Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 107 (2022) 1101–1106

www.elsevier.com/locate/procedia

55th CIRP Conference on Manufacturing Systems

Procedural synthetic training data generation for AI-based defect detection in industrial surface inspection

Ole Schmedemann[a,*], Melvin Baaß[a], Daniel Schoepflin[a], Thorsten Schüppstuhl[a]

[a]*Hamburg University of Technology, TUHH, Institute of Aircraft Production Technology, Denickestr. 17, 21073 Hamburg, Germany*

* Corresponding author. Tel.: +49-40-42878-3234 ; fax: +49-40-42731-4551. *E-mail address:* ole.schmedemann@tuhh.de

Abstract

Supervised machine learning methods are increasingly used for detecting defects in automated visual inspection systems. However, these methods require large quantities of annotated image data of the surface being inspected, including images of defective surfaces. In industrial contexts, it is difficult to collect the latter since acquiring sufficient image data of defective surfaces is costly and time-consuming. Additionally, gathered datasets tend to contain selection-bias, e.g. under representation of certain defect classes, and therefore result in insufficient training data quality. Synthetic training data is a promising alternative as it can be easily generated unbiasedly and in large quantities. In this work, we present a procedural pipeline for generating training data based on physically based renderings of the object under inspection. Defects are being introduced as 3D-models on the surface of the object. The generator provides the ability to randomize object and camera parameters within given intervals, allowing the user to use the domain randomization technique to bridge the domain gap between the synthetic data and the real world. Experiments suggest that the data generated in this way can be beneficial to training defect detection models.

Keywords: Synthetic training data; machine learning; surface inspection; industrial quality control; domain randomization

1. Introduction

Automated optical inspection systems use computer vision to perform inspection tasks. Setting up a classic computer vision system is a tedious and time consuming task, e.g. if free-form geometries have to be inspected. Recent advances in machine learning methods, most of all Convolutional Neural Networks (CNNs), promise to be an alternative to classic image processing pipelines. Critical for the performance of CNNs is the data with which they are trained. The weakness in visual defect detection is sufficient training data showing defective surfaces, mainly due to the following reasons:

- Collecting defective samples is cost intensive [1]
- Biased data through predomination of non-defective samples [2]
- Laborious manual annotation [3]
- Lack of publicly available datasets [4]

Defective samples are naturally rare in an industrial production line. Additionally, the collected defective samples will be biased towards certain types, shapes or positions of defects on the surface making them less suitable as training data. Annotating the collected data by hand is laborious, time-consuming and often requires expertise for identifying defects in images. Therefore, the creation of comprehensive datasets is often challenging.

Synthesizing training data by means of computer graphics is an increasingly used alternative [5, 6]. Benefits of synthetic data generation are: fast and cheap production of large quantities, rare events can be represented in synthetic datasets just as frequently as common events, and automatic annotation of class labels on a pixel level. Furthermore, training data can be generated before the first part has been produced, making AI-assisted detection of surface defects immediately available at production start. While initial research indicates that synthetic data may be beneficial [2], its utilization remains challenging and not ubiquitous. The main obstacle is the transferability from the synthetic domain to the real domain [7].

In this work, a novel approach for synthesizing training data for CNN-assisted defect detection is proposed. Existing approaches for synthetic defect generation are based on 2D mergence of images [8] and 2D defect maps [2, 9–11]. Despite initial success and principal applicability of synthetic data generation for defect detection, those approaches are limited to special use cases, complex in implementation, or not able to completely close the domain gap. Therefore, we aim to introduce a novel approach for synthesizing 3D objects and defect data.

For this approach, a 3D model of the object under inspection is rendered. Defects are introduced into the model using negatives of the defect geometry. To bridge the domain gap to the real world the principle of domain randomization is used [12]. Process parameters like textures, illumination, and defect shape are randomized within intervals. The goal is to extend the synthetic domain to such an extent that the real domain becomes a subset of the synthetic domain, enabling models trained with synthetic data to be applied to real-world data. In addition, less sensor realism is needed to reduce the modeling complexity in the rendering pipeline. To our knowledge, this is the first extensive utilization of the principle of domain randomization for the generation of synthetic data for visual defect detection.

2. Related work

CNNs have improved visual object recognition and object detection significantly [13]. However, when only trained with datasets of small size, models tend to overfit and produce poor results. Popular strategies in dealing with small datasets are data augmentation, transfer learning, use of pre-trained models, or synthetic training data.

For detection of everyday objects, extensive datasets are available, e.g. ImageNet [14], KITTI [15], or MS COCO [16], which can be used to pre-train models. On the contrary, publically available datasets for surface defect detection, e.g. NEU-DET [3] or GC10-DET [4], are not extensive enough to be used for pre-training and are limited to specialized use cases, thus lacking the ability to transfer to new inspection tasks. The availability of suitable training data inhibits the further spread of the use of CNNs in industrial inspection.

2.1. Synthetic training data

Multiple approaches such as [12, 17–20] have shown that synthetic data can be used to solve the data availability problem with small or biased datasets. Nikolenko gives an extensive survey on research regarding synthetic training data [21]: Machine learning models usually assume, that training and test data distributions are similar. However, the distributions of synthesized training data and real world test data differ significantly, making the domain transfer from source to target domain challenging. Several strategies have been proposed to tackle this challenge:

- Domain adaptation
- Sensor-realistic rendering
- Domain randomization

The goal of domain adaptation techniques is to reduce the statistical deviation between source and target domain. Several strategies, e.g. adversarial learning or generative-based, have been suggested to adapt the domains. Toldo et al. [22] summarize state of the art domain adaption methods.

As stated by Hodan et al. [23] a high degree of visual realism can be achieved by focusing on modeling the geometry, textures and materials to a high level of detail and by simulating the lighting as physically correct as possible. A technique known as physically based rendering (PBR) [24] has been shown to help in bridging the domain gap and may even be necessary for rendering usable images featuring complex reflections.

The goal of domain randomization (DR) as introduced by Tobin et al. [12] is to extend the synthetic domain to the point where the real domain becomes a subset of the synthetic domain. They showed that DR enables the use of lower-quality renderers which are optimized for speed. Tremblay et al. [25] demonstrated that parts of the domain, which are not the feature that is to be detected, can be randomized in a non-realistic way. Prakash et al. [18] demonstrate further that generating images which preserve the structure of a scene can increase the performance of DR further.

Several toolboxes like BlenderProc [26], NDDS [27], and Perception [28] have been published in order to assist researchers to render images and generate annotations. These toolboxes assume that 3D models and texture maps of the objects to be rendered are available. For surface defects, these must first be created, thus they cannot be directly deployed for industrial inspection tasks.

2.2. Synthetic data generation for visual quality inspection

Successful automated visual defect detection with CNNs has been extensively demonstrated [29–33]. Typically, these authors approach the data problem by tediously collecting data for their use case or by relying on existing data collected with vision systems in the field. Each of the aforementioned researchers created a dataset specific to their inspection task that cannot be generalized to other inspection tasks.

There exists a variety of approaches for the generation of synthetic data for visual inspection that can be categorized into rendering-based and generative-based methods. Generative approaches use generative adversarial networks (GANs) to create or augment training data [10, 11, 34–36] making use of their intrinsic domain adaption capability, while rendering approaches [1, 2, 7, 9, 37] aim for high sensor-realism.

Retzlaff et al. [7] use procedural modeling and PBR to generate images for glass shade classification. Lee et al. [37] combine DR and sensor-realism to render images for industrial object detection. [2, 8, 9] create synthetic images for defect detection. Haselmann and Gruber [8] use real images and overlay them with randomly generated defect textures. [2, 9]

use PBR pipelines to render images for the defect detection on metal surfaces. In both approaches, defects are introduced using artificially created 2D textures. Gutierrez et al. [2] use photometric stereo to capture bump maps to texture their model. Existing approaches partly lack successful domain transfer, or imply high effort for generating textures.

We aim to overcome limitations of existing approaches by making extensive use of DR. We introduce defects as 3D models in the rendering pipeline. Thus we are able to vary the shape, size, and position of the defects. Additionally, we use procedural textures instead of laboriously collected real-world textures. With the procedural textures, the look of the object can be easily varied by changing the texture parameters which increases the domain randomization potential.

3. Synthetic data generation pipeline

We first present the overall concept of our data generation pipeline. Afterward we detail the parameters that are randomized by the pipeline. Finally, the implementation is presented.

3.1. Training data generator concept

Our training data generator (TDG) takes user inputs like models, textures, and process parameters. Then the TDG generates a scene with a defective object and renders an image for a set of randomly chosen process parameters, see Fig. 2.

Starting point for the TDG is a 3D model of the inspected object, which is usually available at the manufacturer. Additionally, procedural textures are chosen for each surface type of the object. A procedural texture is a texture created using a mathematical description. We use procedural textures for three reasons. Firstly, they can be easily randomized by varying the parameters used to describe them. Secondly, they allow the object to be viewed from arbitrary distances since their resolution scales with the distance. Lastly, texture mapping with procedural textures does not generate seams on the part reducing the risk for unwanted artifacts in the rendered image.

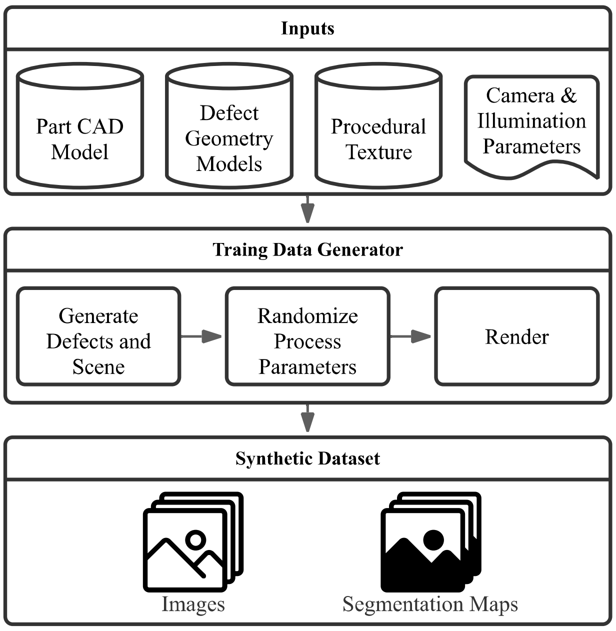

Fig. 2. Concept of the training data generator.

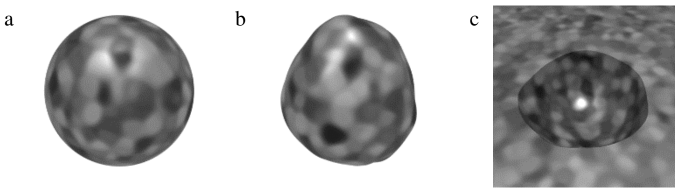

Fig. 1. (a) Negative defect form; (b) negative defect form with displacement modifier; (c) defect introduced to part.

Procedural textures can be chosen from texture libraries for common materials. An exact modeling of the real surface behavior is not necessary, since the parameters describing the texture will be randomized within chosen intervals. Thus the total effort is reduced compared to other methods, e.g. [2]. Together with the procedural textures, we use physically based rendering (PBR) to keep the discrepancy between the synthetic and the real-world domain small.

Additional inputs for the TDG are camera and illumination parameters. Camera parameters are necessary to describe the virtual camera for the rendering engine and include field of view, resolution, and position and orientation. The illumination parameters are used to approximate the illumination behavior in the inspection setup and include shape, size, intensity, color temperature, position, and orientation.

Based on these inputs the TDG generates the synthetic images. The first step is the defect creation. Our defect generation method is inspired by Bosnar et al. [38]. Defect tools are used to modify the mesh of the main object using Boolean operators. A negative defect tool is modelled for each defect type. The tool must have the shape which has to be imprinted into or added onto the surface of the inspected object. Fig. 1 shows the defect generation exemplarily for a spherical defect tool, which can be used to simulate blowhole defects in cast iron. For different defect types tools can be created and used in a similar way in the TDG. Defect tools that are not symmetric may be oriented towards the surface normal at the point of introduction into the object.

Outputs of the TDG are the rendered images as well as a segmentation map, which states which pixel belongs to a defective or a defect-free surface. In that way, our generated images can be used to train a model for object segmentation but also for object detection and image classification tasks. To ensure transferability for our generated images, we aim to randomize as many parameters as possible within our modeling. Next, we specify what those parameters are.

3.2. Domain randomization

We group our parameters, based on the object type the parameter is assigned to, into the four groups Defect, Illumination, Camera, and Texture, see Table 1. Intervals and distributions for the parameters have to be given as an input for the TDG. If no distribution is given then an equal distribution is assumed.

The defect shape is randomized by varying its size, position, and orientation on the surface of the object. Additionally, a displacement modifier based on a procedural texture is placed on the defect tool, see Fig. 1b. For the input parameters of the procedural texture intervals for the randomization have to be set. The texture randomization depends on the chosen procedural texture for a material. An example of an implementation is given in chapter 4.

Camera and illumination position and orientation of the virtual camera are varied within the given intervals. For the illumination the intensity is varied.

Table 1. Randomized parameter in training data generator pipeline.

Group	Parameters for randomization
Defect	Position, orientation, size, shape
Illumination	Position, orientation, intensity, spectral distribution, shape
Camera	Position, orientation
Texture	Roughness, base color, normal map

3.3. Implementation

The TDG was implemented using the 3D computer graphics software toolset Blender. Blender features a python API which allows us to automate the training data generation by executing scripts. Furthermore, Blender allows for advanced 3D object manipulation that is used to execute the Boolean operations to integrate the defect tool shape into the object. For physical based rendering we make use of Blenders raytracing engine Cycles. In addition, we use BlenderProc [26] to create the segmentation maps showing the defect position in the rendered images, see Fig. 4h.

Fig. 3 shows the general structure of the TDG. For each defect object, a script to generate the defects is executed, which contains the object's model with the material applied to it. The defect tool is used to generate all defects. Next, the supplied camera poses get randomized and for each pose, an arbitrary number of additional randomized poses can be added. The middle for-loop iterates over all of these randomized poses. Then the according light pose is computed. In the same step, the light intensity is randomized too. Finally, the innermost for-loop randomizes the texture. Afterwards, all elements needed to execute BlenderProc have been assembled and the first image along with its segmentation map can be generated. When the textures have been changed the number of times that was specified, the next camera pose is chosen and after all camera poses have been used, the next defect object is generated. The size of the dataset is thus the product of iterations of each for-loop.

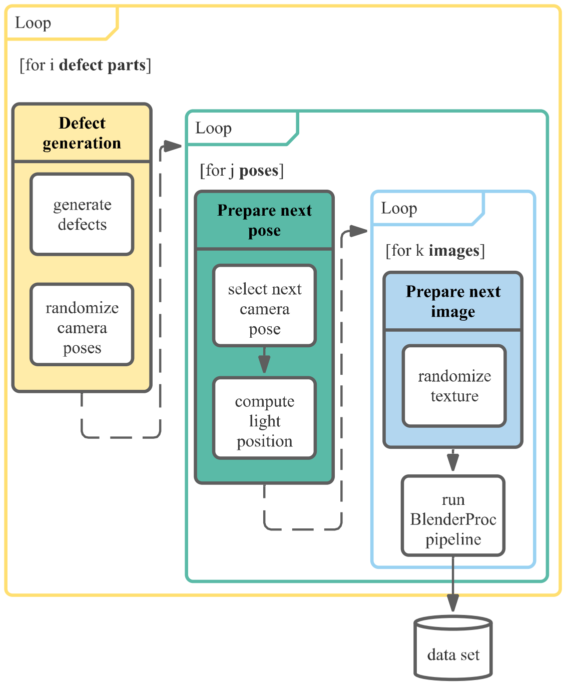

Fig. 3. Flowchart of the training data generator.

4. Demonstration

To demonstrate our training data generation pipeline a part from the manufacturing industry was chosen: a turbocharger housing for the automotive industry. The cast iron component has to be visually inspected after demolding. We choose the endoscopic inspection of the part's cavities for this demonstration. Difficult illumination conditions, varying relative positioning of the camera to the part's surface, as well as freeform surfaces make the visual inspection of the part challenging. The concept of using synthetic data for endoscopic inspection was first published in [39].

The cast iron component consists of only one surface texture, therefore only one texture needs to be modeled. In addition, the cavity allows to neglect having to model the background since it will not be visible in the images reducing the modeling complexity. This makes the part well suited as a demonstration part since it combines a challenging image processing task with relatively limited modeling effort.

4.1. Synthetic dataset generation for use case

To approximate the look of the real-world part, see Fig. 4a, we combined noise textures of different scale to generate a bump map. The result is shown in Fig. 4c. For this experiment we focus on the defect type 'blowhole' as seen in Fig. 4b. We created a spherical defect tool for our TDG. Fig. 4d shows a synthetic defect generated with the defect tool. We randomized image features that are not relevant for detecting the defect. Fig. 4e-g show exemplarily variations in roughness (e), color (f), and defect size (g).

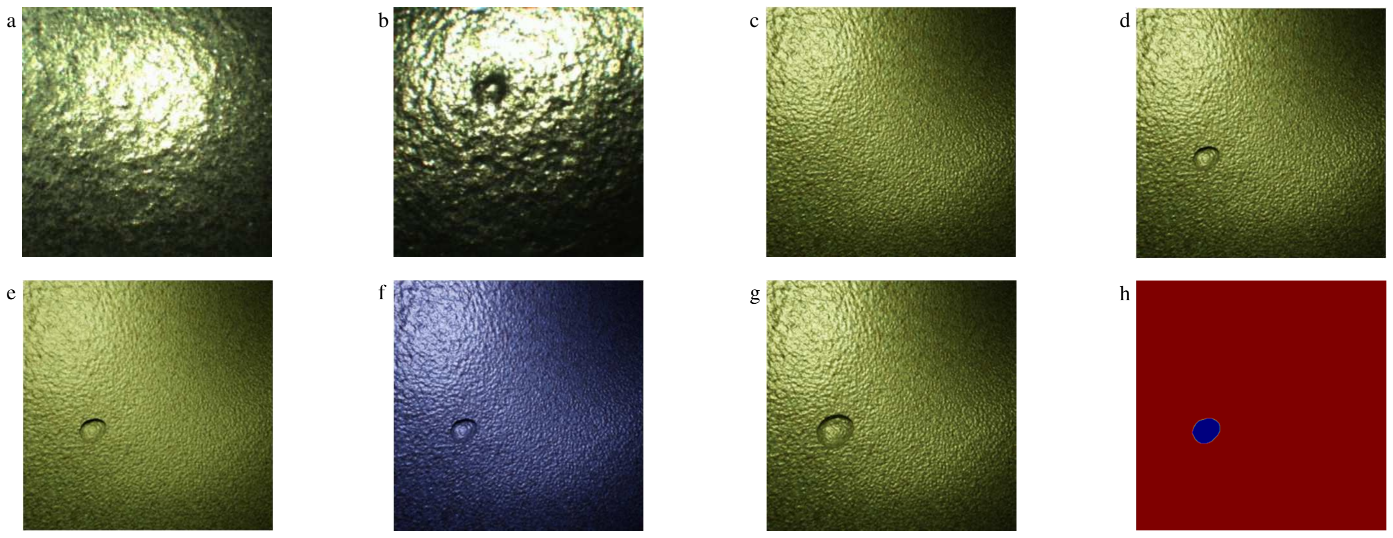

Fig. 4. (a) real-world endoscopic image of turbocharger housing cavity; (b) real-world endoscopic image of blowhole; (c) synthetic image from modeling; (d) synthetic image with generated defect from TDG; (e) example of variation of roughness; (f) example of variation of color; (g) example of variation of defect size; (h) segmentation map

We manually chose a set of 137 inspection poses as an input for the TDG. We generated 100 defects in a part and generated 822 images per part. We repeated this process for 6 times. The generated images were filtered depending on the defect size in the image. Defects larger than 50 px became part of the defective class. Images without visible defects were assigned to the defect-free class. Images with very small defects were not further considered. In this way a synthetic dataset with 4.906 images was created.

4.2. Model training

To evaluate the suitability of our approach we created two real-world datasets of endoscopic images. The training and the test dataset show both defective and defect-free textures, see Fig. 4a-b, and consist of each 110 images. The real-world training dataset was split into equally sized training and validation datasets.

We used an 18-layer ResNet [40] architecture which was pre-trained on the ImageNet dataset. Two experiments were conducted. First, we trained the model with the synthesized dataset. We split the synthetic dataset in an 80/20 ratio in a training and a validation dataset and pre-trained with a learning rate of 1e-4 and a batch size of 64 for 45 epochs. Then, we fine-tuned the model with the real-world training dataset. For the second experiment we trained the model directly with the real-world training dataset without using our synthetic data.

For the two experiments we conducted a hyperparameter search and evaluated combinations of three learning rates (1e-2, 1e-3, and 1e-4) and three batch sizes (16, 32, and 64). We trained for 25 epochs with an SGD optimizer. In both experiments the best performing network based on the validation accuracy was tested on the real-world test dataset. The model that was pre-trained with our synthetic data (accuracy 98.2 %, recall 96.2 %, specificity 100 %) outperformed the model solely trained on real-world data (accuracy 93.6 %, recall 94.2 %, specificity 93.1 %). The results of our investigation indicate that synthetic training data generated with our approach can be beneficial for training defect detection models when few real-world images are available.

5. Conclusion and outlook

We introduced and implemented a training data generator that makes extensive use of the principle of domain randomization and can be used to generate synthetic training data for visual defect detection tasks. Our concept radically reduces the overall complexity and effort needed to construct a synthetic dataset for a given inspection task.

The TDG allows researchers to explore what makes good synthetic training data for defect detection, which parameter to randomize, and how to set appropriate randomization intervals for the process parameters. Researchers can use the TDG to investigate if synthetic data is useful for their inspection task at hand.

Our use case demonstrates the usability of our approach. Future work will apply models with our synthesized data directly on real-world data without fine-tuning taking advantage of the scalability of our TDG to create very large datasets with high variability. Additionally, we will leverage the pixelwise ground truth information that our TDG provides for object detection and segmentation tasks.

CRediT author statement

Ole Schmedemann: Conceptualization, Methodology, Software, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization. **Melvin Baaß:** Conceptualization, Methodology, Software, Writing - Review & Editing. **Daniel Schoepflin:** Writing - Review & Editing. **Thorsten Schüppstuhl:** Writing - Original Draft, Supervision, Resources, Funding acquisition, Project administration.

Acknowledgements

This research was funded by the German Federal Ministry for Economic Affairs and Climate Action under grant number ZF4736301LP9.

References

[1] Peres, R.S., Guedes, M., Miranda, F., Barata, J., 2021. Simulation-Based Data Augmentation for the Quality Inspection of Structural Adhesive With Deep Learning. IEEE Access *9*, p. 76532.

[2] Gutierrez, P., Luschkova, M., Cordier, A., Shukor, M., Schappert, M., Dahmen, T., 2021. Synthetic Training Data Generation for Deep Learning Based Quality Inspection.

[3] He, Y., Song, K., Meng, Q., Yan, Y., 2020. An End-to-End Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features. IEEE Transactions on Instrumentation and Measurement *69*, p. 1493.

[4] Lv, X., Duan, F., Jiang, J.-J., Fu, X., Gan, L., 2020. Deep Metallic Surface Defect Detection: The New Benchmark and Detection Network. Sensors (Basel, Switzerland) *20*.

[5] Peng, X., Sun, B., Ali, K., Saenko, K., 2014. *Learning Deep Object Detectors from 3D Models*.

[6] Su, H., Qi, C.R., Li, Y., Guibas, L.J., 2015. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views, in *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE, p. 2686.

[7] Retzlaff, M.-G., Richter, M., Längle, T., Beyerer, J., Dachsbacher, C., 2016. Combining synthetic image acquisition and machine learning: accelerated design and deployment of sorting systems. Forum Bildverarbeitung 2016, p. 49.

[8] Haselmann, M., Gruber, D.P., 2019. Pixel-Wise Defect Detection by CNNs without Manually Labeled Training Data. Applied Artificial Intelligence *33*, p. 548.

[9] Boikov, A., Payor, V., Savelev, R., Kolesnikov, A., 2021. Synthetic Data Generation for Steel Defect Detection and Classification Using Deep Learning. Symmetry *13*, p. 1176.

[10] Niu, S., Li, B., Wang, X., Lin, H., 2020. Defect Image Sample Generation With GAN for Improving Defect Recognition. IEEE Transactions on Automation Science and Engineering, p. 1.

[11] Li, B., Yuan, X., Shi, M., 2020. Synthetic data generation based on local-foreground generative adversarial networks for surface defect detection. Journal of Electronic Imaging *29*, p. 1.

[12] 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 *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 23.

[13] LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature *521*, p. 436.

[14] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. ImageNet: A large-scale hierarchical image database, in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, p. 248.

[15] Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research *32*, p. 1231.

[16] 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 *Computer Vision – ECCV 2014*, Springer International Publishing, Cham, p. 740.

[17] Hinterstoisser, S., Lepetit, V., Wohlhart, P., Konolige, K., 2017. *On Pre-Trained Image Features and Synthetic Images for Deep Learning*.

[18] Prakash, A., Boochoon, S., Brophy, M., Acuna, D., Cameracci, E., State, G., Shapira, O., Birchfield, S., 2019. Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data, in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, p. 7249.

[19] Schoepflin, D., Holst, D., Gomse, M., Schüppstuhl, T., 2021. Synthetic Training Data Generation for Visual Object Identification on Load Carriers. Procedia CIRP *104*, p. 1257.

[20] Magana, A., Wu, H., Bauer, P., Reinhart, G., 92020. PoseNetwork: Pipeline for the Automated Generation of Synthetic Training Data and CNN for Object Detection, Segmentation, and Orientation Estimation, in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, p. 587.

[21] Nikolenko, S.I., 2021. *Synthetic Data for Deep Learning,* 1st edn. Springer International Publishing; Imprint Springer, Cham.

[22] Toldo, M., Maracani, A., Michieli, U., Zanuttigh, P., 2020. Unsupervised Domain Adaptation in Semantic Segmentation: A Review. Technologies *8*, p. 35.

[23] Hodan, T., Vineet, V., Gal, R., Shalev, E., Hanzelka, J., Connell, T., Urbina, P., Sinha, S.N., Guenter, B., 2019. *Photorealistic Image Synthesis for Object Instance Detection*.

[24] Pharr, M., Jakob, W., Humphreys, G., 2017. *Physically Based Rendering,* 3rd edn. Elsevier.

[25] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, Stan Birchfield, 2018. Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization. Proceedings of the IEEE conference on computer vision and pattern recognition workshops, p. 969.

[26] Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., Katam, H., 2019. *BlenderProc*.

[27] To, T., Tremblay, J., McKay, D., Yamaguchi, Y., Leung, K., Balanon, A., Cheng, J., Hodge, W., Birchfield, S. NDDS: NVIDIA Deep Learning Dataset Synthesizer, 2018.

[28] Unity Technologies. Unity Perception Package, 2020.

[29] Staar, B., Lütjen, M., Freitag, M., 2019. Anomaly detection with convolutional neural networks for industrial surface inspection, in *Procedia CIRP (79)*, Elsevier, p. 484.

[30] Soukup, D., Huber-Mörk, R., 2014. Convolutional Neural Networks for Steel Surface Defect Detection from Photometric Stereo Images, in *Advances in Visual Computing*, Springer International Publishing, Cham, p. 668.

[31] Weimer, D., Scholz-Reiter, B., Shpitalni, M., 2016. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. CIRP Annals *65*, p. 417.

[32] Kim, S., Kim, W., Noh, Y.-K., Park, F.C., 2017 - 2017. Transfer learning for automated optical inspection, in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, p. 2517.

[33] Faghih-Roohi, S., Hajizadeh, S., Nunez, A., Babuska, R., Schutter, B. de, 2016 - 2016. Deep convolutional neural networks for detection of rail surface defects, in *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, p. 2584.

[34] Martin Mundt, Sagnik Majumder, Sreenivas Murali, Panagiotis Panetsos, Visvanathan Ramesh, 2019. *CODEBRIM: COncrete DEfect BRidge IMage Dataset*. Zenodo.

[35] Mery, D., 2020. Aluminum Casting Inspection Using Deep Learning: A Method Based on Convolutional Neural Networks. Journal of Nondestructive Evaluation *39*.

[36] Jain, S., Seth, G., Paruthi, A., Soni, U., Kumar, G., 2020. Synthetic data augmentation for surface defect detection and classification using deep learning. Journal of Intelligent Manufacturing.

[37] Lee, Y.-H., Chuang, C.-C., Lai, S.-H., Jhang, Z.-J., 2019. Automatic Generation of Photorealistic Training Data for Detection of Industrial Components, in *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, p. 2751.

[38] Bosnar, L., Saric, D., Dutta, S., Weibel, T., Rauhut, M., Hagen, H., Gospodnetic, P., 2020. Image Synthesis Pipeline for Surface Inspection.

[39] Bath, L., Schmedemann, O., Schüppstuhl, T., 2021. Development of new means regarding sensor positioning and measurement data evaluation – automation of industrial endoscopy. wt Werkstattstechnik online, p. 644.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 2016. Deep Residual Learning for Image Recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, S. 770-778.