Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 253 (2025) 1668–1679

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Conference on Industry 4.0 and Smart Manufacturing

Synthetic data generation procedures for domain-specific environments in manufacturing

Parth Rawal[a,*], Mrunal Sompura[a], Wolfgang Hintze[a,b]

[a]*Fraunhofer Institute for Manufacturing Technology and Advanced Materials (IFAM), Ottenbecker Damm 12, 21684 Stade, Germany*
[b]*Institute of Production Management and Technology (IPMT), Hamburg University of Technology TUHH, Denickestraße 15, 21071 Hamburg, Germany*

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 always be effective in specialized domains like manufacturing that involve complex assemblies. Individual parts are integrated in much larger assemblies making them indistinguishable from their counterparts. Moreover, individual parts are often partially occluded in the scene. These situations give rise to wrong detections. Target 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 validates synthetic data generation procedures through practical experimentation ensuring that experiments are both comprehensive and reproducible. After combining domain randomization and domain adaptation procedures for parts and assemblies used in manufacturing the model performance improves by up to 15% than the state-of-the-art domain randomization techniques. Reducing the simulation to reality gap in this way can unlock the true potential of robot-assisted production using artificial intelligence.

Keywords: Synthetic data; Domain knowledge; Sim2Real gap; Object detection

1. Introduction

The revolution in computer graphic hardware industry in the last decade has given birth to countless new Artificial Intelligence (AI) applications. Some of these deep learning-based applications are object detection, segmentation and pose estimation. 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 boxes and pixels

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

1877-0509
10.1016/j.procs.2025.01.229

coordinates respectively. In case of object pose estimation, scene data containing 6D poses of all the objects in the images is needed which makes annotating real images extremely challenging [4, 35, 21]. Hence, for these applications, generating annotated synthetic data in an quick and easy way 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 few successful approaches have been proposed and validated for datasets containing household objects with substantial number of features. However, the components used in manufacturing are often textureless, reflective and colorless making them difficult to detect as compared to the objects with varying features [5, 22]. Objects trained in a generic environment tend not to work so well for specialized domains such as production environment [1]. This is why zero-shot object detection models are also not helpful here. For such scenarios, target domain knowledge is extremely essential to further bridge the simulation to reality gap [21, 12]. 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 and are difficult to handle. The individual parts after their integration into assemblies are partly occluded and go undetected. In addition to that, there is a possibility of detecting hundreds of unknown objects lying inside a production cell as false positives. For synthetic data generation, not all the domain randomization aspects can be combined in a single scene. For example, it must be decided whether a scene should have photorealistic textures or preloaded background images. In these cases, the generated dataset is a combined one without prior knowledge of their individual efficacies in real environments [34, 35, 24].

This paper introduces a domain-specific synthetic data generation pipeline designed to address various challenges in production facilities. Section 2 reviews the current state-of-the-art synthetic data generation methods, highlighting their benefits and limitations. In Section 3, a pipeline-based concept for data generation and its implementation is presented. The pipeline uses scripts to handle assembly CAD models and exports the parts as meshes, which are used later for generating images. The focus is on various data generation procedures that incorporate domain randomization and domain adaptation techniques. Section 4 evaluates the synthetic data generated by these techniques against real images within a production cell, with particular emphasis on assessing the impact of different methods on bridging the Sim2Real gap. Finally, Section 5 highlights the findings and outlines potential directions for future research.

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 for generating data have been 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 randomly or at specific realistic locations on background images after scaling and transforming them [7, 14]. Data can be generated cheaply in this way, but lacks 3D scene information due to which it is 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 [6]. 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 [4] and Tobin et al. in [34] point out a number of limitations in MuJoCo and Gazebo while generating 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. [35] and To et al. [33]. Eventhough the game engines can render synthetic images reasonably well, they are tuned to provide real-time performance and lack dynamic characteristics of photorealism [31, 23]. Finally, the last method consists of physically based rendering (PBR), a technique which can achieve a high level of photorealism. Cycles engine from Blender is an example of it. Mayershofer et al. in [21] conclude that a network trained on images generated with physics based rendering (Cycles) outperforms the one trained on images rendered by a game engine (EEVEE).

Normally, generation of synthetic data is 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 by using an application programming inerface (API). NDDS [33] project, based on Unreal Engine (UE4), was used to generate a dataset in [18]. BlenderProc [9] procedural pipeline based on the Cycles engine from Blender also showed improved Sim2Real performance [8]. A very similar project, NViSII [23], was demonstrated on NVIDIA's OptiX ray tracing

engine. A more recent pipeline Kubric [16] uses Blender and PyBullet to generate photorealistic data with ground truth.

Domain randomization and domain adaption techniques have been widely used to bridge Sim2Real gap [4, 26, 28, 34, 35]. While domain randomization is very robust to varying environment, domain adaption helps to achieve a higher precision across domains. Domain randomization generalizes the performance in real world applications. 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 [38, 3], or by using synthetic photorealistic images resembling to the target domain [17], or by using generative adversarial networks (GAN) [15] based methods which can be used to transform synthetic images to target domain [2, 29, 37, 25]. However, domain adaption techniques, often being tricky, require greater manual effort compared to domain randomization [4]. The use of photorealism together with domain randomization leads to higher confidence values and enables training models without freezing backbone layers [32, 8]. Some of the later projects have shown that combining domain randomization with target domain knowledge improves the overall detection performance [1, 21, 12].

As far as reducing Sim2Real gap in a manufacturing environment is considered, almost all research papers use domain adaption techniques along with domain randomization. These have been presented for different sectors such as autonomous driving, logistics and industrial production [13, 38, 21, 30, 10, 12, 3]. Dümmel et al. in [10] use Autodesk Inventor modelling software for data generation and 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 additionally to get realistic orientations of the parts. While Moonen et al. [22] and Baaz et al. [3] generate synthetic images based on rendering pipeline from Unity, Mayershofer et al. [21], Eversberg and Lambrecht [12] and Andulkar et al. [20] use Blender API for data generation, making the mentioned approaches scalable. [22] and [12] also show improvements for industrial use cases. Out of all the studies, only two works [21, 12] show a comparison of data generated with different parameters and their effect.

This work studies domain randomization and domain adaption procedures for industrial components, examining their effects in detail. It also proposes a pipeline capable of handling complex CAD models and exporting parts and assembly models to mesh format with ease. The data generation pipeline implemented here is scalable and built on the BlenderProc [9] procedural framework. This solution enables generation of thousands of images within a few hours using any desired combination of data generation procedures. The generated data can be utilized in robot-assisted manufacturing applications for tasks such as object classification, detection, segmentation and 6D pose estimation tasks without the need of real annotated data.

3. Synthetic Data Generation Pipeline

This section introduces pipeline-based approach for generating synthetic data in a domain-specific environment and the demonstrator setup used 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 [9] pipeline. A comparable framework, NViSII [23] can also be used as an alternative to the BlenderProc pipeline. The goal here is to generate training data from CAD models by reducing manual work and make 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 and use it for training multiple AI applications such as object classification, detection, segmentation and pose estimation.

The structure of the pipeline is depicted in Fig. 1. 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 relevant parts in mesh format for data generation.

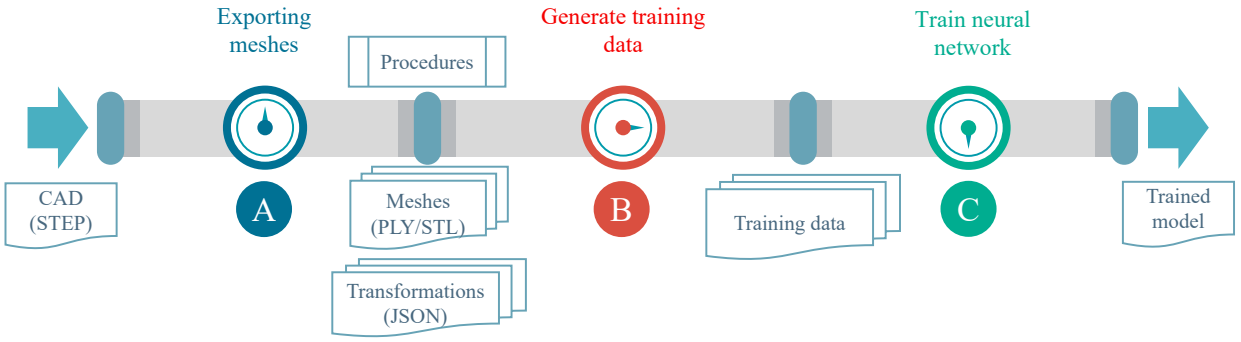

Fig. 1. Scalable pipeline concept for training models purely on synthetic data

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] [9]. 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 with CAD model.

In order to validate the generated data in a production-like environment, a special demonstrator is constructed. The demonstrator consisted of multiple parts which are simplified versions of components used in aviation for manufacturing passenger aircrafts. Fig. 2 shows an image of the validation demonstrator mounted on an industrial hand cart along with its parts and sub-assemblies. 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 synthetic data for industrial parts having a planar geometry and their overall effect on successful detections.

3.2. Exporting meshes

STEP file format is widely used for drafting components due to its accurate geometric representation. Mesh formats like STL and PLY are approximations of geometric models and are more popular in the field of computer vision. Hence, a conversion from STEP format to mesh formats is an essential step for data generation process.

Segregation of part files from an assembly CAD model containing thousands of parts can be cumbersome and labor intensive. This pipeline partly automates the mesh generation process from assembly CAD model using scripts. For this, the *FreeCAD* open-source tool which supports macros is used. The steps undertaken to export the target mesh files from a CAD model are depicted in Fig. 3. 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 which are 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 to be detected. In the final step, another macro automatically generates the mesh files for all the components mentioned in the `CategoryList.csv` and places them in the folder `Classes`. The components which are in the `PartList.csv` but not in `CategoryList.csv` are considered as passive objects and their meshes are exported in the 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 transformations of the individual components in the main assembly coordinate frame are exported in JSON format as text files in the folders

[1] https://www.freecad.org/
[2] https://github.com/DLR-RM/BlenderProc/

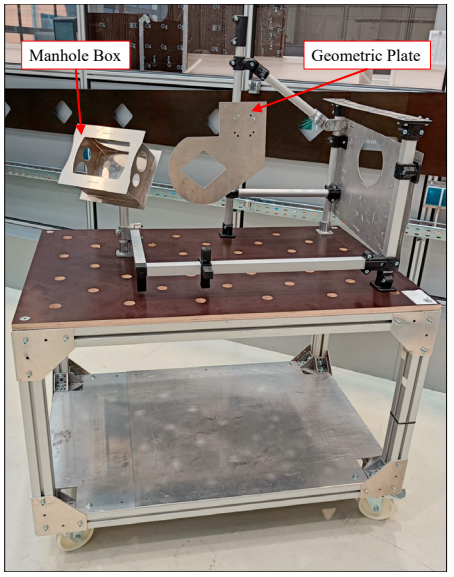

Fig. 2. Demonstrator used for validation of synthetic data in real environment

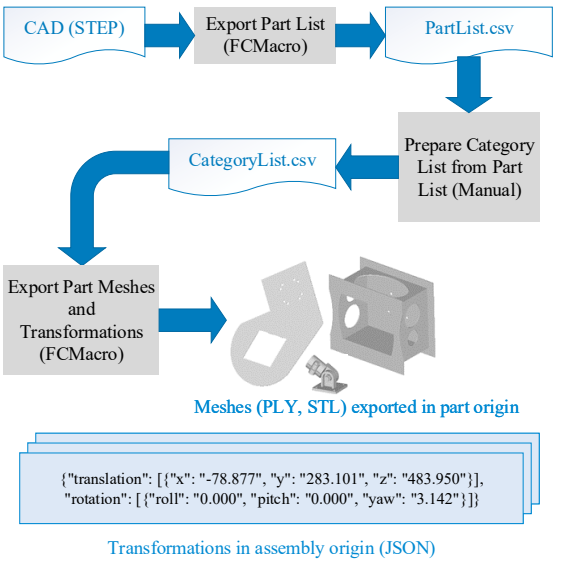

Fig. 3. Steps for exporting part meshes and their transformations in assembly origin

`Classes` and `Structure`. These transformations are a part of target domain knowledge and are later used to recreate the scene where the components can be placed at positions and orientations as defined in the main assembly model. Transformations are saved as translation vector in millimeter and as Euler angles in Roll-Pitch-Yaw convention as shown in Fig. 3.

The macro scripts used here are based on Python and have file extension `.FCMacro`. *FreeCAD API* is used 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.

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 considered similar to cooking recipes. Just like a cooked dish may vary based on the cooking recipe and ingredients followed, the synthetic images generated here also depend on the procedure used. These procedures include creating a 3D scene either 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 planar object in the form of circular plate when 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 random pose, it can possibly happen that the camera perspective only sees its flat edges and not its distinctive features. High losses while training will be observed for such cases. 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.

Fig. 4 provides a comprehensive explanation of the step by step data generation process for all procedures sequentially starting from top to bottom. 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

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	HDR background with objects on the ground
Procedure4	HDR background with objects in 3D space
Procedure5	HDR background and target scene reconstruction

of images, each procedure has an outer loop that orchestrates the process 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 they do not overlap. Procedure1 and Procedure3 use a physics-based approach, allowing the objects to fall on the ground plane.

Various objects within the scene possess surface appearance properties that are randomized, including but not limited to base color, roughness, specular, and metallic. Additionally, the scene incorporates textured planes or backgrounds based on a predetermined procedure. The size of these planes can be customized depending on the size of the objects, and they are further enhanced with randomly generated textures sourced from *CC0Textures*[3]. In case of procedures with backgrounds, only indoor HDR backgrounds are used from *PolyHaven*[4]. 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, color, 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 camera locations are sampled on the sphere as per the defined radius and the number of image samples to be captured in each 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. 5. 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 an inbuilt functionality in the BlenderProc [9] 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 [19] 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. In this way, the generated dataset can be utilized in various applications that employ deep learning techniques.

4. Validation

The datasets generated with all five procedures and their combinations were evaluated for object detection using YOLOv7 [36]. YOLOv7 implementation, with its evaluation scripts, enables quick and easy calculation of object detection metrics. For a fair comparison, all the configuration parameters and hyperparameters were kept unchanged.

[3] https://ambientcg.com/
[4] https://polyhaven.com/hdris

Fig. 4. Five basic data generation procedures explained in top-down approach

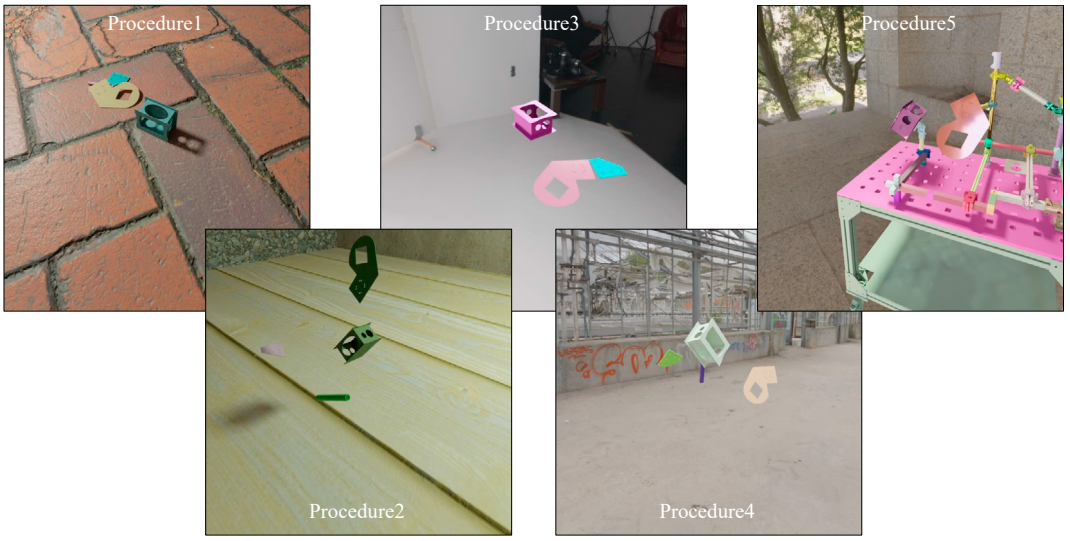

Fig. 5. Synthetic images generated from basic procedures

The normal YOLOv7 model mentioned in [36] 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 mixed procedures, the train test ratio was maintained for data generated by individual procedures. The time taken to generate a single dataset on a computer equipped with a NVIDIA RTX A4000 took between 9 to 12 hours depending on the procedure(s) used. Altogether, five basic procedures and five combination datasets were validated on 500 synthetic and 150 real images. The validation results on synthetic dataset are not elaborated in the paper due to space constraints but will be made available as supplementary material.

The real images for validation were captured with a camera (640 × 480 pixels) in a production environment and annotated manually. 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 [19] denoted as mAP@[0.5, 0.95] and PASCAL VOC's metric [11] 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 [27].

4.1. Evaluation of basic procedures

Table 2 draws a performance comparison for datasets generated with basic procedures for both the objects individually. Here, P1 to P5 indicate the dataset generated using Procedure1 till Procedure5. The models trained on synthetic dataset using any single procedure have best results if validated on the synthetic dataset created with the same procedure. However, 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. P2 consists of object images with textured planes without physics simulation where objects float randomly in 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 behavior can be explained from the geometry of the objects. *ManholeBox* having geometric features on all six sides can be detected easily when placed at a random 3D pose 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 the perspectives. However, 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. Objects lying on the ground plane resemble to deterministic behavior whereas the objects lying in 3D space resemble to 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 can be more helpful. Model trained on P4 dataset, which is rendering

Table 2. Performance comparison of basic procedure datasets

Procedure Datasets	Trained on									
mAP@[0.5:0.95]	*ManholeBox*					*GeometricPlate*				
Validated on	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
Real parts loose	0.25	0.70	0.35	0.46	0.31	0.70	0.03	0.34	0.52	0.00
Real assembly	0.00	0.51	0.00	0.58	0.74	0.72	0.70	0.30	0.86	0.24
Average	0.12	**0.61**	0.17	0.52	0.53	**0.71**	0.36	0.32	0.69	0.12

Note: Values in bold indicate the maximum average value among the five datasets.

with a random HDR background in 3D space, also shows good results for both objects. This is because backgrounds, even though not photorealistic, do contain features related to objects found in real world. Models trained on P5 dataset perform better for real images from assembly demonstrator. This is expected because it is trained on images by recreating the entire assembly model in the scene.

4.2. Evaluation of combination datasets

Based on the performance of the basic procedures on the real datasets, several different combination datasets were generated consisting of multiple procedures in different proportion. Out of them, five different combination datasets with good performance are chosen and compared here. Table 3 shows the composition of different combination datasets represented from C1 till C5. These datasets were used for training and were validated on synthetic and real images.

Table 3. Composition of combination datasets

	Basic procedures				
Combinations	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%

Table 4 draws a performance comparison between combination datasets for both the objects individually. The models trained on combination datasets showed better performance on synthetic images than the ones trained on basic procedures. For real images in production environment, models trained on C2, C3 and C4 perform better than the best performing model trained on basic procedure for *Manholebox* and in case of *GeometricPlate*, the ones trained on C4 and C5 perform better than the best performing model on basic procedure. A proper mixture of procedures can yield better performance by bridging the simulation to reality gap.

4.3. Final results and discussion

Combination datasets show varied overall performance. While the C2 and C4 combinations show a performance improvement of 11% and 15% respectively with respect to the best performing basic procedure, other combinations are not effective enough. Referring to Table 3 again, C2 is composed of three basic procedures: objects on the ground plane with textures, objects in 3D space with indoor HDR background and objects placed as defined in assembly CAD model with indoor HDR backgrounds. C4 is only composed of two basic procedures: objects in 3D space with indoor HDR background and objects placed as defined in assembly CAD model with indoor HDR backgrounds. Textured

Table 4. Performance comparison of combination datasets

Combination Datasets	Trained on									
mAP@[0.5:0.95]	*ManholeBox*					*GeometricPlate*				
Validated on	C1	C2	C3	C4	C5	C1	C2	C3	C4	C5
Real parts loose	0.51	0.61	0.52	0.47	0.41	0.06	0.36	0.18	0.49	0.68
Real assembly	0.27	0.78	0.72	0.85	0.20	0.86	0.94	0.82	0.96	0.81
Average	0.39	**0.69***	0.62*	0.66*	0.31	0.46	0.65	0.50	0.73*	**0.74***

Note: Values in bold indicate the maximum average value among the five datasets.
* Better results than the best performing basic procedure.

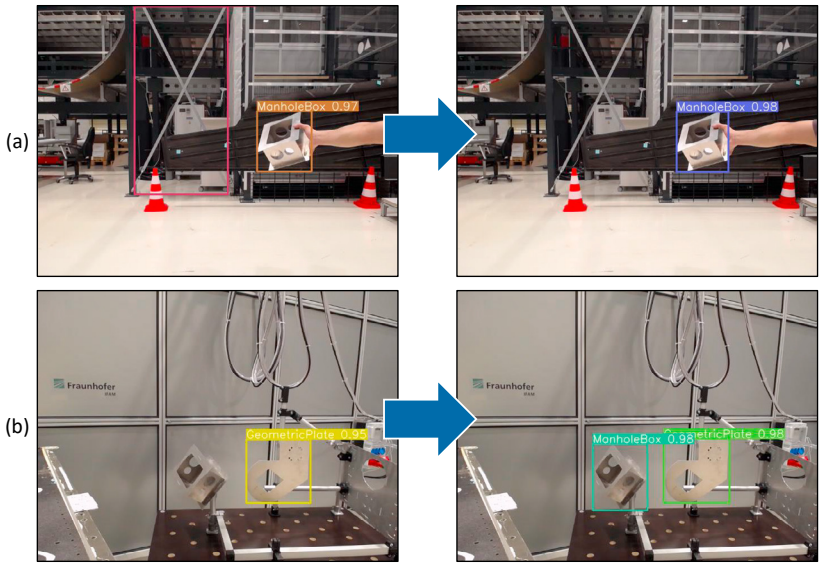

Fig. 6. Validation images for model trained on C2 dataset showing improved performance over model trained on basic procedure P4. Left images in (a) and (b) show a false positive and a false negative respectively which was handled correctly in the right images using model trained on C2.

planes introduce photorealistic characteristics such as reflection and shadows to the rendered images whereas HDR backgrounds introduce realistic objects in the scene reducing the false positives. The standalone performance of the target domain based P5 dataset using assembly CAD and HDR backgrounds is not satisfactory on real images but when this procedure is combined with other procedures, a performance boost is observed. Fig. 6 shows examples of improved performance achieved 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.

5. Conclusion

In this work, procedure-based generation of photorealistic synthetic data for domain-specific environments is presented. The pipeline approach presented here starts by 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 [9] and generates a synthetic dataset using a desired combination of basic procedures, which can then be used for training neural networks for tasks such as object detection and object pose estimation. These procedures generate a comprehensive dataset which incorporates the target domain and at the same time consists of enough variability. To measure the performance, YOLOv7 [36] models trained on five basic procedures are validated on real world images

inside a production environment and the results are used to generate combination datasets with multiple procedures. The models trained on combination datasets reduce the Sim2Real gap and improve the performance by 15% on real images. The drastic performance improvement achieved is due to the consideration of the target domain knowledge.

From the validation of the datasets, it can be concluded that the 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. Future work could involve exploring GAN-based image generation for comparison with current methods. Additionally, for a more realistic scenario, digital models like digital twins or digital shadows could enhance the system's adaptability and performance in real-world scenarios. These directions will broaden the applicability of the approach in various industrial settings.

Acknowledgements

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.

Funding and Declaration of Competing Interests

The present work is conducted with support of the Lower Saxony Ministry of Economic Affairs, Employment, Transport and Digitalization and the N-Bank following the project Skotty under grant ZW 1 80159842. The authors have no relevant financial or non-financial interests to disclose.

References

[1] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, S. Birchfield, 2019. Structured domain randomization: Bridging the reality gap by context-aware synthetic data, in: 2019 International Conference on Robotics and Automation (ICRA), pp. 7249–7255. doi:10.1109/ICRA.2019.8794443.

[2] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, R. Webb, 2017. Learning from simulated and unsupervised images through adversarial training, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2242–2251. doi:10.1109/CVPR.2017.241.

[3] August Baaz, Yonan Yonan, Kevin Hernandez-Diaz, Fernando Alonso-Fernandez, Felix Nilsson, 2024. Synthetic data for object classification in industrial applications, pp. 387–394. URL: https://www.scitepress.org/Link.aspx?doi=10.5220/0011689900003411.

[4] Borrego, J., Dehban, A., Figueiredo, R., Moreno, P., Bernardino, A., Santos-Victor, J., . Applying domain randomization to synthetic data for object category detection. URL: https://arxiv.org/pdf/1807.09834.

[5] C. A. Akar, J. Tekli, D. Jess, M. Khoury, M. Kamradt, M. Guthe, 2022. Synthetic object recognition dataset for industries, in: 2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 150–155. doi:10.1109/SIBGRAPI55357.2022.9991784.

[6] Collins, J., Chand, S., Vanderkop, A., Howard, D., 2021. A review of physics simulators for robotic applications. IEEE Access 9, 51416–51431. doi:10.1109/ACCESS.2021.3068769.

[7] D. Dwibedi, I. Misra, M. Hebert, 2017. Cut, paste and learn: Surprisingly easy synthesis for instance detection, in: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 1310–1319. doi:10.1109/ICCV.2017.146.

[8] Denninger, M., Sundermeyer, M., Winkelbauer, D., Olefir, D., Hodan, T., Zidan, Y., Elbadrawy, M., Knauer, M., Katam, H., Lodhi, A., 2020. Blenderproc: Reducing the reality gap with photorealistic rendering, in: International Conference on Robotics: Sciene and Systems, RSS 2020. URL: https://elib.dlr.de/139317/.

[9] Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., Katam, H., . Blenderproc. URL: https://arxiv.org/pdf/1911.01911.

[10] Dümmel, J., Kostik, V., Oellerich, J., 2021. Generating synthetic training data for assembly processes, Springer, Cham. pp. 119–128. URL: https://link.springer.com/chapter/10.1007/978-3-030-85910-7_13, doi:10.1007/978-3-030-85910-7{\textunderscore}13.

[11] Everingham, M., van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 2010. The pascal visual object classes (voc) challenge. International Journal of Computer Vision 88, 303–338. URL: https://link.springer.com/article/10.1007/s11263-009-0275-4, doi:10.1007/s11263-009-0275-4.

[12] Eversberg, L., Lambrecht, J., 2021. Generating images with physics-based rendering for an industrial object detection task: Realism versus domain randomization. Sensors 21, 7901. URL: https://www.mdpi.com/1424-8220/21/23/7901, doi:10.3390/s21237901.

[13] F. Reway, A. Hoffmann, D. Wachtel, W. Huber, A. Knoll, E. Ribeiro, 2020. 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), pp. 1249–1256. doi:10.1109/IV47402.2020.9304567.

[14] Georgakis, G., Mousavian, A., Berg, A., Kosecka, J., 2017. Synthesizing training data for object detection in indoor scenes, in: Amato, N. (Ed.), Robotics: Science and System XIII, Robotics Science and Systems Foundation, Berlin? doi:10.15607/RSS.2017.XIII.043.

[15] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. Advances in Neural Information Processing Systems 27.

[16] Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D.J., Gnanapragasam, D., Golemo, F., Herrmann, C., Kipf, T., Kundu, A., Lagun, D., Laradji, I., Liu, H.T., Meyer, H., Miao, Y., Nowrouzezahrai, D., Oztireli, C., Pot, E., Radwan, N., Rebain, D., Sabour, S., Sajjadi, M.S.M., Sela, M., Sitzmann, V., Stone, A., Sun, D., Vora, S., Wang, Z., Wu, T., Yi, K.M., Zhong, F., Tagliasacchi, A., 2022. Kubric: A scalable dataset generator, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, Piscataway, NJ. pp. 3739–3751. doi:10.1109/CVPR52688.2022.00373.

[17] J. Dümmel, X. Gao, 2021. Object re-identification with synthetic training data in industrial environments, in: 2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), pp. 504–508. doi:10.1109/M2VIP49856.2021.9665094.

[18] J. Tremblay, T. To, S. Birchfield, 2018. 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), pp. 2119–21193. doi:10.1109/CVPRW.2018.00275.

[19] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context, in: Fleet, D. (Ed.), Computer vision - ECCV 2014, Springer, Cham. pp. 740–755. URL: https://link.springer.com/chapter/10.1007/978-3-319-10602-1_48, doi:10.1007/978-3-319-10602-1{\textunderscore}48.

[20] M. Andulkar, J. Hodapp, T. Reichling, M. Reichenbach, U. Berger, 2018. Training cnns from synthetic data for part handling in industrial environments, in: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), pp. 624–629. doi:10.1109/COASE.2018.8560470.

[21] Mayershofer, C., Ge, T., Fottner, J., 2021. Towards fully-synthetic training for industrial applications, in: LISS 2020. Springer, Singapore, pp. 765–782. URL: https://link.springer.com/chapter/10.1007/978-981-33-4359-7_53, doi:10.1007/978-981-33-4359-7{\textunderscore}53.

[22] Moonen, S., Vanherle, B., de Hoog, J., Bourgana, T., Bey-Temsamani, A., Michiels, N., 2023. Cad2render: A modular toolkit for gpu-accelerated photorealistic synthetic data generation for the manufacturing industry, in: 2023 IEEE Winter Conference on Applications of Computer Vision workshops, IEEE, Piscataway, NJ. pp. 583–592. doi:10.1109/WACVW58289.2023.00065.

[23] Morrical, N., Tremblay, J., Lin, Y., Tyree, S., Birchfield, S., Pascucci, V., Wald, I., . Nvisii: A scriptable tool for photorealistic image generation. URL: https://arxiv.org/pdf/2105.13962.

[24] Nowruzi, F.E., Kapoor, P., Kolhatkar, D., Hassanat, F.A., Laganiere, R., Rebut, J., . How much real data do we actually need: Analyzing object detection performance using synthetic and real data. URL: https://arxiv.org/pdf/1907.07061.

[25] P. Rojtberg, T. Pöllabauer, A. Kuijper, 2020. 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), pp. 188–195. doi:10.1109/AIVR50618.2020.00039.

[26] Ren, X., Luo, J., Solowjow, E., Ojea, J.A., Gupta, A., Tamar, A., Abbeel, P., 2019. Domain randomization for active pose estimation, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, Piscataway, NJ. pp. 7228–7234. doi:10.1109/ICRA.2019.8794126.

[27] S. Bell, C. L. Zitnick, K. Bala, R. Girshick, 2017. 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), pp. 2874–2883. doi:10.1109/CVPR.2016.314.

[28] S. Hinterstoisser, O. Pauly, H. Heibel, M. Martina, M. Bokeloh, 2019. 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), pp. 2787–2796. doi:10.1109/ICCVW.2019.00340.

[29] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, R. Chellappa, 2018. Learning from synthetic data: Addressing domain shift for semantic segmentation, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3752–3761. doi:10.1109/CVPR.2018.00395.

[30] Schoepflin, D., Iyer, K., Gomse, M., Schüppstuhl, T., 2022. 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, pp. 325–336. URL: https://link.springer.com/chapter/10.1007/978-3-030-74032-0_27, doi:10.1007/978-3-030-74032-0{\textunderscore}27.

[31] Šmíd, A., . Comparison of unity and unreal engine. URL: https://core.ac.uk/download/pdf/84832291.pdf.

[32] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. N. Sinha, B. Guenter, 2019. Photorealistic image synthesis for object instance detection, in: 2019 IEEE International Conference on Image Processing (ICIP), pp. 66–70. doi:10.1109/ICIP.2019.8803821.

[33] To, T., Tremblay, J., McKay, D., Yamaguchi, Y., Leung, K., Balanon, A., Cheng, J., Hodge, W., Birchfield, S., . Ndds: Nvidia deep learning dataset synthesizer. URL: https://github.com/NVIDIA/Dataset_Synthesizer.

[34] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P., 2017. Domain randomization for transferring deep neural networks from simulation to the real world, in: IROS Vancouver 2017, IEEE, Piscataway, NJ. pp. 23–30. doi:10.1109/IROS.2017.8202133.

[35] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., Birchfield, S., 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition workshops, IEEE, Piscataway, NJ. pp. 1082–10828. doi:10.1109/CVPRW.2018.00143.

[36] Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M., 2023. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, in: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, Piscataway, NJ. pp. 7464–7475. doi:10.1109/CVPR52729.2023.00721.

[37] X. Peng, K. Saenko, 2018. Synthetic to real adaptation with generative correlation alignment networks, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1982–1991. doi:10.1109/WACV.2018.00219.

[38] Y. Huangfu, W. Deng, B. Ren, J. Ding, 2021. 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), pp. 1–6. doi:10.1109/CVCI54083.2021.9661221.