

Synthetic Data Generation for Bridging Sim2Real Gap in a Production Environment

Parth Rawal^{1*}, Mrunal Sompura¹ and Wolfgang Hintze^{1,2}

¹Fraunhofer Institute for Manufacturing Technology and Advanced Materials (IFAM),
Ottenbecker Damm 12, 21684 Stade, Germany.

²Institute of Production Management and Technology (IPMT), Hamburg University of
Technology TUHH, Denickestraße 15, 21071 Hamburg, Germany.

*Corresponding author. E-mail: parth.rawal@ifam.fraunhofer.de

Abstract

Synthetic data is being used lately for training deep neural networks in computer vision applications such as object detection, object segmentation and 6D object pose estimation. Domain randomization hereby plays an important role in reducing the simulation to reality gap. However, this generalization might not be effective in specialized domains like a production environment involving complex assemblies. Either the individual parts, trained with synthetic images, are integrated in much larger assemblies making them indistinguishable from their counterparts and result in false positives or are partially occluded just enough to give rise to false negatives. Domain knowledge is vital in these cases and if conceived effectively while generating synthetic data, can show a considerable improvement in bridging the simulation to reality gap. This paper focuses on synthetic data generation procedures for parts and assemblies used in a production environment. The basic procedures for synthetic data generation and their various combinations are evaluated and compared on images captured in a production environment, where results show up to 15% improvement using combinations of basic procedures. Reducing the simulation to reality gap in this way can aid to utilize the true potential of robot assisted production using artificial intelligence.

Keywords: synthetic data, photorealistic rendering, production, sim2real gap, object detection

1 Introduction

The revolution in the computer graphic hardwares in the last decade has given birth to countless new Artificial Intelligence (AI) applications. Some of these applications are object detection, segmentation as well as pose estimation using deep learning. The time and effort needed for labelling data varies in different applications. While object classification only needs category labels for training, object detection and segmentation need bounding box and pixels coordinates respectively. In case of

object pose estimation, scene data such as 6D pose of all the objects in the images is needed which makes annotating real images extremely challenging [1–3]. Hence, for these applications, generating fast and easy synthetic data with annotations is indeed the preferred way in comparison to the time consuming and manual practice of labelling real data.

As far as training with synthetic data is concerned, a number of successful approaches have

been proposed and validated for datasets containing household objects with considerable amount of features. However, the components used in production are often textureless, reflective and colorless making them difficult to detect as compared to objects with varying features [4, 5]. Moreover, objects trained in a generic environment with domain randomization tend not to work so well for a specialized domain such as production environment [6]. For such scenarios, target domain knowledge is extremely essential to further bridge the simulation to reality gap [3, 7]. Particularly for the manufacturing industry, there are several challenges to be addressed while working with Convolution Neural Network (CNN). Complex assembly CAD (Computer Aided Design) models contain thousands of parts which have to be manually selected and exported as meshes for training. The individual parts after their integration into assemblies are often occluded and go undetected. Another problem is true detection of similar parts with slight differences. In addition to that, hundreds of unseen objects lying inside a production cell, pose a risk of getting detected as false positives.

This paper presents a solution to the previously mentioned problems in production facilities. In the beginning, an approach is proposed and implemented to automatically segregate the target parts from their assembly CAD model and export them in the form of mesh, which is used later for generating data. In case of synthetic data generation, not all of the domain randomization aspects can be combined in a single scene. For example, it has to be decided whether a texture or a background image needs to be chosen for a scene. In these cases, the generated dataset is a combined one without prior knowledge of their individual efficacies in real environments [2, 8, 9].

In this paper, the focus lies completely on various data generation procedures and photorealistic images are generated with the state of the art data generation tools. Five basic procedures with different levels of domain randomization and domain adaption are presented. Fig. 1 shows images from different basic procedures with different levels of domain randomization. The data generated from the basic procedures is then evaluated on real images inside a production cell. Based on the performance of the basic procedures, synthetic data is generated by combining basic procedures

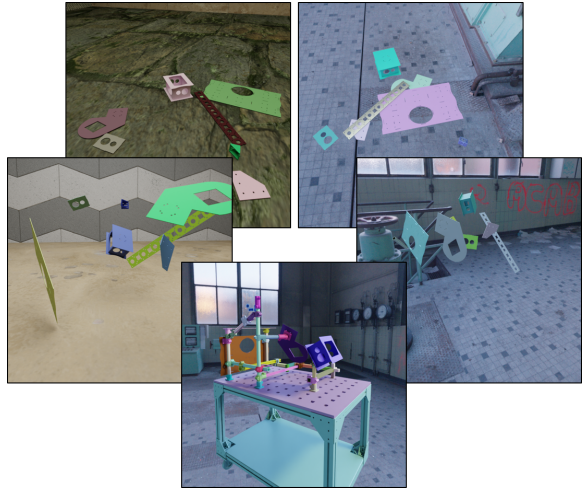


Fig. 1 Synthetic images generated with five different domain randomization procedures

in different proportions and is thereafter evaluated. Some of the combinations boost up the performance by 15% inside a production cell. The proposed work is developed in the form of an automated pipeline being able to generate thousands of images within few hours with a desired combination of basic procedures. The generated data can be used in robot assisted manufacturing applications for object classification, detection, segmentation and 6D pose estimation tasks.

The following section provides an overview of related work in the area of synthetic data generation and domain randomization and their applications in industrial environments. The reusable procedure based data generation pipeline is described in detail in Section 3. The evaluation of the data generated by procedures and their combinations on real images captured in a production environment is presented in Section 4. Finally, the discussion and outlook are mentioned in Section 5.

2 Related Work

Synthetic data generation is a dominating field in computer vision research in the last five years. Over the years, numerous different ways of generating data are presented. Based on the state of the art methods, these can be broadly divided into four parts. Firstly, cut and paste or render and paste where patches of objects are cropped and pasted on randomly or at specific realistic locations on background images after scaling and

transforming them [10, 11]. Data can be generated cheaply in this way, but they lack 3D scene information due to which they are not suitable for generating annotations for object pose estimation. Second approach uses physics engines or simulators, whose reviews are compiled by Collins et al. in [12]. The physics simulators are primarily used in the field of reinforcement learning due to a vast range of availability of sensors in the test environment. However, Borrego et al. in [1] and Tobin et al. in [8] point out a number of limitations using MuJoCo and Gazebo for generation of synthetic images. The third approach for generating quick domain randomized data uses game engines like Unreal Engine (UE4) and Unity3D and is depicted by Tremblay et al. [2] and To et al. [13]. Eventhough the game engines can render synthetic images reasonably well, they are tuned to provide realtime performance and lack dynamic characteristics of photorealism [14, 15]. Finally, the last method consists of physically based rendering (PBR), a technique which can achieve high level of photorealism. Cycles engine from Blender is an example of it. Mayershofer et al. in [3] conclude that a network trained on images generated with physics based rendering (Cycles) outperforms the one that was trained on images rendered by a game engine (EVEE).

Normally, generation of synthetic data, owing to the unavailability of the application programming interface (API), can be a daunting process and is not scalable. However, a number of recently developed pipeline projects on rendering engines have enabled a completely scalable data generation process. NDDS [13] project, based on Unreal Engine (UE4), was used to generate a dataset in [16]. BlenderProc [17] procedural pipeline supporting scripts, based on the Cycles engine from Blender, also showed improved performance by their generated data [18]. A very similar project, NViSII [15] was demonstrated on NVIDIA’s OptiX ray tracing engine. A more recent pipeline Kubric [19] used Blender and PyBullet to generate photorealistic data with ground truth.

The techniques of domain randomization and domain adaption have been widely used to bridge Sim2Real gap [1, 2, 8, 20, 21]. While domain randomization is very robust to varying environment, domain adaption helps to achieve a higher precision across domains. These researches show

that domain randomization can be easily implemented for synthetic data thereby generalizing the trained model to the real world and improve its performance. Domain adaption, on the other hand, reduces the gap by increasing the resemblance between the two domains. This can be done by either using a combination of synthetic and real images [22, 23], or by using synthetic photorealistic images resembling to the target domain [24], or by using generative adversarial networks (GAN) [25] based methods which can be used to transform generated synthetic images on the target domain [26–29]. However, domain adaption techniques, often being tricky, require greater manual effort compared to domain randomization as mentioned by Borrego et al. in [1]. The use of photorealism together with domain randomization leads to higher confidence values and enables training models without freezing the backbone layers [18, 30]. Some of the later researches have shown that combining domain randomization with target domain knowledge improves detection performance [3, 6, 7].

As far as reducing Sim2Real gap in a target domain is considered, almost all research papers use domain adaption techniques along with domain randomization. These have been presented for different applications in the fields of autonomous driving, logistics and industrial production [3, 7, 22, 23, 31–33]. Dümmel et al. in [33] use Autodesk Inventor modelling software for data generation and clearly show a performance improvement for an assembly use case. This was achieved by loading the relevant CAD models directly for rendering images of individual parts as well as entire assemblies. However, the data generation software lacks physics simulation and uses Unity3D as a simulation engine to get realistic orientations of the parts. While Moonen et al. [5] and Baaz et al. [23] generate synthetic images based on rendering pipeline from Unity, Mayershofer et al. [3], Eversberg and Lambrecht [7] and Andulkar et al. [34] use Blender API for data generation, making all the mentioned approaches scalable. [5] and [7] also show improvements for industrial use cases. Out of all the target domain based studies, only two works [3, 7] show a comparison between data generated with different parameters and their effect on results. This paper generalizes the data generation methods

using an approach of basic domain randomization procedures. Further, it compares the object detection performance for different combination of procedures in a production-related environment. Additionally, the paper also proposes a pipeline to handle complicated CAD models and export the parts and assembly models to mesh format with ease. The final outcome is an overall pipeline to generate synthetic data with a desired combination of procedures. This pipeline being already built on the scalable BlenderProc [17] pipeline, makes the entire data generation process scalable for multiple applications.

3 Synthetic Data Generation Pipeline

This section introduces the scalable pipeline approach for synthetic data generation in detail and the demonstrator used later for validation of the generated images.

3.1 Overall Concept

One of the attractive features of data generation is to have a reusable pipeline, where desired training data can be generated without additional effort or with minimum effort. For this reason, the current pipeline is built on the top of the scalable BlenderProc [17] pipeline. A comparable framework, NViSII [15] can also be used as an alternative to BlenderProc pipeline. The goal here is to generate training data from CAD models without manual efforts, making the framework reusable across multiple industrial applications. The procedures for data generation are implemented within the framework and enable the user to generate desired training data within a few hours. The vision of the framework is to generate the training data only once, while using it for training of multiple AI applications such as object classification, detection, segmentation and pose estimation.

The structure of the pipeline is depicted in Fig. 2. The pipeline can be divided into further three pipelines. Pipeline A reads the CAD model in the form of a STEP file and exports the parts of interest in mesh format for data generation. This is implemented using the free

and open-source tool *FreeCAD*¹. Pipeline B takes the mesh data and generates training data with predefined procedures using *BlenderProc*² [17]. Finally, pipeline C is the neural network training pipeline, which is not within the scope of this paper. Pipeline A and pipeline B are presented in detail in Section 3.2 and in Section 3.3 respectively. This pipeline approach encompasses overall data generation process starting from CAD model till training data generation. Additional tools are also developed to aid the visualization of ground truth generated by the data generation pipeline.

In order to validate the generated data in a real production related environment, a special demonstrator constructed for Skotty research project³ was used. The demonstrator consists of multiple parts which are simplified versions of components used in aviation industry in case of passenger aircrafts. The final application in Skotty research project is spray coating the components with wax.

Fig. 3 shows an image of the demonstrator used along with its various parts and sub-assemblies mounted on an industrial hand cart. The STEP model of the main assembly consists of altogether 148 components, comprising of various parts, sub-assemblies and the main assembly. Out of all components, only *ManholeBox* and *GeometricPlate* are chosen as categories for data generation whereas others are considered as passive objects in the scene. Both the components are asymmetric and contain some features helpful for object detection. While *ManholeBox* has distinctive features on all six sides, *GeometricPlate*, being a planar object, has significant features only on its top and bottom sides. This helps to validate the synthetic data for industrial parts having a planar geometry and their overall effect on the trained model.

3.2 Exporting Meshes

STEP file format is widely used during drafting components due to its accurate geometric representation. In the field of computer graphics, mesh

¹<https://www.freecad.org/>

²<https://github.com/DLR-RM/BlenderProc/>

³Skotty - Smart, collaborative, multitechnology production systems for coating aircraft components, Project timeline: 08.2021-12.2023 under grant ZW 1 80159842 is conducted with support of the Lower Saxony Ministry of Economic Affairs, Employment, Transport and Digitalization and the N-Bank

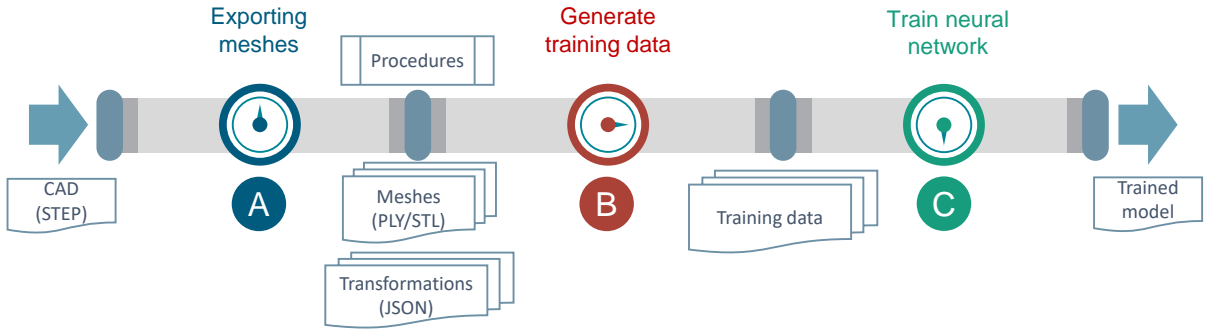


Fig. 2 Scalable pipeline concept from CAD model to trained model with interfaces inbetween

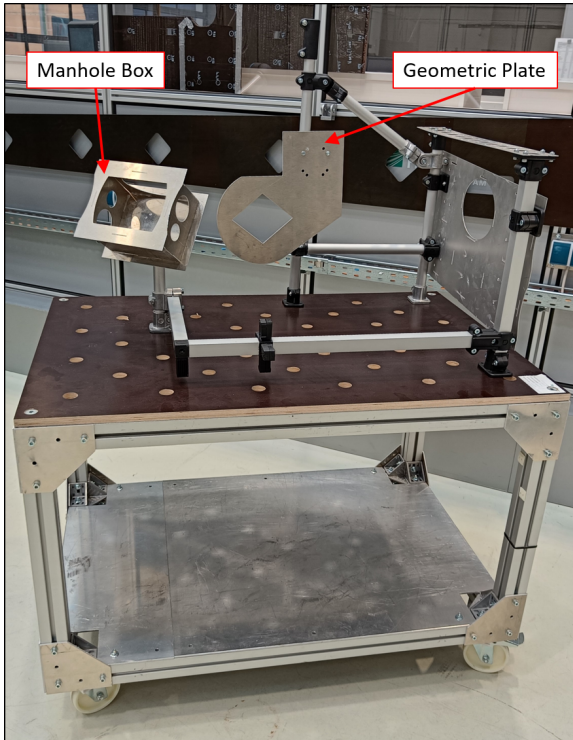


Fig. 3 Demonstrator used for validation of synthetic data in real environment

formats like STL and PLY are however more popular, which are approximations of geometric models. Hence, conversion from STEP format to mesh formats is an essential step for data generation as mesh formats are widely used for photorealistic renderings.

Segregating part files from assembly CAD model containing thousands of parts can be cumbersome and labour intensive. This pipeline partly automates the mesh generation process from assembly CAD model using scripts. For this, the

FreeCAD open-source tool is used which supports macros. The steps undertaken to export the target mesh files from a CAD model are depicted in Fig. 4. At first, the assembly STEP model is loaded in *FreeCAD*. A macro exports all the part, sub-assembly and main assembly file names to a CSV file called here as *PartList.csv*. As the next step, the user manually assigns a category name to all the parts or sub-assemblies to be classified and is saved as *CategoryList.csv*. This step is not extensive as normally not all the components from the assembly CAD model are needed for training. In the final step, another macro automatically generates the mesh files for all the components mentioned in *CategoryList.csv* and places them in a local folder *Classes*. The components which are in *PartList.csv* but not *CategoryList.csv* are considered as passive objects and their meshes are exported in a local folder *Structure*. It is noteworthy that the meshes are exported in the local coordinate frame, in which the parts were drafted and not in the main assembly coordinate frame. In addition, the transformation of the individual components in the main assembly coordinate frame are exported in JSON format as text files in the folders *Classes* and *Structure*. These transformations are a part of target domain knowledge and can be later used to recreate the scene where the components can be placed with positions and orientations as in the main assembly model. Transformations are saved in millimeter and as Euler angles in Roll-Pitch-Yaw convention as shown in Fig. 4.

The macro scripts used here are based on Python having file extension *FCMacro* and use *FreeCAD API* to extract useful information from the CAD models. The final outputs are the meshes and transformations in the folders *Classes* and

Structure. The proposed pipeline is easy to use without much adaption and can export the necessary meshes within a few minutes.

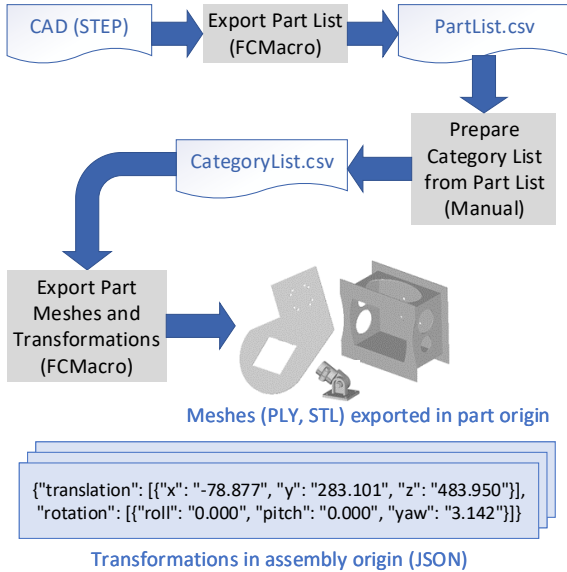


Fig. 4 Multiple steps for exporting part meshes and their transformations in assembly origin

3.3 Procedure based Data Generation

The creation of synthetic datasets involves artificial generation of data to mimic real world scenarios, which in turn requires the use of object meshes, photorealistic textures, real world backgrounds, and other elements for PBR. The resulting images can be either photorealistic or heavily randomized with random choice of colors. In order to establish a pipeline that can be used with different objects or combinations of rendering techniques, various procedures are proposed. These procedures can be imagined to be similar to cooking recipes. Just like the cooked dish may vary based on the cooking recipe followed, the synthetic images generated here also depend on the type of procedure used. These procedures include creating a 3D scene with textured planes or by using a background image with high dynamic range (HDR). Objects are either simulated to fall on the ground plane using physics simulation or are positioned randomly thereby floating in the 3D space. In context of image generation, objects on the

ground plane exhibit a deterministic behavior in the ground truth, whereas randomly posed floating objects demonstrate a stochastic behavior. For instance, a circular plate shaped planar object on being dropped onto a surface is unlikely to assume a vertical or angled position after undergoing physics simulation, thereby reducing randomness in the dataset. However, if the same plate is floating in 3D space with a camera perspective such that only the flat edges are seen and not its distinctive features, high losses while training can be observed. Combining these approaches helps to overcome these limitations and reduce Sim2Real gap. Table 1 provides a list of all the procedures and their respective variations.

Table 1 List of procedures for synthetic data generation

Procedures	Variation
Procedure1	Textured planes with objects on the ground
Procedure2	Textured planes with objects in 3D space
Procedure3	HDRI background with objects on the ground
Procedure4	HDRI background with objects in 3D space
Procedure5	HDRI background and target scene reconstruction

Fig. 5 provides a comprehensive explanation of the step by step data generation process for all procedures in a top-down approach. The configuration file stores various parameters such as the procedure(s) to be used for datasets, camera intrinsic matrix, camera resolution and simulation parameters. To generate the required number of images, each procedure has an outer loop that deals with the number of images required and an inner loop that handles the rendering process, including the addition of objects and distractors to the scene. Object transformations, such as scaling, sampling the pose, etc. are performed for each procedure. Procedure5 differs from all other procedures in terms of setting the object's pose. Specifically in Procedure5, the object's pose is determined using transformations exported to the JSON file in assembly origin, as outlined in Section 3.2. On the other hand, for all the remaining procedures, the pose is sampled randomly after the object has been added to the scene. If any objects overlap, they are reset until the condition is satisfied. Procedure1 and Procedure3 use

a physics-based approach, allowing the objects to fall down on the ground plane.

Various objects within the scene possess surface appearance properties that are randomized, including but not limited to base colour, roughness, specularity, and metalness. Additionally, the scene incorporates textured planes or backgrounds based on a predetermined procedure. The size of these planes can be customized as per requirement, and they are further enhanced with randomly generated textures sourced from *CCOTextures*⁴. In case of procedures with backgrounds, only indoor HDRI backgrounds are used from *PolyHaven*⁵. For Procedure3, an invisible plane has been added to the scene as a floor to enable working with physics simulation.

In order to achieve photorealistic results, random lighting conditions are introduced in the rendered scene. Three types of lights are employed: sun, point light source, and light plane. The intensity, colour, and position of each light source are randomized in accordance with the size of the textured plane. In each scene, one of these lights is randomly selected along with random parameters. Once the scene is prepared for image capture, the camera intrinsic matrix is defined. The camera position is sampled on a sphere around the mean location of the objects in the scene. The location of the camera is sampled on the sphere as per the defined radius and the number of image samples required in the inner loop.

The images based on the sampling of camera locations are rendered in accordance with the defined camera intrinsic matrix. Some samples of these generated images are presented in Fig. 6. The RGB images are saved as arrays within an HDF5 container, along with depth images, class segmentation images, and instance segmentation images that contain information regarding the category ID and name of all objects in the image. This is done using the BlenderProc [17] pipeline. The use of the HDF5 container is due to its compressibility, extensibility, and simplified I/O operations, which make data storage more manageable. Furthermore, a JSON file in COCO annotation format [35] containing information about the 2D bounding boxes for each category ID, is stored alongside the HDF5 files.

In order to utilize the dataset for object pose estimation, 6D object poses are also saved in a JSON file. Finally, camera intrinsic matrix parameters are also stored for future usage. This generated dataset can be utilized in various applications that employ deep learning techniques.

3.4 Ground Truth Visualization

The generated dataset contains a diverse array of images featuring disparate textures and backgrounds. For each image, the pose of an object is stored in the camera coordinate system in the form of homogenous transformation matrix. The translation vector is saved in meters as $[X, Y, Z]$, while the rotation matrix is a 3×3 matrix. This dataset conforms to the BOP [36] dataset format, which is commonly used for 6D pose estimation. As a result, existing deep learning models can be trained on the generated dataset with only a few minor modifications to the data loader.

In order to verify the ground truth generated along with the synthetic images and for preventing any potential issues arising from category ID discrepancies, the saved ground truth data is visualized where it is superimposed onto the generated RGB images. In Fig. 7, both 2D and 3D bounding boxes are visualized and plotted on the images for all the present objects. The 2D bounding box information was extracted from the COCO annotations JSON file for each object in each image. For the 3D bounding boxes with origin, object mesh files in PLY format were loaded, and their dimensions were saved in separate JSON files containing information such as diameter, minimum dimensions, and size in the X, Y, and Z directions. Once the position, orientation, and 3D bounding box of an object are obtained, a transformation is applied from the camera coordinate system to the image coordinate system using the camera intrinsic matrix and the 3D bounding box is projected on the image as seen in the right image of Fig. 7.

4 Validation

The datasets generated with all five basic procedures and their combinations were evaluated for object detection using YOLOv7 [37]. YOLOv7 implementation, with its evaluation scripts, enables quick and easy calculation of object detection metrics. For a fair comparison,

⁴<https://ambientcg.com/>

⁵<https://polyhaven.com/hdri>



Fig. 5 Five basic data generation procedures explained in top-down approach

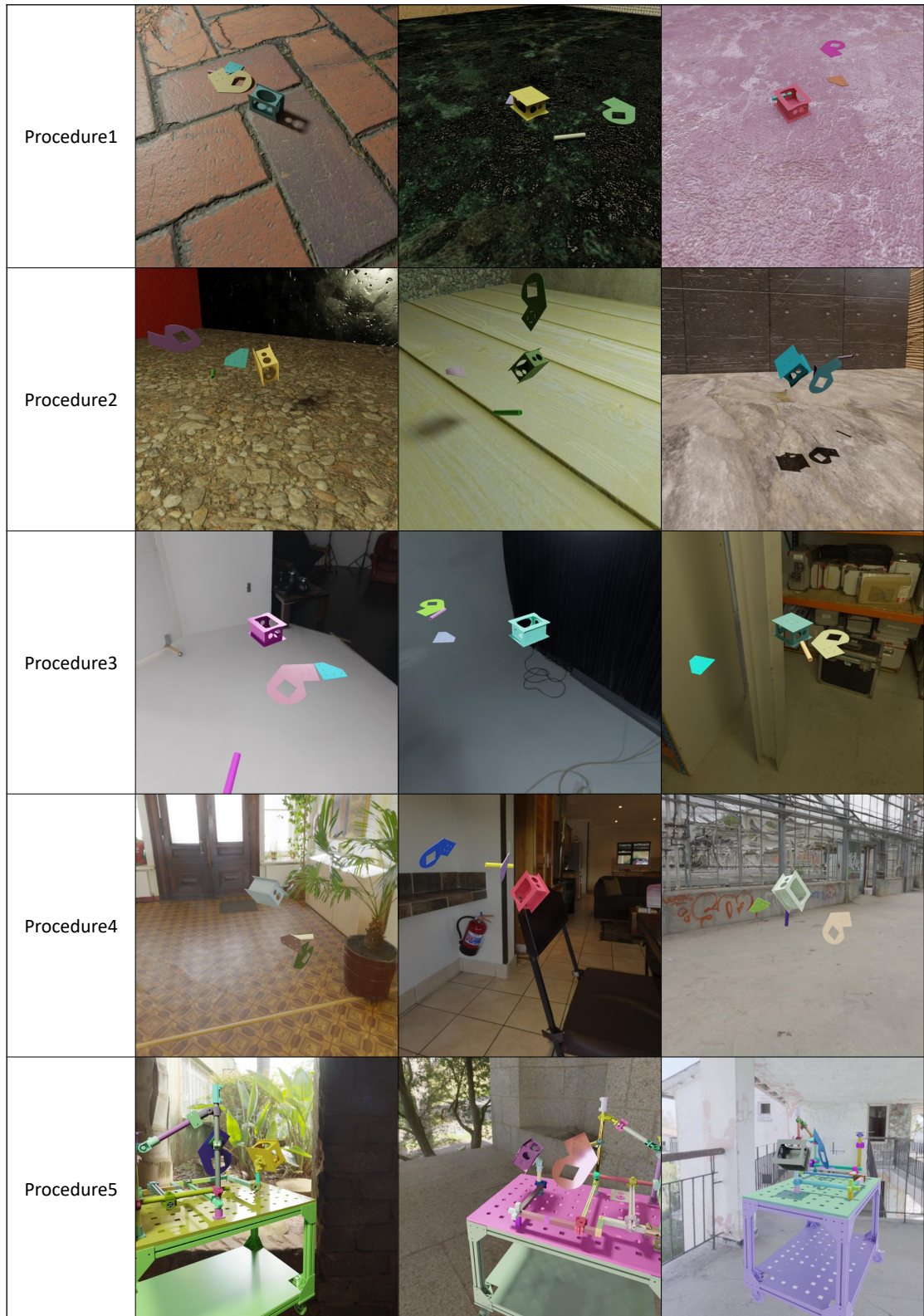


Fig. 6 Synthetic images generated from basic procedures

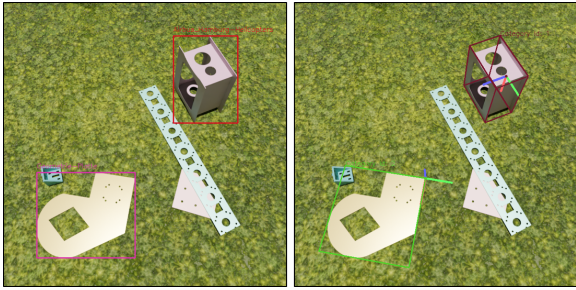


Fig. 7 Visualization of annotations on images. Left image: Plotting 2D bounding boxes and category labels on image. Right image: Plotting 3D bounding boxes and object coordinate system on image

all the configuration parameters and hyperparameters were kept unchanged. The normal YOLOv7 model mentioned in [37] was chosen and was trained for 100 epochs. Datasets with 15 000 images and 70/30 train test split ratio were generated for both the objects together. For combination datasets with a mixed procedures, the same proportion which was used while dataset generation, was maintained for train and test datasets. The time taken to generate a single dataset on a computer equipped with NVIDIA RTX A4000 took between 9 to 12 hours depending on the procedure(s) used. Altogether, five basic procedures and five combination sets of basic procedures were validated on 500 synthetic and 150 real images.

The synthetic images for validation were generated separately with each of the five basic procedures, while the real images were captured with a camera in a production environment and annotated later. The captured real images were taken in two scenarios: first, where both the objects were loose and second, where they were assembled in the demonstrator. For all the scenarios, COCO’s standard metric [35] denoted as $mAP@[0.5, 0.95]$ and PASCAL VOC’s metric [38] denoted as $mAP@0.5$ are calculated. However, only $mAP@[0.5,0.95]$ results are presented here as they put a larger emphasis on the localization of bounding boxes [39].

4.1 Evaluation of Basic Procedures

Table 2 and Table 3 draw a performance comparison between basic procedures for *ManholeBox* and *GeometricPlate* respectively. Here, P is the short form for representing the procedures. It can be seen that all the diagonal elements of the 5x5

Table 2 Performance comparison of basic procedures for *ManholeBox*

<i>ManholeBox</i> $mAP@[0.5:0.95]$	Trained on					
	Validated on	P1	P2	P3	P4	P5
P1		0.98	0.99	0.98	0.99	0.48
P2		0.75	0.99	0.83	0.98	0.53
P3		0.98	0.99	0.99	0.99	0.40
P4		0.75	0.98	0.89	0.99	0.56
P5		0.43	0.91	0.69	0.90	0.99
Average Sim		0.78	0.97	0.88	0.97	0.59
Real parts loose		0.25	0.70	0.35	0.46	0.31
Real assembly		0.00	0.51	0.00	0.58	0.74
Average Real		0.12	0.61	0.17	0.52	0.53

Note: Values in bold indicate the maximum average value across different trained models.

matrix have the best results across different synthetic validation sets. This behavior is expected as models trained on synthetic data using any single procedure have best results if it is validated on the synthetic dataset created with the same procedure. However, some procedures also perform better on validation datasets generated by other procedures. As an example, the average simulation values for P2 and P4 show that they perform overall better on all procedures.

Table 3 Performance comparison of basic procedures for *GeometricPlate*

<i>GeometricPlate</i> $mAP@[0.5:0.95]$	Trained on					
	Validated on	P1	P2	P3	P4	P5
P1		0.96	0.98	0.97	0.98	0.19
P2		0.52	0.94	0.51	0.92	0.11
P3		0.96	0.99	0.99	0.99	0.16
P4		0.58	0.95	0.60	0.96	0.07
P5		0.37	0.73	0.35	0.66	0.99
Average Sim		0.68	0.92	0.68	0.90	0.30
Real parts loose		0.70	0.03	0.34	0.52	0.00
Real assembly		0.72	0.70	0.30	0.86	0.24
Average Real		0.71	0.36	0.32	0.69	0.12

Note: Values in bold indicate the maximum average value across different trained models.

The performance of the basic procedures on real dataset drops to a considerable amount which is due to simulation to reality gap. In case of *ManholeBox*, model trained on P2 performs the best on real datasets overall. P2 consists of object images with textured planes without simulating physics or letting the objects float in the space. In contrast to that, for *GeometricPlate*, model trained on P1 performs the best. P1 consists of object images with textured plate with physics simulation, meaning that the objects are lying on the ground plane.

This behaviour too can be explained from the geometry of the objects. *ManholeBox* having geometry on all six sides can be detected easily when placed with random 3D poses in space. On the contrary, *GeometricPlate* being a planar object, has geometrical features only on two of the six sides. A random sampling of poses in space will not work the best as the features are not always visible from all perspectives. In this case, when the object falls on the ground plane, most of the features are always visible to the camera as the object has less probability to fall in a position which places itself into an unstable state. Letting objects fall on the ground plane resemble to a deterministic behavior whereas the objects lying in 3D space resemble to a stochastic behavior. The results show that for planar objects like *GeometricPlate*, synthetic images taken after letting them fall on the ground plane yield better results. For other objects, random sampling of poses as shown in P2 is helpful. Model trained on P4, which is rendering with a random background in 3D space, also shows good results for both objects. This is because backgrounds, even though not photorealistic, do contain high variance of features as compared to textures which are photorealistic.

Table 4 Composition of combination datasets from basic procedures

Combinations	Basic procedures				
	P1	P2	P3	P4	P5
C1	20%	20%	10%	30%	20%
C2	40%	0%	0%	40%	20%
C3	0%	40%	0%	40%	20%
C4	0%	0%	0%	80%	20%
C5	50%	0%	0%	50%	0%

4.2 Evaluation of Combination Datasets

Based on the performance of the basic procedures on the real datasets, five different combination datasets made up of different proportion of synthetic data from basic procedures were generated. Table 4 shows the composition of different combination datasets represented from C1 till C5. These datasets were used for training and were validated on simulated and real images.

Table 5 Performance comparison of combination datasets for *ManholeBox*

<i>ManholeBox</i> mAP@[0.5:0.95]	Trained on				
	Validated on	C1	C2	C3	C4
P1	0.99	0.99	0.99	0.99	0.98
P2	0.98	0.98	0.98	0.98	0.96
P3	0.99	0.99	0.99	0.99	0.99
P4	0.99	0.98	0.99	0.99	0.97
P5	0.99	0.99	1.00	0.99	0.87
Average Sim	0.99	0.99	0.99	0.99	0.95
Real parts loose	0.51	0.61	0.52	0.47	0.41
Real assembly	0.27	0.78	0.72	0.85	0.20
Average Real	0.39	0.69*	0.62*	0.66*	0.31

Note: Values in bold indicate the maximum average value across different trained models.

* Better results than the best performing basic procedure.

Table 5 and Table 6 draw a performance comparison between combination datasets for *ManholeBox* and *GeometricPlate* respectively. It can be easily seen that in general the combination datasets perform way better for synthetic images generated from all procedure unlike in case of basic procedure mentioned in Section 4.1. For real images in a production environment, C2, C3 and C4 perform better than the best performing basic procedure for *Manholebox*, while C4 and C5 perform better than the best performing basic procedure for *GeometricPlate*.

Summarizing the results of models trained on combination datasets, in case of *Manholebox*, there is upto 13% improvement whereas for *GeometricPlate*, performance improvement of about 5% is observed. A proper mixture of procedures

Table 6 Performance comparison of combination datasets for *GeometricPlate*

<i>GeometricPlate</i> mAP@[0.5:0.95]	Trained on					
	Validated on	C1	C2	C3	C4	C5
P1	0.98	0.99	0.98	0.98	0.97	0.97
P2	0.94	0.90	0.94	0.91	0.83	0.83
P3	0.99	0.99	0.99	0.99	0.97	0.97
P4	0.96	0.95	0.96	0.96	0.88	0.88
P5	0.99	0.99	0.99	0.99	0.99	0.54
Average Sim	0.97	0.96	0.97	0.97	0.84	0.84
Real parts loose	0.06	0.36	0.18	0.49	0.68	0.68
Real assembly	0.86	0.94	0.82	0.96	0.81	0.81
Average Real	0.46	0.65	0.50	0.73*	0.74*	0.74*

Note: Values in bold indicate the maximum average value across different trained models.

* Better results than the best performing basic procedure.

can yield better performance thereby bridging the simulation to reality gap.

4.3 Final results and discussion

The results from basics and combination datasets for both objects are summarized in Table 7. These values are the mean values of the performance of *ManholeBox* and *GeometricPlate* on validation datasets.

Combination datasets show varied overall performance. While the C2 and C4 combinations show improved performance of 11% and 15% respectively with respect to the best performing basic procedure, other combinations do not prove to be effective enough. Referring to Table 4 again, C2 is composed of three basic procedures: objects on the ground plane with textures, objects in 3D space with indoor background and objects placed as defined in assembly CAD model with indoor backgrounds. C4 is only composed of two basic procedures: objects in 3D space with indoor background and objects placed as defined in assembly CAD model with indoor backgrounds. Textured planes introduce photorealistic characteristics such as reflection and shadows to the rendered images whereas indoor backgrounds introduce versatility in the images reducing the detected false positives. The standalone performance of the target domain based P5 using assembly CAD and indoor backgrounds is not satisfactory on real

images but when this procedure is combined in given proportions with other procedures, a performance boost is observed. However, the best performance yielded by C4 does not consist of photorealistic images with textured planes and implies that the need of photorealistic data using textured planes, even after delivering good results, is not a must for performance improvement.

Fig. 8 shows some real world object detection results in a production environment using the trained model on C2 dataset. A score threshold of 0.80 was used while inferencing. Fig. 9 depicts examples of improved performance on combination dataset C2 compared to a basic procedure P4. A reduction in the occurrence of false positives and false negatives can be observed here. For the current example, model trained on C2 dataset was compared with model trained on P4.

5 Conclusion

In this work, a scalable pipeline based approach with procedures based on domain randomization and domain adaption techniques for photorealistic synthetic data generation is presented. Object detection models trained on five basic procedures are validated on real world images inside a production environment and the results are used to derive combination datasets from the best performing procedures. The models thus trained on some of the combination datasets show an improved performance than those which were purely trained on datasets with basic procedures. The final results bridge the Sim2Real gap with about 15% of performance improvement using combination of basic procedures on real images as compared to the best performing basic procedure. The achieved drastic performance improvement is due to the knowledge of the target domain undertaken while generating synthetic data. This is done by constructing a scene similar to the structure in the CAD model. The models trained with only synthetic images on YOLOv7 [37] show promising performance in a real world production environment.

The pipeline approach presented here starts from generating the mesh files and their transformations in assembly origin from the assembly STEP model using scripts. Then next module is built on the top of BlenderProc pipeline [17] and generates a synthetic dataset with a desired combination of basic procedures, which can then be

Table 7 Results summary

Both objects mAP@[0.5:0.95]	Trained on										
	Validated on	P1	P2	P3	P4	P5	C1	C2	C3	C4	C5
P1	0.97	0.99	0.98	0.99	0.34	0.99	0.99	0.99	0.99	0.99	0.98
P2	0.64	0.97	0.67	0.95	0.32	0.96	0.94	0.96	0.96	0.95	0.90
P3	0.97	0.99	0.99	0.99	0.28	0.99	0.99	0.99	0.99	0.99	0.98
P4	0.66	0.97	0.74	0.98	0.31	0.97	0.96	0.97	0.97	0.97	0.92
P5	0.40	0.82	0.52	0.78	0.99	0.99	0.99	0.99	0.99	0.99	0.70
Average Sim	0.73	0.95	0.78	0.94	0.45	0.98	0.97	0.98	0.98	0.98	0.90
Real parts loose	0.47	0.36	0.34	0.49	0.15	0.28	0.48	0.35	0.48	0.48	0.54
Real assembly	0.36	0.60	0.15	0.72	0.49	0.56	0.86	0.77	0.90	0.90	0.51
Average Real	0.42	0.48	0.25	0.60	0.32	0.42	0.67*	0.56	0.69*	0.69*	0.53

Note: Values in bold indicate the maximum average value across different trained models.

* Better results than the best performing basic procedure.

used for training neural networks for tasks such that object detection and object pose estimation. This enables easy generation of thousands of annotated images in few hours.

From the validation of the dataset, it can be concluded that inclusion of target domain plays an important role for bridging Sim2Real gap. The use of photorealistic images leads to large improvements and hence reduces the size of training datasets and time required for training. For the future work, some of the GAN based approaches can be tested and evaluated in combination with these procedures. Moreover, the generated synthetic data can be validated for object pose estimation neural networks to check if the same procedures also yield better performance in those cases.

Acknowledgments. We are grateful for the valuable expert information provided especially by every member of the project team Integrated Production Systems as well as the Head of Stade Branch Dr. Dirk Niermann at Fraunhofer IFAM.

References

- [1] Borrego J, Dehban A, Figueiredo R, Moreno P, Bernardino A, Santos-Victor J.: Applying Domain Randomization to Synthetic Data for Object Category Detection. Available from: <https://arxiv.org/pdf/1807.09834>.
- [2] Tremblay J, Prakash A, Acuna D, Brophy M, Jampani V, Anil C, et al.: Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. Available from: <https://arxiv.org/pdf/1804.06516>.
- [3] Mayershofer C, Ge T, Fottner J. Towards Fully-Synthetic Training for Industrial Applications. In: LISS 2020. Springer, Singapore; 2021. p. 765–782. Available from: https://link.springer.com/chapter/10.1007/978-981-33-4359-7_53.
- [4] C A Akar, J Tekli, D Jess, M Khoury, M Kamradt, M Guthe. Synthetic Object Recognition Dataset for Industries. In: 2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI); 2022. p. 150–155.
- [5] Moonen S, Vanherle B, de Hoog J, Bourgana T, Bey-Temsamani A, Michiels N. CAD2Render: A Modular Toolkit for GPU-Accelerated Photorealistic Synthetic Data Generation for the Manufacturing Industry; 2023. p. 583–592. Available from:

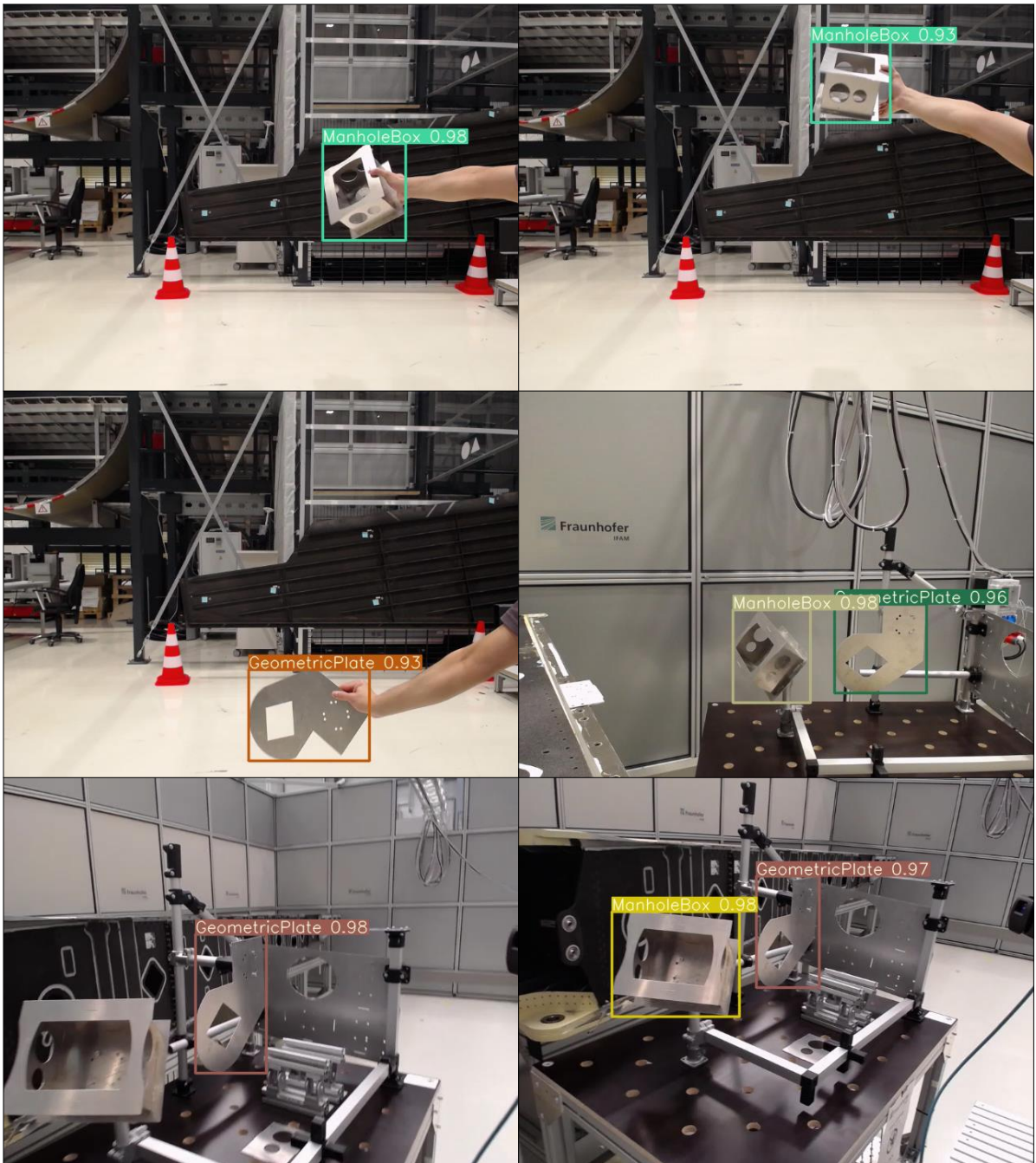


Fig. 8 Object detection results for both objects with model trained on C2 dataset in a production environment with different scenarios

https://openaccess.thecvf.com/content/WACV2023W/PIES-CV/html/Moonen-CAD2Render_A_Modular_Toolkit_for_GPU-Accelerated_Phorealistic_Synthetic_Data_Generation_WACVW_2023_paper.html.

html.

- [6] A Prakash, S Boochoon, M Brophy, D Acuna, E Cameracci, G State, et al. Structured Domain Randomization: Bridging the Reality

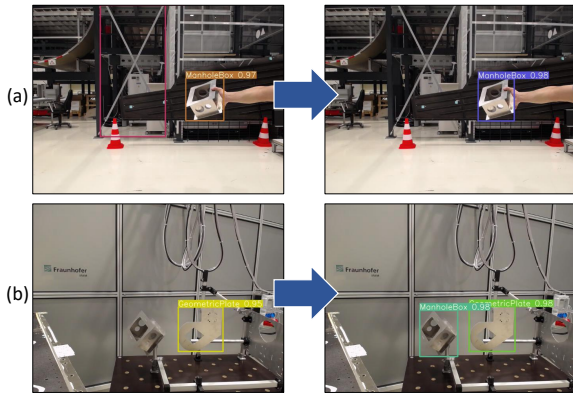


Fig. 9 Validation images for model trained on C2 dataset showing improved precision and recall over model trained on basic procedure P4. Left images in (a) and (b) show a false positive and a false negative respectively which was correctly handled in the right images using model trained on C2

Gap by Context-Aware Synthetic Data. In: 2019 International Conference on Robotics and Automation (ICRA); 2019. p. 7249–7255.

- [7] Eversberg L, Lambrecht J. Generating Images with Physics-Based Rendering for an Industrial Object Detection Task: Realism versus Domain Randomization. *Sensors*. 2021;21(23):7901. <https://doi.org/10.3390/s21237901>.
- [8] Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P. Domain randomization for transferring deep neural networks from simulation to the real world. In: *IROS Vancouver 2017*. Piscataway, NJ: IEEE; 2017. p. 23–30.
- [9] Nowruzzi FE, Kapoor P, Kolhatkar D, Hassanat FA, Laganieri R, Rebut J.: How much real data do we actually need: Analyzing object detection performance using synthetic and real data. Available from: <https://arxiv.org/pdf/1907.07061>.
- [10] D Dwibedi, I Misra, M Hebert. Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection. In: 2017 IEEE International Conference on Computer Vision (ICCV); 2017. p. 1310–1319.
- [11] Georgakis G, Mousavian A, Berg AC, Kosecka J.: Synthesizing Training Data for Object Detection in Indoor Scenes. Available from: <https://arxiv.org/pdf/1702.07836>.
- [12] Collins J, Chand S, Vanderkop A, Howard D. A Review of Physics Simulators for Robotic Applications. *IEEE Access*. 2021;9:51416–51431. <https://doi.org/10.1109/ACCESS.2021.3068769>.
- [13] To T, Tremblay J, McKay D, Yamaguchi Y, Leung K, Balanon A, et al.: NDDS: NVIDIA Deep Learning Dataset Synthesizer. Available from: https://github.com/NVIDIA/Dataset_Synthesizer.
- [14] Šmíd A.: Comparison of unity and unreal engine. Available from: <https://core.ac.uk/download/pdf/84832291.pdf>.
- [15] Morrical N, Tremblay J, Lin Y, Tyree S, Birchfield S, Pascucci V, et al.: NViSII: A Scriptable Tool for Photorealistic Image Generation. Available from: <https://arxiv.org/pdf/2105.13962>.
- [16] J Tremblay, T To, S Birchfield. Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW); 2018. p. 2119–21193.
- [17] Denninger M, Sundermeyer M, Winkelbauer D, Zidan Y, Olefir D, Elbadrawy M, et al.: BlenderProc. Available from: <https://arxiv.org/pdf/1911.01911>.
- [18] Denninger M, Sundermeyer M, Winkelbauer D, Olefir D, Hodan T, Zidan Y, et al. BlenderProc: Reducing the Reality Gap with Photorealistic Rendering. In: *International Conference on Robotics: Science and Systems, RSS 2020*; 2020. Available from: <https://elib.dlr.de/139317/>.
- [19] Greff K, Belletti F, Beyer L, Doersch C, Du Yilun, Duckworth D, et al.: Kubric: A scalable dataset generator. Available from: <https://arxiv.org/pdf/2203.03570>.
- [20] Ren X, Luo J, Solowjow E, Ojea JA, Gupta A, Tamar A, et al.: Domain Randomization

for Active Pose Estimation. Available from: <https://arxiv.org/pdf/1903.03953>.

- [21] S Hinterstoisser, O Pauly, H Heibel, M Martina, M Bokeloh. An Annotation Saved is an Annotation Earned: Using Fully Synthetic Training for Object Detection. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW); 2019. p. 2787–2796.
- [22] Y Huangfu, W Deng, B Ren, J Ding. A Generation Method of Synthetic Images with Reduced Domain Gap for Car Detection. In: 2021 5th CAA International Conference on Vehicular Control and Intelligence (CVCI); 2021. p. 1–6.
- [23] Baaz A, Yonan Y, Hernandez-Diaz K, Alonso-Fernandez F, Nilsson F.: Synthetic Data for Object Classification in Industrial Applications. Available from: <https://arxiv.org/pdf/2212.04790>.
- [24] J Dümmel, X Gao. Object Re-Identification with Synthetic Training Data in Industrial Environments. In: 2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP); 2021. p. 504–508.
- [25] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Nets. *Advances in Neural Information Processing Systems*. 2014;27.
- [26] A Shrivastava, T Pfister, O Tuzel, J Susskind, W Wang, R Webb. Learning from Simulated and Unsupervised Images through Adversarial Training. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017. p. 2242–2251.
- [27] S Sankaranarayanan, Y Balaji, A Jain, S N Lim, R Chellappa. Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2018. p. 3752–3761.
- [28] X Peng, K Saenko. Synthetic to Real Adaptation with Generative Correlation Alignment Networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV); 2018. p. 1982–1991.
- [29] P Rojtborg, T Pöllabauer, A Kuijper. Style-transfer GANs for bridging the domain gap in synthetic pose estimator training. In: 2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR); 2020. p. 188–195.
- [30] T Hodaň, V Vineet, R Gal, E Shalev, J Hanzelka, T Connell, et al. Photorealistic Image Synthesis for Object Instance Detection. In: 2019 IEEE International Conference on Image Processing (ICIP); 2019. p. 66–70.
- [31] F Reway, A Hoffmann, D Wachtel, W Huber, A Knoll, E Ribeiro. Test Method for Measuring the Simulation-to-Reality Gap of Camera-based Object Detection Algorithms for Autonomous Driving. In: 2020 IEEE Intelligent Vehicles Symposium (IV); 2020. p. 1249–1256.
- [32] Schoepflin D, Iyer K, Gomse M, Schüppstuhl T. Towards Synthetic AI Training Data for Image Classification in Intralogistic Settings. In: *Annals of Scientific Society for Assembly, Handling and Industrial Robotics 2021*. Springer, Cham; 2022. p. 325–336. Available from: <https://link.springer.com/chapter/10.1007/978-3-030-74032-0-27>.
- [33] Dümmel J, Kostik V, Oellerich J. Generating Synthetic Training Data for Assembly Processes. Springer, Cham; 2021. p. 119–128. Available from: https://link.springer.com/chapter/10.1007/978-3-030-85910-7_13.
- [34] M Andulkar, J Hodapp, T Reichling, M Reichenbach, U Berger. Training CNNs from Synthetic Data for Part Handling in Industrial Environments. In: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE); 2018. p. 624–629.
- [35] Lin TY, Maire M, Belongie S, Bourdev L, Girshick R, Hays J, et al.: Microsoft COCO: Common Objects in Context. Available from: <https://arxiv.org/pdf/1405.0312>.

- [36] Hodan T, Michel F, Brachmann E, Kehl W, Buch AG, Kraft D, et al.: BOP: Benchmark for 6D Object Pose Estimation. Available from: <https://arxiv.org/pdf/1808.08319>.
- [37] Wang CY, Bochkovskiy A, Liao HYM.: YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. Available from: <https://arxiv.org/pdf/2207.02696>.
- [38] Everingham M, van Gool L, Williams CKI, Winn J, Zisserman A. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*. 2010;88(2):303–338. <https://doi.org/10.1007/s11263-009-0275-4>.
- [39] S Bell, C L Zitnick, K Bala, R Girshick. Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017. p. 2874–2883.