128–6135.

[10] J. Borrego, A. Dehban, R. Figueiredo, P. Moreno, A. Bernardino, and J. Santos-Victor, "Applying domain randomization to synthetic data for object category detection," *arXiv:1807.09834*, 2018.

[11] 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.

[12] 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 (NeurIPS)*, 2014, pp. 2672–2680.

[13] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *International Conference on Learning Representations (ICLR)*, 2018.

[14] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations (ICLR)*, 2019.

[15] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *Advances in neural information processing systems (NeurIPS)*, 2020.

[16] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric Actor Critic for Image-Based Robot Learning," in *Robotics: Science and Systems XIV*, 2018.

[17] S. Pouyanfar, M. Saleem, N. George, and S.-C. Chen, "ROADS : Randomization for Obstacle Avoidance and Driving in Simulation," in *Computer Vision and Pattern Recognition Workshops*, 2019.

[18] M. Yan, I. Frosio, S. Tyree, and J. Kautz, "Sim-to-Real Transfer of Accurate Grasping with Eye-In-Hand Observations and Continuous Control," *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Acting and Interacting in the Real World: Challenges in Robot Learning*, 2017.

[19] D. D. R. Meneghetti, P. H. S. Domingues, B. de Freitas Vece Perez, T. S. B. Meyer, K. K. G. Cardoso, A. M. de Lima, M. Y. Gonbata, F. de Assis Moura Pimentel, and P. T. A. Junior, "Annotated image dataset of household objects from the robofei@home team," in *IEEE Dataport*. IEEE Dataport, 2020.

[20] Image*After, "Wooden textures," http://www.imageafter.com/category.php?category=woods, 2019.

[21] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-Berkeley dataset for robotic manipulation research," *International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[22] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg, "A dataset for developing and benchmarking active vision," in *International Conference on Robotics and Automation (ICRA)*, 2017.

[23] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 1310–1319.

[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 *International Conference on Computer Vision (ICCV)*, 2015, pp. 2686–2694.

[25] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield, "NDDS: NVIDIA deep learning dataset synthesizer," 2018, https://github.com/NVIDIA/Dataset_Synthesizer.

[26] U. Engine. [Online]. Available: https://www.unrealengine.com/

[27] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 2961–2969.

[28] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., 2014, pp. 740–755.

[29] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[30] 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 (ICCV)*, vol. 111, no. 1, pp. 98–136, 2015.