https://doi.org/10.2352/EI.2024.36.6.IRIACV-276

Synthetic Data Generation for AI-based Machine Vision Applications

Frederik Seiler, Verena Eichinger, Ira Effenberger
Fraunhofer IPA, Stuttgart, Germany

Abstract

This paper presents a method for synthesizing 2D and 3D sensor data for various machine vision tasks. Depending on the task, different processing steps can be applied to a 3D model of an object. For object detection, segmentation and pose estimation, random object arrangements are generated automatically. In addition, objects can be virtually deformed in order to create realistic images of non-rigid objects. For automatic visual inspection, synthetic defects are introduced into the objects. Thus sensor-realistic datasets with typical object defects for quality control applications can be created, even in the absence of defective parts. The simulation of realistic images uses physically based rendering techniques. Material properties and different lighting situations are taken into account in the 3D models. The resulting tuples of 2D images and their ground truth annotations can be used to train a machine learning model, which is subsequently applied to real data. In order to minimize the reality gap, a random parameter set is selected for each image, resulting in images with high variety. Considering the use cases damage detection and object detection, it has been shown that a machine learning model trained only on synthetic data can also achieve very good results on real data.

1. Introduction

Solving machine vision tasks, e.g. automated part inspection or object recognition, with machine learning based approaches requires large and representative datasets. Since optical data acquisition and annotation is typically laborious the use of synthetic data might be a promising measure in order to reduce this effort and save time and costs. The main motivation for using synthetic data is twofold. First, in visual inspection tasks, there is often a lack of defective parts and consequently a lack of data with defects for training datasets, making it difficult to create robust machine learning models. By introducing synthetic defects into the 3D model of the object, the method presented in the paper allows the generation of datasets for quality control even in the absence of defective parts. Second, manually annotating large datasets for machine learning can be a time-consuming and repetitive task that often leads to inconsistencies in the annotated dataset.

Rendering techniques have made significant advances in recent years, enabling the creation of more realistic and immersive experiences. However, not only the entertainment industry benefits from these advances, they offer also great potential for other fields. Thus the technology behind synthetic data generation has strongly profited by these rendering techniques. The objective is to create a synthetic dataset that can be used for training a machine learning model capable of being deployed on real-world camera images or 3D sensor data.

By using virtual scenes and automatically generating ground truth data, the method can significantly reduce the amount of manual effort required for annotation, enabling more efficient and scalable data generation. One major challenge using this technique is the domain gap that exists between real-world images and synthetic data. Models trained only on synthetic data tend to experience a significant decrease in performance when applied to real-world camera images. To overcome this problem, synthetic data is modelled very accurately and a high degree of diversity is introduced into the synthetic data. We demonstrate the effectiveness of our method on two real world scenarios, showing that a machine learning model trained only on synthetic data can achieve very good results on real data.

Figure 1. *Deformation with Lattice*

2. Related Work

In recent years, several pipelines have been introduced to automatically generate synthetic datasets by simulating random arrangements of objects. These pipelines are usually built on top of 3D graphics software like Blender [1], [2] or Unity3D [3]. Within these frameworks a multitude of solid object instances are produced. Subsequently, these objects are subjected to a rigid body simulation that accurately portrays their falling behaviour, mimicking the natural physics of movement and collision. This simulation ensures that a diverse, representative, large and unbiased range of scenarios is generated. The resulting data is valuable for machine vision tasks as object detection and bin picking. One limitation of the existing pipelines for synthetic data generation is their restriction to rigid objects. Deformations are not considered, which results in an unrealistic simulation of deformable objects. In addition to the simulated images, depth maps, object poses, and semantic segmentation are generated automatically. However, the data lacks information on the graspability of the objects in the scene.

Instead of rigid body simulation Raistrick et al. [4] present the Infinigen framework for generating procedural scenes of the natural world. The generated images exhibit high variety, but are not photo realistic. Furthermore no experiments with real datasets were performed.

Another application focuses on the generation of synthetic images for visual inspection and object detection. Napier et al. [5] demonstrate how a network for segmentation can be trained solely on procedurally generated synthetic data. Due to a bias in the synthetic data, only a portion of the real test data yields good results. Schmedemann et al. [6] demonstrate that procedural defect generation can be used to augment a small real-world dataset. Here the combination of synthetic and real data yields better results than using only real data. Results based solely on synthetic data are not considered in the paper.

Previous works identified the domain gap as a common challenge in the usage of synthetic training data for machine vision tasks [7]. Models typically assume that training and test data are very similar. However, synthetic data differ from real data. Various techniques are elaborated in order to facilitate the transfer between synthetic training data and real test data. There are mainly three fundamental strategies:

1. Domain Adaptation: The aim of Domain Adaptation is to reduce the statistical deviation between the source and target domains [8]. Adaptation can occur at the input level, for example, by using a certain loss function, such as Adversarial Loss.
2. Sensor-Realistic Rendering: This approach strives for photorealism in the generated images. This is achieved by realistic object geometry, physically correct positioning of objects in a scene, precise material models, and lighting. An important technique for this approach is Physically Based Rendering (PBR) [9].
3. Domain Randomization: Developed by Tobin et al. [10], this concept involves randomizing synthetic data in such a way that the domain of real images becomes a subset of the synthetic domain. In other words, for a model trained on synthetic data, real data appear as just one of the variations in synthetic data. Domain Randomization allows the use of less realistic renderings, which require less effort to create. In this technique, parameters for rendering data are randomly varied. Examples of such parameters include textures, lighting settings, size and position of objects, object shape, and backgrounds. However, the structure should still reflect the context of the real data.

3. Method

Our method is based on the open source software Blender [11] and follows the traditional processing pipeline for renderings from 3D data. Initially, a model is created for the required object. Alternatively, existing CAD data can be utilized. In addition a UV map is generated for the object, which can be used in the shading process. In shading, all parameters and textures for the object's material are defined. Subsequently, a 3D scene is built based on the real measurement set-up with the simulation of light and sensors. In this scene additional processing steps such as the simulation of deformation, defects, or random arrangements of objects are carried out. Finally, the Cycles renderer is applied for physically based rendering of the scene. The shading process and processing steps will be outlined in the following sections.

If the basic 3D scene is prepared, our method will allow the user to define parameters and set thresholds for all further processing steps and the image generation. The user can specify parameters regarding all parts of the 3D scene like scene arrangement, lighting, shading, deformation, defects and rendering. Based on the user settings scene variation, rendering and processing are fully automated in order to generate large datasets with little effort. Furthermore, chosen annotations are determined automatically during the rendering process.

3.1 Shading

Real world objects often have a high variety in appearance. This complicates the shading process, as at some point one shader cannot cover all possible variations. This is also observed in the washers utilized for examining defect generation. Therefore, four basic shaders were defined for the washers in order to represent extreme cases in variety. Figure 2 shows the selected reference images in the upper row and examples for the four shaders in the lower row.

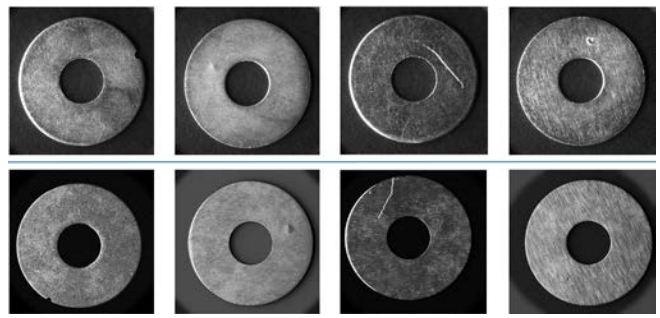

Figure 2. *Comparison of real camera images (top row) and synthetically generated images (bottom row). It is obvious that the differences between the real and synthetic data are minimal. Furthermore, the diversity present in the real data is also represented in the synthetic data.*

The majority of the appearance is defined by surface structures, which can be reproduced by adding a normal map to the shader. For the washer the shader is composed of various structures and scratch patterns. The basic structure is given by a noise texture. Different scratches are generated by combining noise, wave, and voronoi textures, as shown in Figure 3. The combination of these structures in colour and normal maps allows modelling complex surface appearances. As all textures are generated procedurally an automatic variation in size, intensity and pattern enables the creation of highly varied synthetic data.

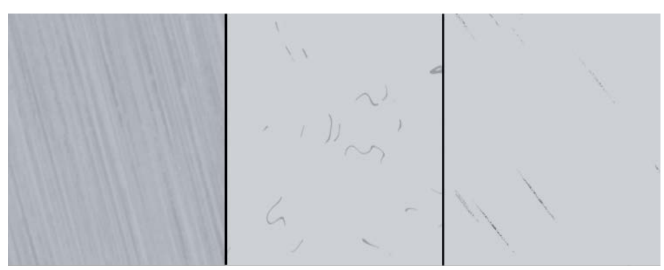

Figure 3. *Basic structure of the washer shader (left) and different scratch patterns (middle and right).*

3.2 Deformations

If the objects are not rigid, random deformation of the objects can be used to generate realistic data. In this section, two methods for implementing random deformations on object models are introduced. For automated deformations a pipeline using the described methods is implemented as one step in the automated data generation process.

3.2.1 Cage Deformations

Cage deformation, also known as lattice deformation, is a method that utilizes a grid structure enveloping the object to facilitate deformation. This grid comprises control points, which can be manipulated freely for achieving the desired transformation. The alterations performed on these control points are then propagated to the vertices of the object, effectively changing its shape. The extent to which each control point influences a vertex is proportional to their relative distance; vertices closer to a control point will be more affected by its transformation. This technique allows for a more intuitive and flexible approach to object deformation. The process is illustrated in Figure 1. By randomly selecting control points and applying random transformations to them, all deformed objects are unique, allowing for the automated generation of a diverse dataset.

3.2.2 Bending and Twisting

The bending and twisting approach for object deformation is defined by setting an origin, axis, and rotation angle to manipulate the shape of an object within its local coordinate space. This deformation can be selectively applied to certain parts of the object or can comprise the entire object for a more uniform transformation. The process involves rotating vertices based on these specifications, where the proximity of each vertex to the origin determines the degree of its rotation. For twisting, vertices are rotated along the predefined axis, while bending induces a rotational deformation over a given axis, altering the object's structure accordingly. Examples of bent and twisted objects are depicted in Figure 4.

Figure 4. *The original object model is bent (left and middle) and twisted (right).*

3.3 Random Arrangements

The method of creating random arrangements utilizes the Blender Physics Engine. This process involves generating multiple copies of a solid object and then applying a rigid body simulation in order to depict their falling behaviour accurately. The simulation is designed for altering the objects' position and orientation without causing any deformation, as deformations are accounted for in a previous step. In the context of these simulated environments, objects, that can be grasped, are identified by determining whether they are obscured by another object or not. This is a key step facilitating the creation of datasets suitable for bin picking applications.

3.4 Defect Generation

For automatic visual inspection, a defect generator is implemented that can apply defects to 3D models automatically, by modifying the object’s geometry. This procedure has been demonstrated for defect classes such as scratches, dents, bumps, and notches. Each of these defects is created at a random location and with a random appearance.

3.4.1 Bumps and Scratches

Displacement Maps enable simulating bumps and scratches on a surface, providing a one-dimensional translation of vertices along the normal vector. Thus, a grayscale image encodes the degree of displacement for each vertex through its intensity values. For applying these maps onto a 3D object's surface, it is necessary to establish a UV-Layout that translates the object’s three-dimensional surface into a two-dimensional representation.

For simulating bumps, the displacement maps are characterized by a Gaussian intensity profile, which models the 3D topology in order to mimic the appearance of authentic bumps found in the real world. Displacement Maps for scratches are crafted with procedural generation techniques contributing to a considerable variability in their shape, size, and depth. For the accurate creation of the fine details of scratches, a high-resolution mesh is indispensable, ensuring that the nuances of each scratch are realistically portrayed.

Figure 5. *Synthetic scene of a random arrangement of syringes in a box.*

3.4.2 Dents and Notches

An "auxiliary object" is created which acts as a negative-mold for deformation purposes of dents and notches. This auxiliary object is subject to random deformation, translation and scaling processes, which are key steps for enhancing the diversity of the resulting image set. Subsequently, a Boolean operator is applied to the original and auxiliary objects performing a subtractive intersection. This intersection carves out the auxiliary object’s shape from the original object resulting in the deformed original object. Figure 6 shows the simplified representation of this method for generating notches.

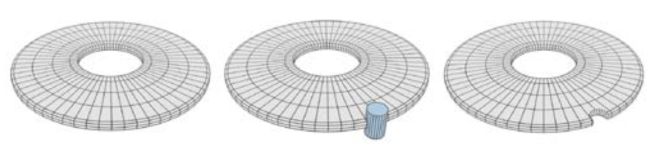

Figure 6. *Simplified generation process for notches using a Boolean operator.*

4. Experiments and Results

The utility of synthetic data in real-world machine vision applications is shown in two different use cases by applying the presented methods and process steps. The two use cases originate from the field of visual inspection and object recognition.

4.1 Object Detection

The application of object detection is demonstrated using a bin picking scenario. The aim is to detect plastic syringes that are graspable for a robot, meaning they are not occluded by other syringes. Utilizing the method presented in Section 3.3 for generating random arrangements, 2000 simulated scenes of syringes in a box are automatically created. An example of a synthetic scene is shown in Figure 5. A state-of-the-art object detector is trained to identify the graspable syringes. The neural network selected for this purpose is EfficientDet [12]. The network, trained on synthetic data, is tested on six real-world images. This testing achieves a precision of 96% and a recall of 93%. The successful detection rates demonstrate the efficiency of the neural network model in distinguishing graspable syringes. The visualization of a test image with the model's detections is shown in Figure 7.

4.2 Visual Inspection

Defective metal washers with bumps, dents, scratches and notches are regarded in order to demonstrate the presented procedure. For the test dataset, 40 images are captured for each defect class. For the training dataset, 500 synthetic images are generated for each defect class. These images are used for training EfficientDet as a state-of-the-art object detector. The trained model is tested on real data. For performance comparison, additional models are trained on a completely real dataset and a combined dataset respectively.

***Table 1**. Overview Results of the visual inspection*

		Precision	Recall
Real Data	Bump	1.0	0.971
	Dent	1.0	0.971
	Notch	1.0	0.971
	Scratch	0.950	0.543
	Total	0.992	0.864
Synthetic Data	*Bump*	1.0	0.914
	Dent	0.909	0.857
	Notch	1.0	0.914
	Scratch	0.939	0.886
	Total	0.962	0.893
Combined Data	*Bump*	0.972	1.0
	Dent	1.0	1.0
	Notch	1.0	1.0
	Scratch	0.972	1.0
	Total	0.986	1.0

With a precision of 99% and a recall of 100% achieved, the model based on mixed data performs best. The combination of synthetic and real data potentially leads to more robust features for the classes and can thus justify the improvement in the mixed data. The performance of the model based on synthetic data (precision: 96%, recall: 89%) hardly differs from the result with real data (precision: 99%, recall: 86%). The models can be considered equivalent. In visual inspection, smaller real datasets can thus be enriched with synthetic data for a robust error detection. Table 1 provides a summary of the outcomes using real, synthetic, and combined training data.

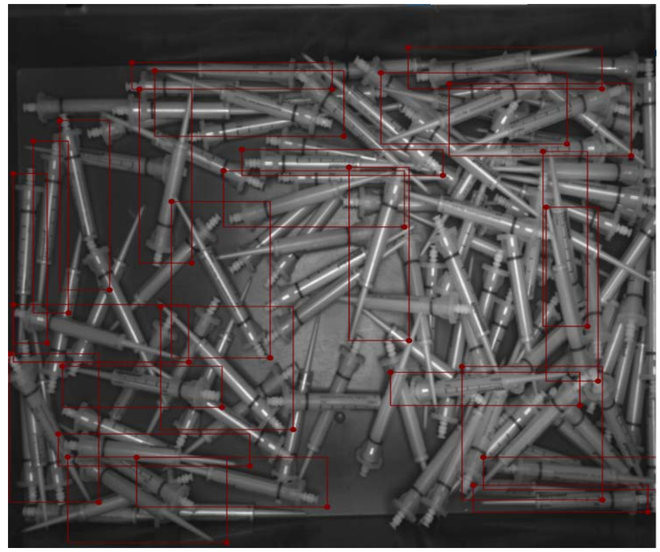

***Figure 7.** Camera image of a random arrangement of syringes in a box. The detections from an AI-based object detector, trained solely on synthetic data, are depicted with red bounding boxes.*

5. Conclusion

This paper describes a method for generating realistic and diverse synthetic datasets. The main focus is on the realism of the data through realistic shaders and the simulation of defects, deformations and physically based random arrangements. The variability of data is achieved through flexible parameter definitions adapted to the target application. The process of data generation is automated by processing a pipeline with components for deformation, defect generation, scene arrangement and rendering. In two practical use cases the efficiency and applicability of the method is demonstrated. Synthetic data generated for object recognition as well as for visual inspection are successfully used for training neural networks. The trained models are able to achieve accurate and reliable results when applied to real camera images. Accordingly, synthetic data can be utilized as an alternative to real data when there is a lack of data. Besides, it can be used when real data acquisition is too time consuming or real data sets need to be enriched in order to achieve more robust AI models.

References

[1] M. Denninger et al., "BlenderProc2: A Procedural Pipeline for Photorealistic Rendering", Journal of Open Source Software, vol. 8, no. 82, p. 4901, Feb. 2023.

[2] K. Greff et al., "Kubric: A scalable dataset generator", Mar. 07, 2022. doi: 10.48550/arXiv.2203.03570.

[3] Unity Technologies, "Unity Perception Package", https://github.com/Unity-Technologies/com.unity.perception, 2020.

[4] A. Raistrick et al., "Infinite Photorealistic Worlds using Procedural Generation", arXiv, Jun. 26, 2023. doi: 10.48550/arXiv.2306.09310.

[5] C. C. Napier, D. M. Cook, L. Armstrong, and D. Diepeveen, "A Synthetic Wheat L-System to Accurately Detect and Visualise Wheat Head Anomalies", in 3rd International Conference on Smart and Innovative Agriculture (ICoSIA 2022), Atlantis Press, May 2023.

[6] O. Schmedemann, M. Baaß, D. Schoepflin, and T. Schüppstuhl, "Procedural synthetic training data generation for AI-based defect detection in industrial surface inspection", Procedia CIRP, vol. 107, pp. 1101–1106, Jan. 2022.

[7] S. I. Nikolenko, "Synthetic Data for Deep Learning", arXiv, Sep. 25, 2019.

[8] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in 2011 International Conference on Computer Vision, Barcelona, Spain, 2011, pp. 999-1006. doi: 10.1109/ICCV.2011.6126344.

[9] M. Pharr, W. Jakob and G. Humphreys, Physically Based Rendering: From Theory To Implementation, 3rd ed. Burlington: Morgan Kaufmann, 2016.

[10] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World", Mar. 20, 2017.

[11] Blender Foundation, "Blender," 3.8, Available: https://www.blender.org/.

[12] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," CoRR, vol. abs/1911.09070, 2019. Available: http://arxiv.org/abs/1911.09070

Author Biography

Frederik Seiler studied electrical engineering at the Karlsruhe Institute of Technology (KIT) in Germany and has been a Research Associate at Fraunhofer Institute for Manufacturing Engineering and Automation in Stuttgart since 2021, focusing on Machine Vision and Signal Processing. The author's work concentrates on leveraging synthetic data in industrial applications.

Verena Eichinger studied Computer Science and Media at the Stuttgart Media University. Since 2023, she is a Research Associate at the Fraunhofer Institute for Manufacturing Engineering and Automation in Stuttgart. The author's work lies in the fields of Machine Vision and Signal Processing with a focus on the generation of synthetic data for AI models and their application.

Dr Ira Effenberger is research group manager and deputy head of the department Machine Vision and Signal Processing at Fraunhofer IPA. In her research group the main focus lies on synthetic data generation and intelligent sensor data analysis for machine vision applications.