Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 138 (2026) 1097–1102

www.elsevier.com/locate/procedia

18th CIRP Conference on Intelligent Computation in Manufacturing Engineering

An Edge Is All You Need: Cracking the Code for Generating Synthetic Datasets for Robust Crack Detection Models

Dirk Holst*, Ole Schmedemann, Thorsten Schüppstuhl

Hamburg University of Technology, TUHH, Institute of Aircraft Production Technology, Denickestraße 17, 21073 Hamburg ,Germany

* Corresponding author. Tel.: +4940 42878-2999. *E-mail address:* dirk.holst@tuhh.de

Abstract

Generating synthetic datasets for training crack detection models remains a challenge due to the variability of crack appearances and the need for pixel-level annotations. As a promising alternative to manual labeling, synthetic data generation using Perlin Noise has gained attention. However, the relationship between Perlin Noise parameters and the physical characteristics of cracks is not well-established, leading to potential dataset bias and suboptimal model performance. This paper presents a novel approach that maps Perlin Noise generated crack parameters to crack characteristics such as width, length, curvature, and bifurcations, enabling the generation of diverse and realistic crack patterns. We employ a broad domain randomization technique by projecting the generated cracks onto randomly selected background images from the ImageNet database to reduce dataset bias and enhance model robustness. Using the Partitioning Around Medoids (PAM) algorithm, we create six distinct datasets capturing a comprehensive range of crack parameter variations. We demonstrate the effectiveness of our approach by fine-tuning the state-of-the-art SegFormer 5b model on our synthetic datasets and benchmarking its performance on the Crack500 dataset. Through linear regression analysis, we can identify the key crack parameters that influence model performance. Our results show that synthetic cracks have to be long, wide, straight, and should have few bifurcations to maximize evaluation metrics such as Intersection over Union (IoU), precision, recall, and F1.

Keywords: Synthetic Data, Crack Detection, Perlin Noise, Dataset Analyses, Pavement Cracks

1. Introduction

Ensuring the quality and longevity of goods and buildings is a significant challenge for the manufacturing and maintenance industries. One crucial aspect of this process is detecting and assessing surface-level cracks in various materials, which can lead to structural failures and safety hazards if left unaddressed. Traditional crack detection methods rely heavily on manual labor and visual inspection, which can be time-consuming, costly, and prone to human error. Therefore, there is a growing need for automated solutions that can efficiently and accurately detect and analyze surface cracks in various environments.

Recent advancements in computer vision and deep learning have shown great promise in addressing this challenge. However, the success of these techniques heavily depends on the availability of large, diverse, and accurately labeled datasets [1]. Obtaining such datasets for surface-level crack inspection can be difficult due to the rarity of defective samples, the variability of crack appearances, and the need for pixel-level annotations. Moreover, manually labeling images of cracks is a tedious and error-prone task, often resulting in inaccurate and inconsistent annotations [2].

To overcome these limitations, synthetic data generation has emerged as a promising solution [2]. By leveraging advanced

10.1016/j.procir.2026.01.189

computer graphics techniques and domain knowledge, it is possible to generate realistic and diverse images of cracked surfaces with accurate pixel-level annotations. This approach offers several advantages over relying solely on real-world data. First, it allows for the creation of large datasets with minimal human effort and cost. Second, it enables the generation of a wide range of crack patterns and backgrounds, enhancing the robustness and generalization capabilities of the trained models. Finally, it provides a controlled environment for studying the impact of various factors, such as crack modeling algorithms or the final appearance of synthetic cracks, on the performance of crack detection algorithms.

In this paper, we focus on evaluating the influence of different crack characteristics generated using Perlin Noise on the performance of neural networks for crack detection. To achieve this, we map the parameters of the Perlin Noise generated cracks to the physical characteristics of real cracks, such as length, width, number of bifurcations, and straightness. This approach allows us to analyze the impact of various parameterizations of the Perlin Noise algorithm on the quality of the synthesized datasets and the performance of the trained models. To address the domain gap problem and reduce dataset bias, we adopt a generalist approach using broad domain randomization. Instead of fitting the training data to a specific domain, we project each generated crack onto randomly selected background images from the ImageNet database. This ensures that the model is trained in a diverse environment, reducing the risk of overfitting to a particular domain and enhancing the robustness of the trained crack detection system.

2. State of the Art

One of the biggest challenges in the application of AI technologies lies in the availability of data [1]. Although datasets of real cracks are available, they are often too small, specialized in one domain, or have a strong bias. This can affect the angle of view, the characteristics of cracks, the frequency of crack types, or the material being analyzed.

In contrast to real data generation, synthetic data generation using rendering software offers advantages, such as a high degree of automation, simple scalability, reduction and analysis of dataset bias, as well as the integration of various methods of crack data synthesis.

One of the most prominent methods of crack data synthesis is Perlin Noise [3–5]. Here, a depth map is generated using a chaotic mathematical process, and partial areas are cut out of this using thresholding and then rendered as a representation of a crack. One issue with this approach is that users have access to a wide range of parameters to adjust the appearance of cracks, which may not be directly related to the physical characteristics of the cracks. The selection of parameters is heavily influenced by the user's perception of cracks, which can introduce unwanted bias into the dataset.

In sharp contrast to Perlin Noise, cracks can be generated using the finite element method (FEM) [6]. This forms a strong basis for simulating physically correct crack propagation and simulating a large number of parameters such as the influence of force, material properties, and load cycles. Developing the simulation requires a high level of expertise. A small change in the initial conditions can have a considerable influence on the result, furthermore, the size of the generated dataset is limited by the high computational effort.

Another method of generating synthetic crack datasets is to augment existing datasets. In addition to classic methods such as rotation, scaling, and cropping, as used in the training of neural networks, it is also possible to use cut-and-paste methods [7] or generative approaches. The former approach uses available data, cuts out existing cracks based on segmentation masks, and inserts them into new image material. The latter approach uses neural networks in the form of GANs [8] to learn the appearance of cracks based on real crack data and synthesize deceptively real data. Both approaches are only applicable when real-world data is available. Although the performance of neural networks improves when using these techniques, the question arises as to the degree of novelty of the generated data and the extent to which an existing dataset bias is transferred to the newly generated data.

With each of the methods presented, it should be noted that a domain or perception gap arises. This is primarily characterized by the fact that each simulation or replication only represents a reduced subset of reality and can therefore lack important features. To counteract the problem of the domain gap, it is possible, for example, to replicate the target domain in detail or to use domain randomization to make the training dataset as diverse as possible and to represent desired features in a variety of manifestations so that neural networks can learn them more generally [9].

Domain randomization can be divided into two different paths. On the one hand, the structured domain randomization method can be used, in which rules are developed under which a scene is to be constructed and which logical relationships are to be observed [10]. This approach is suitable for target domains where causalities must be adhered to fulfill a task. In complete contrast to this is the full or broad domain randomization [11], in which a few features are to be presented in as many permutations as possible and a logical connection is of less importance. The goal is to maximize robustness against interference sources and design a training domain as large as possible to include the target domain in the training data.

Although machine learning is already used in crack inspection and enables tasks such as semantic segmentation, classification, the creation of bounding boxes, or the calculation of a severity index for the degree of damage, one of the main problems is the provision of training data in sufficient quantity and quality [1].

There is currently no best practice method available and approaches for synthetic data generation that have already been tested prove to be insufficiently comparable with each other. In addition, there is still a need for research into individual methods, such as how to reduce user bias or to what extent parameter fine-tuning influences the quality of synthesized datasets.

Figure 1: Example images of the generated cracked images including the segmentation mask to illustrate the different crack shapes generated during dataset synthesis

3. Synthetic Crack Dataset Generation

The process of creating a synthetic crack dataset in the 3D rendering suite Blender [12] begins with the creation of a straight surface that serves as the base for the cracks. Once the surface is ready, a texture-based approach, using shader nodes, is employed to synthesize crack data. The shader node consists of two essential elements: one for background generation, where a random image from the ImageNet [13] database is selected for each generated environment, and another for crack generation, using the shader node to design the shape and appearance of the cracks through Perlin Noise [14].

To ensure a diverse range of crack shapes and minimize dataset bias, the range of crack parameters is defined and randomized using a broad approach with an even distribution of the defined space. The generated cracks are then transferred to the background image so that the cracks are displayed as darker pixels. This is intended to imitate the behavior of cracks so that they reflect less incident light and consequently make the material less visible.

To reduce the impact of varying lighting and camera positions on later AI model training and to better observe the effect of the crack parameters, both are placed in a fixed position above the surface.

For optimal rendering quality, the Cycles rendering engine is selected to produce photo-realistic results. Finally, a rendered image is captured from a bird's eye view, which includes the crack annotation.

4. Crack Mask Analyses

Since Perlin Noise is a mathematical method that is not primarily related to physical crack propagation, it is of central importance to analyze the generated crack data and transfer the characteristics of generated crack shapes to parameters of real cracks. The generated segmentation masks serve as the basis for analyzing the bias of the dataset. Each Image in the dataset undergoes the following algorithm and consists of the following steps:

- Creating skeletal structures of the crack: Skeletal structures are created to identify the course and centerline of cracks. This simplifies the crack representation and enables further analysis.
- Exclusion of small cracks: Cracks consisting of only three pixels or less are excluded from the analysis, as they are considered insignificant or noise
- Measuring crack width: The crack width is measured for each pixel of the generated skeleton. The average crack width and the resulting standard deviation are determined to characterize the crack width distribution.
- Determining crack length: The crack length is determined based on the number of pixels in the skeleton. This provides an estimate of the extent of each crack.
- Identifying bifurcations: The skeleton is graphically represented to identify bifurcations, which are points where a crack splits into multiple branches.
- Calculating crack curvature: The degree of curvature of each crack is determined. This parameter quantifies the deviation of the crack path from a straight line.

After the analysis, the individual images can be differentiated based on their crack characteristics, revealing information that enables a deeper understanding of the data. The analysis provides insights into several key characteristics. These include average crack width, curvature, and resulting standard deviation. It also examines crack length, number of individual cracks, bifurcations, and the ratio of background to crack pixels. By quantifying these properties and matching them with real crack parameters, the analysis forms the foundation for comparing images within a dataset and drawing meaningful conclusions about the effect of crack parameters on the overall training results.

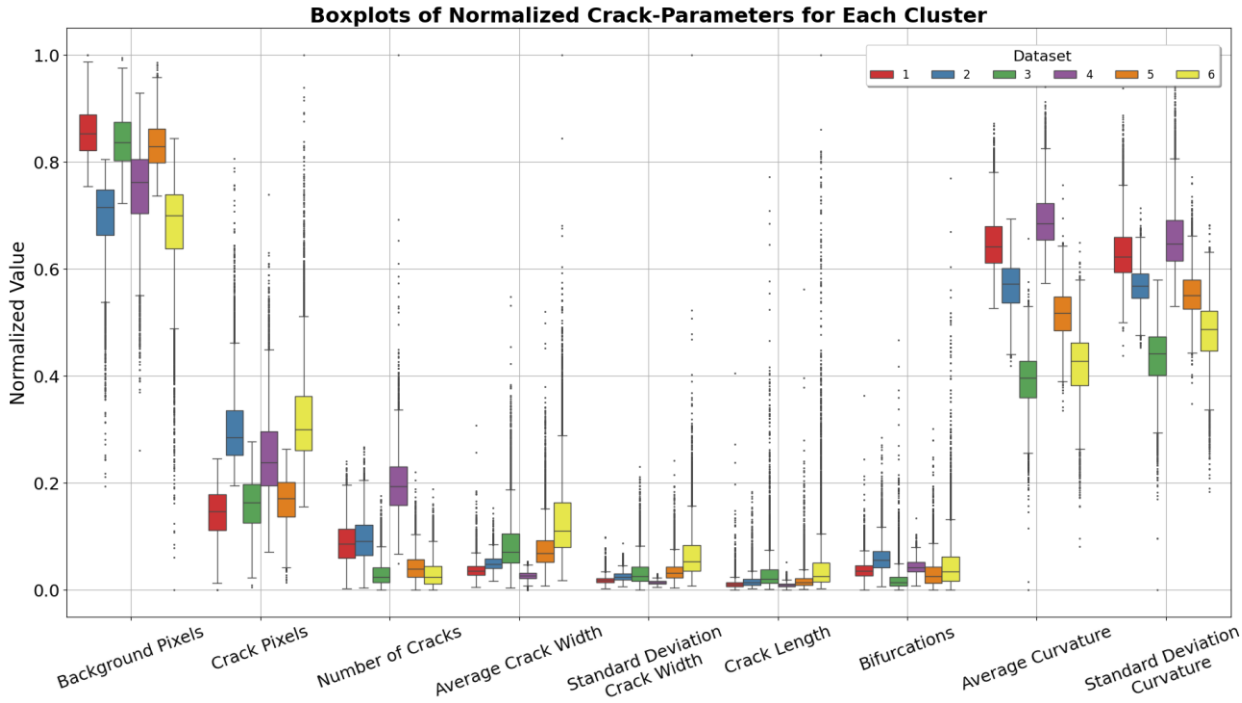

Figure 2: Boxplot representation of the six datasets created by Partitioning Around Medoids (PAM), based on the crack parameters determined in chapter 4

5. Dataset Partitioning

To thoroughly examine the algorithm's performance under diverse conditions, we used the Partitioning Around Medoids (PAM) algorithm to generate test datasets that capture a wide range of crack parameter variations. PAM creates clusters based on the extreme values of the dataset's parameters, maximizing the diversity of crack characteristics in each subset.

We started with an initial dataset of 40,000 images, generated with broad randomization overall crack parameters. We then normalized the analyzed crack mask parameters and strategically distributed the images into six distinct datasets, each containing 5,000 images. To optimize the clustering process and achieve a balanced distribution of crack parameters across the subsets, we used evaluation metrics such as the sum of the squared Euclidean distance (SSE) and Silhouette Score (Figure 3). The SSE measures the compactness of clusters, with lower values indicating better clustering, while the Silhouette Score assesses both the compactness and separation of clusters, with higher values suggesting better-defined clusters. By considering these metrics, we ensure that the resulting datasets are well-structured and representative of the diverse crack characteristics present in the original dataset.

Figure 4 illustrates the differences between the individual datasets using a boxplot with normalized crack parameters, allowing clear visualization of the relative differences between the datasets and the distribution of the range of crack parameters within a dataset.

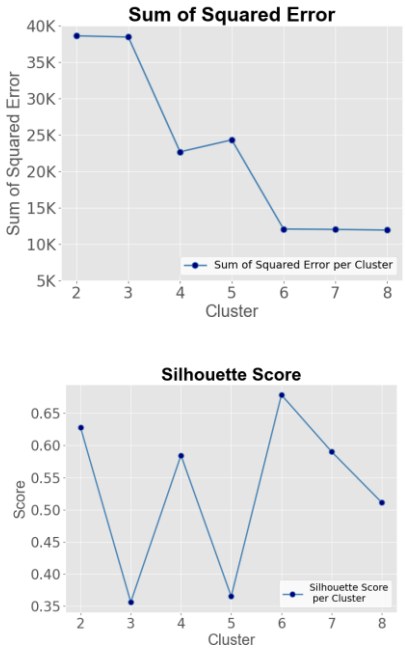

Figure 3: Analysis of the optimal cluster distribution for the creation of the most differentiated datasets, Sum of Squared Error (left) and Silhouette Score (right)

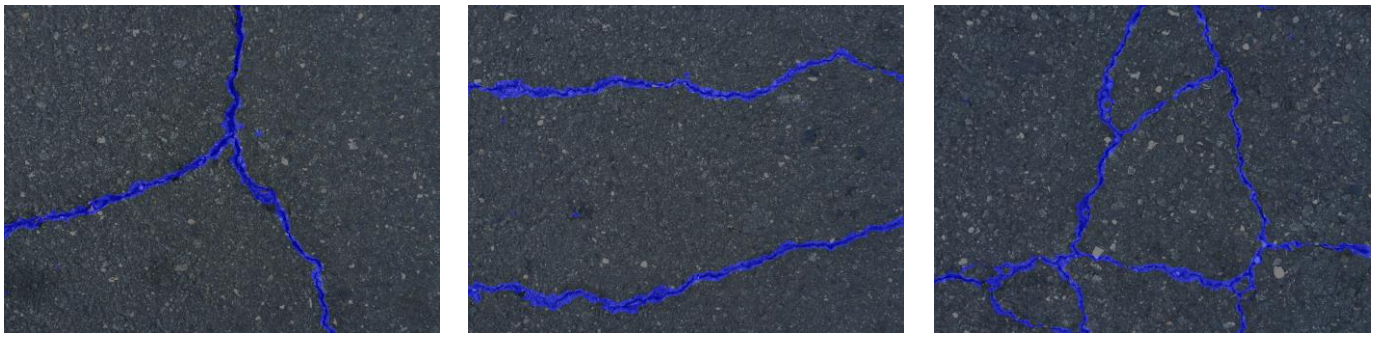

Figure 4: Inference results on real world crack 500 dataset. Best performing network, trained on dataset 6

6. Training & Evaluation

Table 1: Segformer5b inference results on the Crack 500 Dataset, based on different synthetic training datasets used during finetuning

Parameter	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
IoU	0.09	0.1	0.16	0.03	0.17	0.29
Pixel Accuracy	0.95	0.92	0.96	0.97	0.97	0.96
Precision	0.17	0.15	0.28	0.02	0.34	0.37
Recall	0.19	0.27	0.29	0.04	0.288	0.64
F1-Score	0.17	0.17	0.26	0.07	0.27	0.43

For testing the SegFormer 5b [15] model was deployed, a state-of-the-art neural network architecture to perform pixel-accurate semantic segmentation on images. The pre-trained version NVIDIA5b was fine-tuned using our synthetic crack datasets to adapt it to the specific task of crack segmentation. The training process was conducted using 5,000 images for a single epoch, and this procedure was repeated for each of the six datasets. To optimize computational resources and maintain consistency, the image material was reduced to a resolution of 512x512 pixels. The segmentation mask was designed to have two classes: background and crack, enabling the network to distinguish between these two.

After the training phase, the performance of the fine-tuned SegFormer 5b model was tested on the crack500 [15] dataset (pavement cracks) to evaluate its effectiveness in a real-world scenario. For this purpose, the metrics shown in Table 1 were calculated. The Intersection over Union (IoU) [16] describes the overlap of the calculated segmentation mask for the crack class with the manually created segmentation mask. Precision and recall [17] are important metrics that evaluate the accuracy and completeness of the crack class. Precision indicates how many of the pixels classified as cracks are cracks, while Recall indicates how many of the actual crack pixels were recognized by the model. The F1 [17] score is the harmonic mean of Precision and Recall and provides a balanced assessment of model performance for the crack class. Pixel Accuracy [17], on the other hand, refers to both the "Crack" class and the "Background" class. It indicates how many pixels were correctly classified in total, to measure the overall performance of the model in classifying the image pixel by pixel. Due to unknown label noise in the validation data, it should be noted that the trend of the results is significantly more important than the absolute results.

7. Effect of Crack Parameters on Performance Results

The central contribution of this work lies in the determination of correlations between crack parameters within the training data and the subsequent performance of neural networks in crack detection. Linear regression was employed to identify which crack parameters have a positive or negative influence

Table 2: linear regression results of mapping crack parameters to evaluation metrics. Each element consists of the **r**/**p** values for correlation and probability.

Parameter	IoU	F1 score	Pixel accuracy	Precision	Recall
Crack width μ	0.96/ 0.01	0.95/ 0.01	0.26/ 0.68	0.74/ 0.15	0.84/ 0.07
Crack width σ	0.97/ 0.01	0.97/ 0.01	0.17/ 0.79	0.73/ 0.16	0.89/ 0.04
Crack length	0.78/ 0.12	0.78/ 0.12	0.15/ 0.80	0.49/ 0.40	0.72/ 0.17
Background pixel	0.51/ 0.38	0.50/ 0.39	0.53/ 0.35	0.53/ 0.36	0.26/ 0.68
Crack pixel	-0.51/ 0.38	-0.50/ 0.39	-0.53/ 0.35	-0.53/ 0.36	-0.26/ 0.68
Bifurcations	-0.56/ 0.32	-0.55/ 0.34	-0.76/ 0.14	-0.74/ 0.15	-0.25/ 0.69
Curvature σ	-0.81/ 0.09	-0.81/ 0.09	-0.11/ 0.86	-0.49/ 0.40	-0.77/ 0.12
Curvature μ	-0.88/ 0.05	-0.87/ 0.05	-0.15/ 0.81	-0.58/ 0.31	-0.82/ 0.09
Number of cracks	-0.97/ 0.01	-0.98/ 0.00	0.04/ 0.95	-0.49/ 0.40	-0.95/ 0.01
Dropped cracks	-0.96/ 0.01	-0.97/ 0.01	0.06/ 0.93	-0.47/ 0.43	-0.95/ 0.01

on the model's performance as shown in Table 2. The analysis revealed several key findings:

Firstly, crack width (both mean and standard deviation) exhibited very strong positive correlations with average IoU (r=0.96 and r=0.97, respectively) and average F1 score (r=0.95 and r=0.97), suggesting that as crack width increases, the model's performance in detecting and segmenting cracks improves as well.

Conversely, the number of cracks and dropped cracks showed very strong negative correlations with average IoU (r=-0.97 and r=-0.96), average F1 score (r=-0.98 and r=-0.97), and average recall (r=-0.95 for both), indicating that a higher number of cracks and dropped cracks can negatively impact the model's performance.

Additionally, crack length demonstrated strong positive correlations with average IoU (r=0.78) and average F1 score (r=0.78), while curvature (both mean and standard deviation) showed strong to very strong negative correlations with most performance metrics, implying that shorter and more curved cracks may pose challenges for accurate detection.

Background pixels and crack pixels exhibited moderate correlations with average IoU, average F1 score, and average precision, suggesting some influence on the model's performance. Lastly, the number of bifurcations had strong negative correlations with average pixel accuracy (r=-0.76) and average precision (r=-0.74), indicating that more bifurcations may negatively impact these specific metrics.

In summary, the linear regression analysis revealed that crack width and length have a very strong positive influence on the model's performance, while the number of cracks, dropped cracks, and curvature negatively impacts its accuracy. While background/crack pixels exhibited moderate correlations with the performance metrics

8. Summary

In this paper, we have investigated the potential of using Perlin Noise as a basis for modeling surface cracks in synthetic datasets to train crack detection models. By mapping the parameters of Perlin Noise generated cracks to the physical properties of cracks, such as width, length, curvature, and bifurcations, we have established a relationship between the process of generating synthetic data and the performance of trained neural networks.

The SegFormer 5b model was fine-tuned using different parameterized synthetic crack datasets and benchmarked against the Crack500 dataset to evaluate its performance in a real-world scenario. The results show that the design of the training datasets has a significant impact on the model's ability to segment cracks.

Linear regression analysis was used to identify the key crack parameters that influence the performance of the model. Crack width and length showed strong positive correlations with the performance metrics used. Conversely, a higher number of individual cracks and strong curvature showed strong negative correlations. Future work could explore the integration of additional crack modeling techniques such as finite element methods to further improve the realism and diversity of synthetic datasets. Furthermore, investigating the impact of domain adaptation techniques to bridge the gap between synthetic and real data could contribute to more effective crack detection models in different industries.

References

1. Chakurkar PS, Vora D, Patil S et al. Data-driven approach for AI-based crack detection: techniques, challenges, and future scope. Front Sustain Cities 5 (2023)
2. Rill-García R, Dokladalova E, Dokládal P Syncrack: Improving Pavement and Concrete Crack Detection through Synthetic Data Generation. In: Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. SCITEPRESS - Science and Technology Publications, pp 147–158 (2022)
3. Boikov A, Payor V, Savelev R et al. Synthetic Data Generation for Steel Defect Detection and Classification Using Deep Learning. Symmetry 13:1176 (2021)
4. Guanghao Zhai, Yasutaka Narazaki, Shuo Wang, Shaik Althaf V. Shajihan, Billie F. Spencer Jr. Synthetic data augmentation for pixel-wise steel fatigue crack identification using fully convolutional networks
5. Narazaki Y, Hoskere V, Yoshida K et al. Synthetic environments for vision-based structural condition assessment of Japanese high-speed railway viaducts. Mechanical Systems and Signal Processing 160:107850 (2021)
6. Hoskere V, Narazaki Y, Spencer BF Physics-Based Graphics Models in 3D Synthetic Environments as Autonomous Vision-Based Inspection Testbeds. Sensors (Basel) 22 (2022)
7. Kanaeva IA, Ivanova JA Road pavement crack detection using deep learning with synthetic data. IOP Conf Ser.: Mater Sci Eng 1019:12036 (2021)
8. Jain S, Seth G, Paruthi A et al. Synthetic data augmentation for surface defect detection and classification using deep learning. J Intell Manuf 33:1007–1020 (2022)
9. Tobin J, Fong R, Ray A et al. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World (2017)
10. Schoepflin D, Iyer K, Gomse M et al. Towards Synthetic AI Training Data for Image Classification in Intralogistic Settings. In: Schüppstuhl T, Tracht K, Raatz A (eds) Annals of Scientific Society for Assembly, Handling and Industrial Robotics 2021. Springer International Publishing, Cham, pp 325–336 (2022)
11. Sun B, Saenko K From Virtual to Reality: Fast Adaptation of Virtual Object Detectors to Real Domains. In: Valstar M, French A, Pridmore T (eds) Proceedings of the British Machine Vision Conference 2014. British Machine Vision Association, 82.1-82.12 (2014)
12. Deng J, Dong W, Socher R et al. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp 248–255 (2009)
13. Foundation B blender.org - Home of the Blender project - Free and Open 3D Creation Software. https://www.blender.org/. Accessed 08 Apr 2024 (2024)
14. YouTube Blender Tutorial - Procedural Cracked Surface Material. https://www.youtube.com/watch?v=XFDQrdRmDwc&ab_channel=BlenderMadeEasy. Accessed 08 Apr 2024 (2024)
15. Xie E, Wang W, Yu Z et al. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers (2021)
16. Zhang, Lei and Yang, Fan and Zhang, Yimin Daniel and Zhu, Ying Julie Crack 500. https://www.kaggle.com/datasets/pauldavid22/crack50020220509t090436z001. Accessed 26 Jun 2022 (2016)
17. Wikipedia Jaccard index. https://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=1196092673. Accessed 09 Apr 2024 (2024)
18. David M W Powers Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Evaluation: From Precision, Recall and F-Factorto ROC, Informedness, Markedness & Correlation (2007)