A Scalable Synthetic Data Creation Pipeline for AI-Based Automated Optical Quality Control

Christian Schorr[1,2], Sebastian Hocke[1], Tobias Masiak[3] and Patrick Trampert[3]
[1]German Research Centre Artificial Intelligence, Saarland Informatics Campus D 3 2, 66123 Saarbrücken, Germany
[2]University of Applied Sciences Kaiserslautern, Amerikastraße 1, 66482 Zweibrücken, Germany
[3]Artificial Intelligence Lab, ZF Friedrichshafen AG, Scheer Tower II, Uni-Campus Nord, D5 2, 66123 Saarbrücken, Germany

Keywords: Synthetic Data, Digital Reality, Digital Twin, Defect Detection, Deep Learning, Quality Control, Computer Vision, Visual Inspection, Simulation, Rendering.

Abstract: In recent years, the industry's interest in improving its production efficiency with AI algorithms has grown rapidly. Especially advancement in computer vision seem promising for visual quality inspection. However. the proper classification or detection of defects in manufactured parts based on images or recordings requires large amounts of annotated data, ideally containing every possible occurring defect of the manufactured part. Since some defects only appear rarely in production, sufficient data collection takes a lot of time and may lead to a waste of resources. In this work we introduce a configurable, reusable, and scalable 3D rendering pipeline based on a digital reality concept for generating highly realistic annotated image data. We combine various modelling concepts and rendering techniques and evaluate their use and practicability for industrial purposes by customizing our pipeline for a real-world industrial use case. The generated synthetic data is used in different combinations with real images to train a deep learning model for defect prediction. The results show that using synthetic data is a promising approach for AI-based automated quality control.

1 INTRODUCTION

Recent advancements in visual computing have revolutionized how deep learning algorithms can be used for industrial purposes. Especially automating tasks like quality control can drastically improve the efficacy and speed of production (J. Li et al., 2018; S. Li & Wang, 2022; Wang et al., 2021). However, introducing these kinds of models into active production environments comes with its own set of challenges.

One of the biggest issues of deep learning for quality inspection in industrial environments is the availability of annotated training data. There are many different types of defects that can occur on machine manufactured parts that need to be classified to ensure proper quality of the produced good. While some of these defects will occur more regularly in the production environment, there are many others that will only appear on rare occasions. It is almost impossible to gather sufficient training data for every possible defect shape, without spending a considerable amount of time collecting and annotating or wasting resources by recreating known malformations on actual production parts.

Another aspect is the extensive annotation process that is necessary to generate training datasets containing similar amounts of data for each defect to ensure class balance. Overall: the creation of a good training dataset, which produces satisfactory results when used for training deep learning algorithms, is a costly and time-consuming undertaking.

A promising approach to eliminate these obstacles is the generation of synthetic image data using a digital reality concept (Dahmen et al., 2019). Some common methods for the creation of synthetic image data include generative adversarial networks (Goodfellow et al., 2014) or using 3D models and rendering techniques. Rendered annotated synthetic images of production parts have already been used to improve the performance of an algorithm for the semantic segmentation of defects on a gear fork (Gutierrez et al., 2021).

In this paper we will further elaborate on the process of generating 3D rendered image data and develop a generalizable and scalable pipeline to

Schorr, C., Hocke, S., Masiak, T. and Trampert, P.
A Scalable Synthetic Data Creation Pipeline for AI-Based Automated Optical Quality Control.
DOI: 10.5220/0012717400003758

In *Proceedings of the 14th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2024)*, pages 37-46
ISBN: 978-989-758-708-5; ISSN: 2184-2841

simplify the process. This will be demonstrated by using the specific use case of a diaphragm spring provided by the automotive supplier ZF Friedrichshafen AG.

2 STATE OF THE ART

The automatization of quality inspection tasks during the manufacturing process is typically done by analysing images of the product using computer vision and deep learning algorithms (Bhatt et al., 2021). Common techniques include classification or localisation of defects on images from the production environment. This can be achieved, among other methods, by training Convolutional Neural Networks for object detection like YOLO (Redmon et al., 2016) or R-CNN (Ren et al., 2017) but also segmentation approaches such as U-NET (Ronneberger et al., 2015) or Mask-R-CNN (He et al., 2020). However, deep learning algorithms also face some major problems, such as lack of sufficient and diverse data and adaption to new and unseen environments, affecting the generalization performance of the model (Mazzetto et al., 2020). Due to the low probability of occurrence for some anomalies, generating a balanced dataset containing vast amounts of image data for each defect is nearly impossible (Jain et al., 2022).

To address these issues simulated data has been studied for the use of pedestrian detection (Fabbri et al., 2021), general object detection (Tremblay et al., 2018), pose estimation (Chen et al., 2016) and many other use cases, see (Schraml, 2019) and (Mumuni et al., 2024) for further examples. Synthesizing image data with 3D modelling tools enables the creation of precisely controlled datasets with automatically generated error-free annotations (de Melo et al., 2022). The use of photorealistic synthetic images to train semantic segmentation algorithms has also been successfully applied to the visual inspection of production parts (Gutierrez et al., 2021). Increased realism of the synthetic training images has been shown to have a significant impact on the performance of Convolutional Neural Networks, especially in combination with domain randomization.

Domain randomization (Tobin et al., 2017) is a technique that randomizes specific conditions inside the simulation by randomizing the parameters, such as colour, texture, position, and orientation, during the data generation process. Borrego et al achieved significant performance improvements applying this technique by finetuning a model with synthetic image data for multi-category object detection (Borrego et al., 2018).

By exposing the model to a wide variation of data and scenarios we aim to improve the domain adaption of the deep learning model and prepare it for unexpected situations in the manufacturing environment.

The goal of domain adaption (Lee et al., 2020) is to reduce the gap between real and synthetic data. This is especially important when using different domains for creating the model, such as using a synthetic dataset for the training process and generalizing the model on a real dataset. The generation of a highly realistic synthetic dataset can be achieved with 3D modelling software or rendering solutions using physically accurate and GPU-accelerated rendering solutions, such as Maxwell Render 5, which is built on the Nvidia CUDA computing platform (*Maxwell Documentation*, 2024).

The generation of the synthetic data itself follows the concept of generative partial modelling, a technique that simplifies the analysis of complex systems by focusing on a specific set of features from the real world. In our case this set of features consists of the shape of the manufactured part, its location and rotation in the production environment, the textures including potential defects, as well as environmental properties that affect the object, like lighting or the camera setup.

3 MATERIALS AND METHODS

3.1 Digital Reality Concept

We base our simulation pipeline on the digital reality concept (Dahmen et al., 2019). The idea behind this concept is to build parametrizable partial models governing geometry, textures, lighting, camera settings and defects among others, which can be combined in a modular way into a generative scenario describing all aspects of the object and its environment we want to simulate (figure 1). This scenario model acts as a digital twin that can be sampled using different parametrizations of the partial models to obtain an arbitrary amount of rendered training images with the desired properties for subsequent AI model training. Partial models we want to describe in greater detail in this section are the geometric model of the object's shape (section 3.2), its surface texture model (section 3.3) and the defect model of possible manufacturing faults (section 3.4).

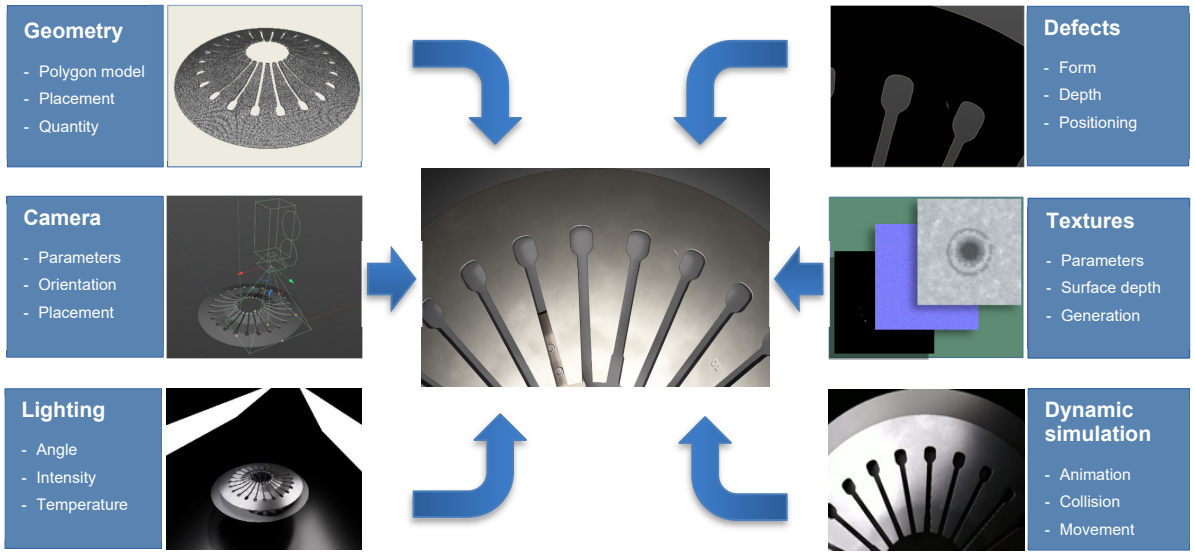

Figure 1: Digital reality concept diaphragm spring.

For the digital twin we use techniques like texture and geometry parameterisation, procedural texture synthesis, parametric damage modelling and characterization, as well as deep learning for defect detection to evaluate the quality of the synthesised dataset. The following sections describe the development and implementation of these techniques based on creating a digital twin of a diaphragm spring provided by ZF Friedrichshafen AG. With the help of this production part, we design our synthetic data creation pipeline to be scalable and usable for different real-world production scenarios.

3.2 Geometry Modelling

The 3D model of the digital twin of the production environment is realized in Maxon Cinema 4D R21. The simulation environment is created based on images and information about the size and position of the object it contains. The position of the lighting and other influences that affect the component to be simulated are particularly important to be able to recreate a representation as physically accurate as possible. The object itself is then placed in this digital twin of the production environment and is properly aligned based on real images. Figure 2 shows an exemplary placement and rendering of the diaphragm spring inside the Cinema 4D viewport.

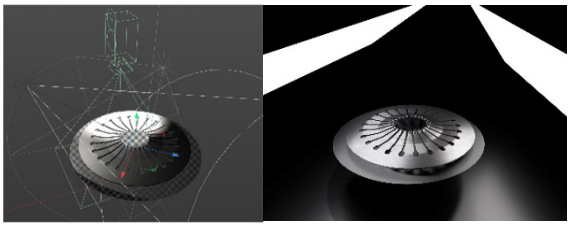

Figure 2: Cinema 4D scene and viewport rendering.

We use the Cinema 4D dynamic simulation feature for the correct positioning of the object to avoid polygon overlapping. For the simulation and modelling of realistic surfaces for the diaphragm spring, we use various texture mapping techniques as part of the automation process to represent complex and customisable materials in the synthetic environment. To compute these textures in a reproducible and automatable process, we use tools and algorithms available in the software Substance Designer 3D from Adobe.

3.3 Defect Modelling

The creation of realistic surface defects requires a precise characterisation of all damage types that could occur in the production. Ideally, a material scientific model for the defect formation and shape is available. In real life, this is almost never the case, so we must manage with images of real defects only, which again are often only available in a very limited quantity. For our use case, we use a defect catalogue provided from ZF Friedrichshafen AG, containing images of real defects from the production line, as well as several physical diaphragm springs to gain an understanding of the different defect forms. These can range from scratches and small indents to material inclusions or broken-out material. To simulate the defects on the surface of the virtual object, we use the displacement mapping technique (Cook, 1984).

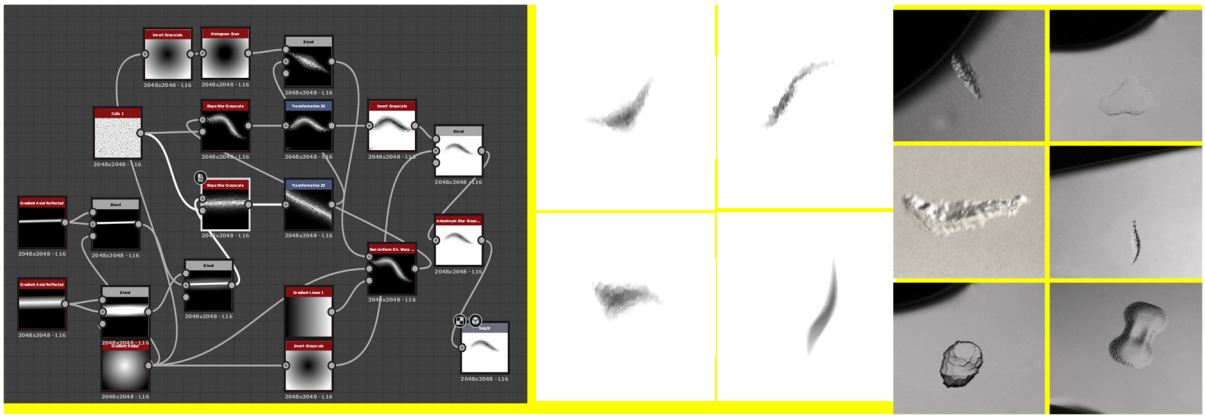

Figure 3: Structure of the Substance Designer 3D defect creation workflow, resulting defect textures and rendered examples.

In our case, this approach starts with the defect-free surface of a three-dimensional object and modifies it to add defects. An 8-bit RGB texture called a displacement map or height map is used for this purpose, which contains information about the displacement of the polygon mesh at every point in the form of a grey value in each pixel. As the actual polygon mesh of the object is changed during displacement mapping, the number of polygons may have to be greatly increased to create a realistic and detailed surface.

This technique improves reflection effects and ensure that the surface deviation of defects is always visible, even if they are viewed from a steep angle. For the generation of the displacement maps we utilize the software Substance Designer 3D from Adobe, which allows us to procedurally generate complex surfaces and materials through a node-based workflow (figure 3). The individual nodes consist of various mathematical image operations, which are connected in series to model the corresponding defect from a basic geometric shape. The nodes were selected in such a way that a change of individual values within the image processing operation has comprehensible effects on the displacement maps created to obtain an estimate of the extent to which these can be changed so that as many different realistic variants of the modelled defect as possible can be created. This automation technique is based entirely on the procedural modification of individual values within the nodes.

To simplify the automation process, a Python plugin for Substance Designer was developed, which offers simple customisation options for adapting the generation process to new defects or other Substance Designer graphs. The plugin is configured via a JSON file that contains information or IDs of the individual mathematical operations, the corresponding variables, and a list of possible values. Within the automatic process, each node configured in the JSON file is iterated over and the graph is recalculated for each value used, thus generating an arbitrary number of output textures. The advantage of this procedural process is that the resulting displacement maps are precisely defined by a suitably selected graph, duplicates can be avoided, and the result is correspondingly predictable.

The resulting displacement maps for the defects must now be applied to the texture corresponding to the correct location on the diaphragm spring. This requires mapping the 3D mesh of the object to a 2D texture, which we realized within Cinema 4D´s UV editor.

For the placement of the defect, we developed a comprehensible python workflow, executable in Jupyter notebooks. For this, we first need to mark all polygons of the 3D object, where a defect can occur. The script can then be used to export this selection into a mask and automatically scale and place the defect texture within the exported mask on the final displacement texture for the object.

3.4 Texture Modelling

Besides the damage modelling, we also need textures to control the reflections and colouring of the simulated object and environment, to achieve a realistic result. For this reason, we expanded the functionality of our defect generation Python plugin to be able to automate all kinds of texture graphs. Substance Designer's extensive library makes it easy to find suitable textures, manipulate and randomize them in a targeted manner.

In case of the simulation of the diaphragm spring, which is made of slightly reflective steel with some production induced discolorations, we need two

components - a colour map and a normal map. The colour map will be used to simulate blemishes on the surface that do not indent the diaphragm spring itself in a significant way. The normal map will simulate deeper indents and impurities that affect the depth of the surface but still are not classified as defect. Normal mapping is a texturing technique that simulates the appearance of dents and bumps without changing the actual polygon mesh (Cignoni et al., 1998). It uses a normal map to change the way the surface reflects light by storing the alignment of the surface normal in each pixel as a RGB value that corresponds directly to the x, y, and z coordinates of the 3D environment.

Realistic normal maps play a key role for the variety of the synthetic dataset, since we use them to simulate imperfections on the surface which are not classified as a defect and should not be identified as such by our deep learning model. A multitude of different normal maps should therefore reduce the number of false positives detected by the model. For the basis of the normal map, we characterized the surface of numerous real diaphragm springs, as well as the images from the real dataset from the production line. This resulted in multiple different Substance Designer 3D graphs, each highlighting distinct aspects of the surface for more individual control. For example, small scratches that can occur during machining were simulated by generating random splines on the surface, specks of dirt by overlaying randomly generated grunge maps, or the surface composition itself by choosing different combinations of noise generators.

Each of these workflows is then automated with the help of our Substance Designer plugin, for which we specify the values that can occur within individual nodes in a configuration file for each graph.

For the stamped part number on the diaphragm spring, we utilized Cinema 4D´s bump mapping feature, which uses a technique similar to normal mapping, but is configured via grayscale values instead of RGB information and is available inside a different material channel. The bump maps were generated by placing random numbers along a spline corresponding to the location, where the stamp can occur on the diaphragm spring´s texture.

Afterwards we selected some generated textures from each automated texture graph and inserted them into the Cinema 4D material for the diaphragm spring, to ensure that they achieve the desired effect.

Figure 4: Rendered image/defect mask/object mask.

This also helps finding the right configuration for the material settings, so we can finetune different materials for specific visual effects for later automation purposes. For the reflection layer of the material, the GGX reflection model is used together with the metal absorption presetting. This model is particularly suitable for simulating metallic surfaces[1].

3.5 Rendering Pipeline

Finally, the components created from the previously developed methods are combined and assembled in an automatic pipeline to generate the desired synthetic image data of the diaphragm spring.

We developed two approaches to automate this process with either Cinema 4D´s internal render solutions or Maxwell Render 5´s engine. Although these two methods overlap in their basic functionality, they offer different advantages and disadvantages when customised more precisely for different production parts or processes.

Our first approach utilizes the Python API of Maxwell Render 5 in combination with the digital twin exported from Cinema4D. It is parametrized via a configuration file, from where all elementary functions of the Python automation script can be accessed. We chose this variable configuration concept for better scalability, so that new functions or scenarios can be implemented into our automation script with little effort. Furthermore, this concept allows us to easily run the automation script on a separate rendering server and profit from the GPU Render Engine from Maxwell Render to minimize the required runtime.

The current configuration file can be divided into the following broad parent categories for configuring and randomising the simulation. It contains the paths to all the folder and files necessary for the simulation, such as texture and output directories, settings to configure which textures should be randomized and the range within the texture parameters can be changed, dynamic simulation parameters that affect the physical environment, such as object rotation and placement, configuration options for automatically

[1] Maxon Cinema 4D R23 Handbook: Material Editor: Reflectance

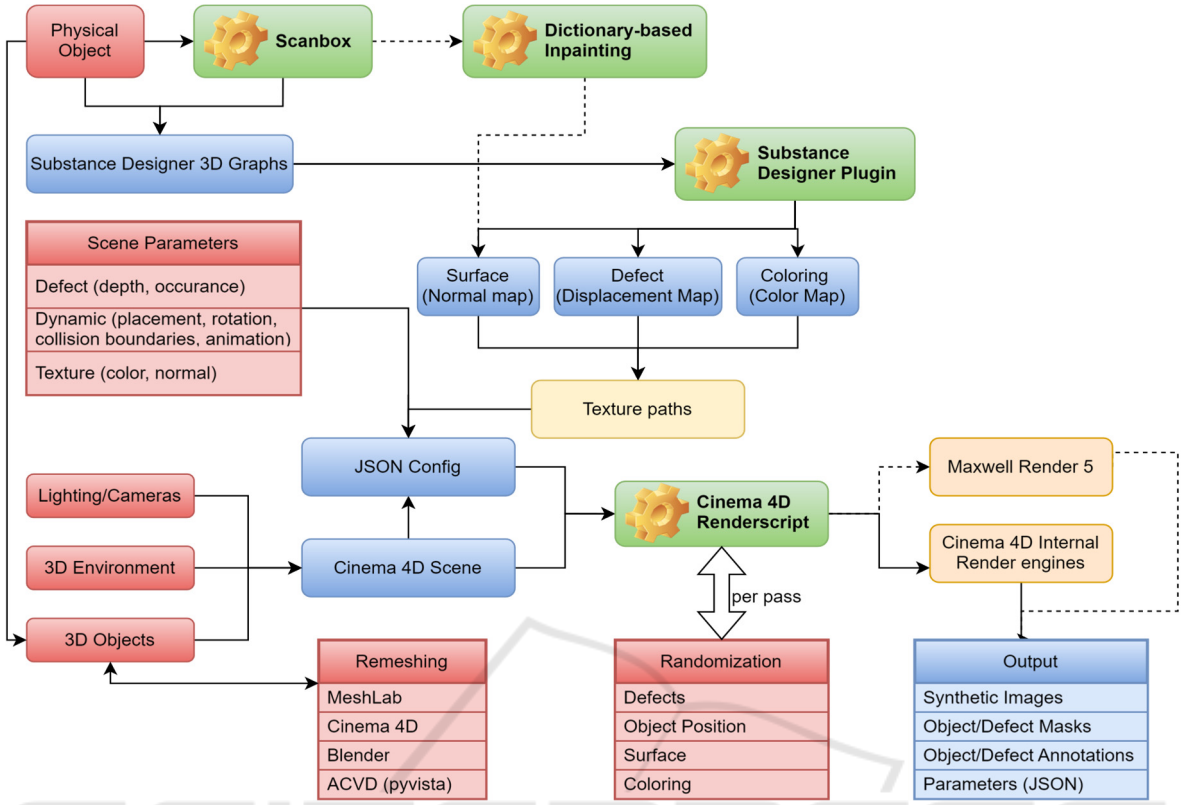

Figure 5: Flowchart of the rendering pipeline.

generated annotations, including options for multiple object and defects and settings that affect the rendering process, such as number of renderings, render quality settings and additional command line arguments for the render engine.

To improve the physical accuracy of the rendered images, the basis for a simple simulation of the physical properties of the environment was created. This means that basic rules for placing the objects in the scene can be defined in the configuration file in such a way that overlapping polygon meshes of the objects can be minimised or completely excluded. This also makes it possible to create simulations with multiple objects in which the objects can be randomly distributed in the scene.

For each object and defect in the scene, a mask with the respective segmented defect or object is output for each rendered image (figure 4), as well as a text file containing information about the corresponding bounding box in YOLO PyTorch TXT format.

The second automation approach we created relies purely on Cinema 4D and its internal render solutions. This development was initiated in order to have more direct customizations of the Cinema 4D scene or simulation itself available, as the Python interface of Cinema 4D offers a few more configuration options than that of the Maxwell Render engine.

In addition, one of the Cinema 4D internal render engines can be used for certain components, which reduces the required runtime of the automatic rendering process in some cases. The use of the dynamic simulation engine opens further possibilities for the physically accurate placement of objects within a scene and can also be used in conjunction with the Maxwell render script, for more realistic lighting conditions in the final rendering. Cinema 4D offers far more configuration options for materials, making it possible to customise the finer details of the materials and surfaces and add specific markings or effects to the rendered industrial component. It is also possible to use other functions of Cinema 4D, such as the creation of animation videos or the use of various generators for modelling objects/scenes during the rendering process. The basic concept of the Cinema 4D automation script is still very similar to the Maxwell script and offers, among other things, the same functions. The configuration is carried out via a config file, the Cinema 4D scene itself and the textures required for randomisation.

Figure 6: Synthetic images with different defects and normal maps generated by the rendering pipeline.

As output, we receive the rendered image, a binary mask for each object and defect in the scene and a JSON file for each image. The mask(s) for the defect(s) are created by automatically going through an additional rendering pass, where all textures, except for the displacement map are removed, leading to a rendering where only the defects are illuminated. The mask(s) for the object(s) are generated by using Cinema 4D´s own multiple-render-pass function, which will create a mask for each specified object. The JSON file, contains information about rendering parameters, including textures used, strength of normal map and bump mapping, as well as the strength of the defect.

For the synthetic dataset, which is used for supplementing the real dataset, we generated images with the same configuration, but a different lighting environment and processed them with the help of another script to simulate the image process that is already used in the real production. Figure 6 showcases some rendered image examples of the diaphragm spring with different automatically generated defects and normal maps from the resulting synthetic dataset.

3.6 Automated Defect Detection

There are many deep learning architectures suitable for detecting defects. Due to the precise information on the position of the defects, which is simulated and annotated with the help of the synthetic data, it was decided to localise the defects to simplify subsequent manual quality checks of the predictions.
Object detection algorithms such as YOLO (Redmon et al., 2016) or R-CNN (Ren et al., 2017), but also segmentation approaches such as U-NET (Ronneberger et al., 2015) or Mask-R-CNN (He et al., 2020) are suitable for such purposes.

With respect to the fact that the synthetic data can later be used to supplement real data from the production, the creation of the model for the pipeline was based on the conditions of the production environment and existing real image data from this environment. For this purpose, we were provided a real image data set with images from the current production as well as a defect catalogue containing information on which surface anomalies have already been classified as defects on the diaphragm springs.

The real images were manually annotated using the defect catalogues by marking each defect with a bounding box, so that the resulting real dataset can be used to train deep learning models for object detection. This dataset is used to train a baseline model, which was exclusively trained with real data from the production, as well as to evaluate the models that were trained with the synthetic dataset and to create a supplemented dataset containing real-world and synthetic data.

To conduct initial tests, the deep learning model "YOLOv8" from Ultralytics (Jocher et al., 2023) was used. We chose this model for its low hardware resource consumption for inference, its fast inference rate and easy-to-use configuration options. We expect that Mask-R-CNN or U-NET models will perform comparably on the data set. Note that the focus of this paper is the simulation pipeline and not an optimized defect detection model, therefore we show only first results on the impact of adding synthetic data.

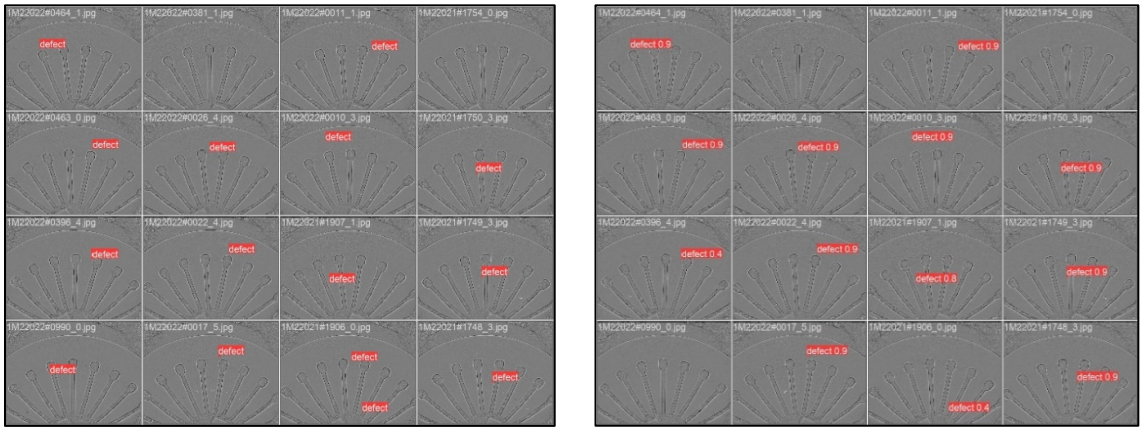

Figure 7: Defect detection Real + Synthetic V2: Ground Truth (left) / Prediction with Confidence (right).

The real image dataset from production used for training consisted of a total of 231 images, 105 of which were used for validation. A total of 300 images were generated for the basic synthetic dataset, of which 100 were without defects, 75 with multiple defects and 125 with single defects. The second version of the synthetic dataset was further supplemented with 100 more single defect images and 100 more images with multiple defects.

To assess the quality of the synthetic data in comparison to the real-world data, the models were trained with different datasets, and each evaluated with the same validation data set consisting of 105 real-world images. The first model "Real" was trained exclusively with real images from production, "Synthetic v2" only with synthetic image data and for the model "Real + Synthetic v1" and "Real + Synthetic v2" these two training data sets were combined.

All models were trained for 200 epochs, with an image size of 1408x1408 pixels, a batch size of 32 and the SGD Optimiser (learning rate initial: 0.1, learning rate final: 0.01). Since the defects were quite small on the real production images, it was necessary to use such a high image resolution for the training process. For the evaluation of the resulting models, we used the metrics Mean Average Precision at an Intersection of Union threshold of 0.5, Precision and Recall.

Based on the results in Table 1 it can be observed that supplementing the real data set with the synthetic image data in the "Real + Synthetic v1" model improves the precision, while in "Real + Synthetic v2" all of the three evaluation metrics improved compared to the "Real" model. This improvement is important when it comes to real production use of such a model.

Table 1: Evaluation metrics of the training.

Dataset	mAP	Precision	Recall
Real	0.672	0.703	0.655
Synthetic v1	0.207	0.268	0.306
Synthetic v2	0.343	0.428	0.575
Real + Synthetic v1	0.751	0.885	0.616
Real + Synthetic v2	0.785	0.911	0.658

To properly implement the model in an active environment, we will still need to evaluate on more real-world data and generate suitable and especially more synthetic image data. Although the mAP of the “Synthetic v2” model which was exclusively trained on synthetic data is significantly lower, when compared to the “Real” model, it also shows that the synthetic data is quite close to the real data, since the model can detect defects on the real data to some degree. This also indicates that we can use synthetic data to train a model to recognize defects it has never seen before on real data. Note that these tests were performed with the goal of assessing the impact of using synthetic data and not of fine-tuning the model for peak performance. Additional hyperparameter optimization is expected to improve the metrics further. Increasing the amount of synthetic data further will also bring additional improvements.

4 CONCLUSION & OUTLOOK

In this work we presented a digital reality pipeline to create synthetic image data that can be applied to

visual quality inspection. We demonstrated that our approach can be used to train deep learning models and improve the performance of models when used in combination with real data.

The digital reality pipeline is designed for low manual configuration effort allowing for a quick, affordable, and thus broad use in industry. We have analysed numerous processes and previous developments in this area, adapted and improved them, and combined these methods in a toolbox to enable easy implementation for different production parts. The main advantage of our approach compared to other simulation tools lies in the implementation of the modular digital reality concept. The next step is to conduct a comprehensive study with different types of objects and more synthetic data in combination with fine-tuning the object detection models to elaborate on the preliminary results described in section 3.6. Furthermore, the processes for synthetic data generation and defect detection described in the previous sections are suitable for integration as an automated visual quality control system into an active production environment. This requires a continuous process to allow the defect detection model to be improved or enriched with new data. An example of this would be training with a new defect type that only occurred on the object after the model has been deployed. The digital reality concept of using partial models reduces the workload of integrating this new defect type to parameterising the defect model accordingly and finetuning the deep learning model with the newly generated data. In the future, we aim to further evaluate the pipeline with additional use cases and an integration study in a real production environment.

ACKNOWLEDGEMENTS

Part of this research has been funded by the Ministry of Economics, Innovation, Digitalisation and Energy of Saarland under grant number D/2-ML-SYNTHOM-7/2022.

REFERENCES

Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., Thakar, S., Yoon, Y. J., & Gupta, S. K. (2021). Image-Based Surface Defect Detection Using Deep Learning: A Review. *Journal of Computing and Information Science in Engineering*, *21*(4). https://doi.org/10.1115/1.4049535

Borrego, J., Dehban, A., Figueiredo, R., Moreno, P., Bernardino, A., & Santos-Victor, J. (2018). *Applying Domain Randomization to Synthetic Data for Object Category Detection*. http://arxiv.org/abs/1807.09834

Chen, W., Wang, H., Li, Y., Su, H., Wang, Z., Tu, C., Lischinski, D., Cohen-Or, D., & Chen, B. (2016). Synthesizing Training Images for Boosting Human 3D Pose Estimation. *2016 Fourth International Conference on 3D Vision (3DV)*, 479–488. https://doi.org/10.1109/3DV.2016.58

Cignoni, P., Montani, C., Rocchini, C., & Scopigno, R. (1998). A general method for preserving attribute values on simplified meshes. *Proceedings Visualization '98 (Cat. No.98CB36276)*, 59–66. https://doi.org/10.1109/VISUAL.1998.745285

Cook, R. L. (1984). Shade Trees. *Computer Graphics (ACM)*, *18*(3), 223–231. https://doi.org/10.1145/964965.808602

Dahmen, T., Trampert, P., Boughorbel, F., Sprenger, J., Klusch, M., Fischer, K., Kübel, C., & Slusallek, P. (2019). *Damen et al. (2019). Digital reality. A model-based approach to supervised learning from synthetic data.pdf*. 1–12.

de Melo, C. M., Torralba, A., Guibas, L., DiCarlo, J., Chellappa, R., & Hodgins, J. (2022). Next-generation deep learning based on simulators and synthetic data. *Trends in Cognitive Sciences*, *26*(2), 174–187. https://doi.org/10.1016/j.tics.2021.11.008

Fabbri, M., Brasó, G., Maugeri, G., Cetintas, O., Gasparini, R., Ošep, A., Calderara, S., Leal-Taixé, L., & Cucchiara, R. (2021). MOTSynth: How Can Synthetic Data Help Pedestrian Detection and Tracking? *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 10829–10839. https://doi.org/10.1109/ICCV48922.2021.01067

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 27). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

Gutierrez, P., Luschkova, M., Cordier, A., Shukor, M., Schappert, M., & Dahmen, T. (2021). *Synthetic training data generation for deep learning based quality inspection*. https://doi.org/10.1117/12.2586824

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2020). Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(2), 386–397. https://doi.org/10.1109/TPAMI.2018.2844175

Jain, S., Seth, G., Paruthi, A., Soni, U., & Kumar, G. (2022). Synthetic data augmentation for surface defect detection and classification using deep learning. *Journal of Intelligent Manufacturing*, *33*(4), 1007–1020. https://doi.org/10.1007/s10845-020-01710-x

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8*. https://github.com/ultralytics/ultralytics

Lee, S., Park, E., Yi, H., & Lee, S. H. (2020). StRDAN: Synthetic-to-real domain adaptation network for

vehicle re-identification. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2020-June*, 2590–2597. https://doi.org/10.1109/CVPRW50498.2020.00312

Li, J., Su, Z., Geng, J., & Yin, Y. (2018). Real-time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network. *IFAC-PapersOnLine*, *51*(21), 76–81. https://doi.org/10.1016/j.ifacol.2018.09.412

Li, S., & Wang, X. (2022). YOLOv5-based Defect Detection Model for Hot Rolled Strip Steel. *Journal of Physics: Conference Series*, *2171*(1), 1–7. https://doi.org/10.1088/1742-6596/2171/1/012040

Maxwell Documentation. (2024). https://nextlimitsupport.atlassian.net/wiki/spaces/maxwell/pages/22683383/GPU+engine

Mazzetto, M., Teixeira, M., Rodrigues, É. O., & Casanova, D. (2020). Deep Learning Models for Visual Inspection on Automotive Assembling Line. *International Journal of Advanced Engineering Research and Science*, *7*(3), 473–494. https://doi.org/10.22161/ijaers.74.56

Mumuni, A., Mumuni, F., & Gerrar, N. K. (2024). *A survey of synthetic data augmentation methods in computer vision*. https://arxiv.org/abs/2403.10075v2

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9351*. https://doi.org/10.1007/978-3-319-24574-4_28

Schraml, D. (2019). *Physically based synthetic image generation for machine learning: a review of pertinent literature*. *September 2019*, 51. https://doi.org/10.1117/12.2533485

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. *IEEE International Conference on Intelligent Robots and Systems*, *2017-Septe*, 23–30. https://doi.org/10.1109/IROS.2017.8202133

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. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1082–10828. https://doi.org/10.1109/CVPRW.2018.00143

Wang, Y., Hao, Z., Zuo, F., & Pan, S. (2021). A fabric defect detection system based improved YOLOv5 detector. *Journal of Physics: Conference Series*, *2010*(1). https://doi.org/10.1088/1742-6596/2010/1/012191