

Tuning domain randomization for 3D-rendered synthetic training data for defect detection in aircraft engine borescope inspection

Ole Schmedemann[✉],* Dirk Holst, Jorrid Lund, and Thorsten Schüppstuhl

Hamburg University of Technology, TUHH, Institute of Aircraft Production Technology, Hamburg, Germany

ABSTRACT. Visual surface defect detection in complex industrial inspection scenarios remains predominantly manual. Deep learning-based vision systems promise to automate this task but require large quantities of representative, annotated defect images, which are often unavailable due to the rarity and diversity of relevant defects. We address this challenge with a modular synthetic data generation pipeline leveraging physically based 3D rendering, which enables systematic domain randomization throughout the entire rendering chain. The pipeline comprises dedicated modules for scene composition, geometry-based defect generation, procedural material modeling, rendering, and automatic pixel-wise annotation, allowing users to expose and tune key parameters such as defect size, position, and penetration depth, object materials, camera poses, and lighting. We first provide a structured analysis of all process steps that influence image appearance in defect synthesis and then instantiate the pipeline for a case study on aircraft engine borescope inspection, targeting missing-material defects on high-pressure compressor blades. Using a two-stage experimental design, we employ a Plackett–Burman screening over 22 candidate factors followed by a one-factor-at-a-time optimization of the most influential parameters. Our results show that the material model, defect positioning, defect size, and penetration depth randomization dominate detection performance. Training an object detector solely on synthetic images generated with the optimized configuration yields a maximum of 0.89 mAP@0.5IoU on a real-world borescope validation set, with performance saturating at ~500 rendered training images. These findings demonstrate that carefully structured domain randomization in a physically based rendering pipeline can provide practically useful synthetic training data for industrial defect detection.

© The Authors. Published by IS&T and SPIE under a Creative Commons Attribution 4.0 International License. Distribution or reproduction of this work in whole or in part requires full attribution of the original publication, including its DOI. [DOI: [10.1117/1.JEI.35.4.043008](https://doi.org/10.1117/1.JEI.35.4.043008)]

Keywords: synthetic training data; machine learning; deep learning; visual inspection; industrial quality control; domain randomization; rendering; defect detection

Paper 251832G received Dec. 17, 2025; revised Apr. 30, 2026; accepted Jun. 1, 2026; published Jul. 15, 2026.

1 Introduction

Deep learning-based computer vision has emerged as a promising approach for automating defect detection in complex industrial inspection tasks. In principle, training deep networks requires substantial quantities of representative image data to capture the range of defect types, appearances, and locations encountered in real-world settings. However, the scarcity of certain defects or those that manifest infrequently often limits the collection of sufficiently diverse training datasets. This mismatch between the high data demands of deep learning and the limited

*Address all correspondence to Ole Schmedemann, ole.schmedemann@tuhh.de

availability of real-world defect images remains a key bottleneck for deploying machine vision in industrial quality inspection.

Synthetic image generation via 3D rendering has gained increasing attention as a way to address this challenge. By constructing virtual scenes, placing virtual cameras and lights, and embedding programmatically generated defects, one can systematically generate training images tailored to a specific application. This approach allows engineers to incorporate domain knowledge about plausible defect shapes, sizes, and locations while exercising precise control over surface appearance, background, and illumination. Synthetic images can thus complement or partially replace real-world data and, in some scenarios, enable training even when only a small number of defect examples are available.

Despite these advantages, transferring models trained on synthetic images to real-world data is far from trivial. Differences in content and appearance between rendered and real images, which are often summarized as the sim-to-real or domain gap, can cause substantial performance degradation when models are deployed. Existing work has explored two main strategies to mitigate this gap. One line of research focuses on highly realistic rendering pipelines that aim to approximate sensor characteristics, material behavior, and lighting as closely as possible. A complementary strategy is domain randomization (DR), in which a wide range of visual properties (e.g., textures, lighting, geometry, background clutter) is deliberately varied, sometimes beyond physically realistic bounds, to force the model to learn domain-invariant features. For defect detection, several studies have reported that appropriately designed synthetic data can yield competitive performance,^{1–3} yet most of these works focus on specific use cases and provide limited guidance on how to configure a new synthetic pipeline.

For a new defect detection application, practitioners are typically confronted with a large number of design choices along the rendering pipeline: the level of geometric detail in the scene, how to parameterize defect geometry, which material models to employ, how to place cameras and lights, and how strongly to randomize each of these elements. In practice, these choices are often made based on engineering intuition and ad hoc trial and error. Systematic investigations of which pipeline parameters matter most for task performance, and over which ranges they should be varied, are still relatively rare. As a result, it remains unclear how to best balance realism and variation for a given industrial inspection task and how to prioritize modeling effort under limited engineering resources.

In this work, we take a step toward addressing this gap for geometry-based defect generation in industrial surface inspection. We develop a configurable 3D rendering pipeline that exposes key stages of scene setup, defect modeling, material definition, camera and lighting configuration, and automatic annotation. Building on this pipeline, we employ a simple yet structured experimental design to study how selected rendering parameters influence the performance of object detectors trained solely on synthetic images. Concretely, we make the following contributions:

- We implement a modular Blender-based rendering pipeline for defect image generation that enables explicit control and randomization of scene composition, geometry-based defect creation, material properties, camera placement, and lighting.
- We apply a two-stage design-of-experiment approach (Plackett–Burman⁴ screening followed by one-factor-at-a-time (OFAT) refinement) to identify a subset of practically influential rendering parameters and to tune their ranges for a specific defect detection task.
- We instantiate the approach in a case study on borescope inspection of aircraft engine compressor blades with missing-material defects, demonstrating that detectors trained solely on appropriately configured synthetic datasets reach competitive performance on a real-world validation set.

The remainder of the paper is organized as follows. Section 2 reviews related work on synthetic data generation and domain randomization for defect detection. Section 3 analyzes the main process steps in the defect image generation pipeline. Section 4 introduces the proposed rendering pipeline, and Sec. 5 presents the experimental setup and results for the borescope inspection case study. Section 6 discusses limitations and practical implications, and Sec. 7 concludes with an outlook on future work.

2 Related Work

In this chapter, an overview of synthetic data generation approaches for visual defect tasks is presented. We refer to synthetic images in general when images are computer-generated or significantly manipulated by computers. Different methods for generating synthetic image data for application in computer vision are used, which can be roughly grouped into *generative learning methods*, *rule-based 2D image composition*, and *rule-based 3D rendering*.

2.1 Methods for Generating Synthetic Image Data

2.1.1 Generative learning methods

This family of approaches uses machine learning models, such as generative adversarial networks (GANs), variational autoencoders, or diffusion models, to create synthetic images. In the context of industrial inspection, GAN-based schemes are utilized to insert defects into nominal (defect-free) images, effectively augmenting limited datasets.^{5–10} Other works employ convolutional autoencoders (CAEs) for a similar purpose.^{11,12} Some recent studies explore text-to-image diffusion probabilistic models (DPMs),¹³ such as DALL-E¹⁴ or Imagen,¹⁵ to generate training images from textual prompts.¹⁶

Despite the often highly realistic visual quality these methods can achieve, they have several drawbacks when used for defect detection. First, they rely on training data of the same domain, which does not solve the fundamental problem of having few or no real-world defect images. Second, they limit user control over specific defect parameters such as size, depth, or exact location and thus cannot easily incorporate extensive domain knowledge. On the contrary, these models are prone to unwanted artifacts. Third, because the generation mechanism is image-based, the method does not automatically produce pixel-level ground-truth annotations: one must rely on manual or semi-automatic labeling to create large-scale training sets. Consequently, ML-based generation may be viewed as a form of advanced data augmentation rather than a general synthetic data solution.¹⁷

2.1.2 Rule-based 2D image composition

Image composition approaches manipulate two-dimensional image data to create synthetic variations. For instance, *cut-and-paste*¹⁸ strategies superimpose new objects, background regions, or defect patches onto existing images.^{19–21} Other examples are procedurally generated images,²² manipulating real-world images, or combining real-world and procedurally generated images.^{23–25}

Although these rule-based techniques are straightforward to implement, they can struggle to produce physically accurate lighting, shadows, or material changes. A recurring challenge is the inability to simulate geometric deformations. If a defect implies missing or protruding material, simple 2D overlays will not account for occlusions or changes in shading. Moreover, when an object's shape is significantly altered, 2D compositing cannot reveal what lies behind newly created holes or cracks. As a result, rule-based 2D rendering often suffices for minor surface anomalies but fails to capture more complex defects or detailed lighting interactions.

2.1.3 Rule-based 3D rendering

Rule-based 3D rendering addresses these shortcomings by constructing a synthetic scene comprising 3D geometry, camera, and lighting, and rendering it. This technique can yield high-fidelity images and automatically produce pixel-accurate annotations. Although it requires more effort, 3D rendering provides greater realism and better handles occlusions, highlights, and shadows.

3D rendered datasets have been produced for various applications, from object detection^{26–28} to robotic vision.²⁹ General-purpose toolboxes such as Kubric,³⁰ BlenderProc (<https://github.com/DLR-RM/BlenderProc>),³¹ NDDS (https://github.com/NVIDIA/Dataset_Synthesizer), CAD2Render,³² Infinigen,³³ and Perception (<https://github.com/Unity-Technologies/com.unity.perception>) ease the process of generating annotated images. These frameworks typically assume that accurate 3D models are readily available. Although CAD models of the objects under investigation are often available for industrial inspection tasks, these models represent only the defect-free state. Consequently, they cannot be used directly to generate images of defective parts.

2.2 Minimizing the Domain Gap

Synthetic image data can differ significantly from real-world images, leading to reduced performance of models trained with this data when applied to real-world data. This observation is commonly referred to as the *domain gap*,³⁴ *sim-to-real-gap*, or *reality gap*. This is due to rendering engines only approximating real-world cameras, and real-world light interactions with surfaces, and that 3D models are solely approximations of their real-world counterparts (appearance gap). Furthermore, synthetic objects and scenes are often idealized or have reduced detail compared with real scenes, and the probability of occurrence and placement of objects does not correspond to the real world, sometimes referred to as the content gap. Two main strategies are used to mitigate this issue.

2.2.1 Sensor-realistic rendering

One approach is to push rendering fidelity as far as possible by using detailed 3D models, advanced material shaders, complex lighting setups, and even sensor-specific artifacts. Reference 35 shows that adding more realistic materials can improve object recognition performance. In robotics, physically based simulators for self-driving or manipulation tasks also adopt high-quality renders to reduce the sim-to-real gap.³⁶

However, realism alone can be costly and may risk overfitting to the specific synthetic environment. Minor errors in reflectance models or missing microgeometry might lead to biases that neural networks exploit in synthetic data but fail on real-world test images.

2.2.2 Domain randomization

Domain randomization (DR) is a technique that generates synthetic image data with a wide array of randomized visual properties to serve as training material for deep learning-based computer vision models. By systematically varying elements such as textures, lighting conditions, object placements, and background details, this approach compels models to focus on underlying, intrinsic features rather than superficial visual cues. In doing so, the model learns to extract a domain-invariant representation that remains robust and effective even when confronted with the variability and unpredictability of real-world data.^{26,37,38} An advantage of this approach is scalability: because the focus is on coverage rather than perfect realism, generating large quantities of randomized images is often faster and less resource-intensive than meticulously crafting photorealistic scenes. When randomness is structured (i.e., physically plausible, context-aware), models can still reap the benefits of coverage while preserving realism.³⁹

Effective DR pipelines often blend physically accurate rendering (materials, shadows) with broad variation (texture randomization, camera jitter, lighting changes). The consensus is that the ideal balance between sensor realism and randomization is highly application-specific, requiring empirical tuning.³⁶

2.3 Synthetic Data for Defect Detection

A growing body of research focuses on specialized pipelines for defect image synthesis. One can differentiate between the underlying defect-introducing principle used; although geometry-based methods directly alter the 3D mesh, texture-based methods^{3,40–50} modify surface appearance without significantly changing the mesh geometry.

2.3.1 Geometry-based defect generation

An example of a geometry-based defect generation is Ref. 51. They introduced a procedural crack model that subtracts geometry along fracture lines. Early work by Refs. 52 and 53 used parametric shapes and physics simulations to produce realistic glass shards, with cracks adapted from prior procedural models.⁵⁴ Similarly, Ref. 55 advocates a structured pipeline of 3D simulation and parametric modeling to recreate cracks on microchips.

References 1, 56, and 57 propose a pipeline that programmatically creates flaws in parts by Boolean operations in the object's mesh. These geometry modifications (e.g., dents, scratches) are rendered under physically based illumination, yielding photorealistic images. In previous work, we introduced a similar approach to imprinting defect negative objects in cast-iron parts and extensively randomizing lighting and camera parameters.⁵⁸ Recent research by Refs. 59 and 60 explores multiview setups, emphasizing the importance of capturing consistent defect

geometry across different viewpoints. Meanwhile, Ref. 2 systematically varies mesh changes to represent flash defects on injection-molded parts, highlighting that camera and lighting variation can profoundly affect the final model performance even for a single defect type.

The main advantage of geometry-based approaches is physical plausibility: lighting, self-occlusion, and shadows emerge naturally from the shape modifications. However, modeling complex material removal or cracks may be computationally demanding.

2.3.2 Texture-based defect generation

Texture-based approaches either modify the material, thereby locally altering its reflectance properties and creating deviations from the defect-free regions, or they adjust the geometry exclusively during rendering, for example by employing textures for virtual displacement mapping.⁶¹ Previous work leverages 2D textures or shader manipulations to represent defects such as cracks, stains, or corrosion. For example, Ref. 40 generates procedural defect textures (cracks, foreign inclusions) on a steel surface, applying local displacement maps to mimic slight geometric disturbance. Reference 41 overlays defect masks on a car door's texture, randomly varying roughness and normal maps. Similarly, Ref. 42 combines photogrammetry with photometric stereo to capture real surface textures, then adds blowhole-like defects via local noise-based displacement.

Texture-based methods often suffice for small or shallow defects, such as scratches or corrosion, and can be more efficient than large Boolean operations. However, they cannot accurately depict scenarios where significant geometry changes reveal occluded areas (holes, breakages) unless combined with partial geometry editing. Several authors thus advocate a hybrid pipeline, where texture-based shading changes handle subtle scratches, and mesh-based operations handle bulk deformations.^{3,48}

2.3.3 Open challenges

Despite these advancements, open questions remain. First, many works use synthetic data to show anecdotal or case-specific success but do not systematically quantify how each pipeline parameter (e.g., lighting, mesh detail) contributes to final accuracy. It was not examined holistically how to consider and analyze all potentially influencing steps in generating a synthetic dataset. Pipelines are often tailored to a single part geometry or defect type. Generalizing to new surfaces, new lighting configurations, or new defect physics can require nontrivial engineering.

2.4 Summary and Our Contribution

Efforts to minimize the domain gap between synthetic and real images in defect detection largely fall into two categories: highly realistic rendering and domain randomization. Realistic rendering aims to approximate sensor behavior, material response, and illumination as closely as possible, whereas domain randomization deliberately varies visual properties such as textures, lighting, geometry, and background content to encourage robustness. For industrial surface inspection, both geometry-based and texture-based defect creation strategies have been shown to be feasible, but most existing pipelines are tailored to specific parts, defect types, or inspection setups and provide limited guidance on how to configure a new application. Most existing pipelines do not systematically vary the rendering pipeline to optimize defect recognition.

In this work, we address this gap for geometry-based defect generation in industrial surface inspection. Building on the process steps outlined above, we first analyze which stages of a rendering pipeline are likely to influence the performance of deep learning-based defect detectors, with a focus on parameters that can be explicitly controlled or randomized. Based on this analysis, we implement a modular, Blender-based rendering pipeline that exposes key configuration options for scene composition, geometry-based defect modeling, material definition, camera placement, and lighting, and automatically produces pixel-accurate annotations. This structure enables practitioners to systematically adjust the realism and variation of generated datasets by editing compact configuration files rather than modifying the underlying implementation.

Finally, we instantiate the proposed pipeline in a representative case study on borescope inspection of aircraft engine compressor blades with missing-material defects. Using a Plackett–Burman⁴ screening followed by an OFAT refinement, we empirically identify a subset of influential rendering parameters and tune their ranges to improve object detection performance on real borescope images, using only synthetic training data. Although the present work focuses

on geometry-based defects, the pipeline is designed to be extendable with texture-based defect generation methods in future studies.

3 Analysis of Influencing Process Steps

The degree of realism and domain randomization required varies by application and must be determined case by case. To generate a suitable synthetic data set, researchers and practitioners must understand which image characteristics can potentially enhance model performance and employ a tool capable of producing data sets with these features. Building on the clustering approach described by Paulin et al.,¹⁷ we adapt the categorization of all potentially influencing steps along the image generation pipeline specifically to defect generation, see Fig. 1. Each step influences the appearance of the images and thus may affect the quality and suitability of the final data set.

For the rendering pipeline, these steps include origin and manipulation of scene objects, generating a defect, additional scene elements such as cameras and lighting, composing all objects into a 3D scene, setting shader properties for all objects, and finally rendering an image. In addition, postrendering steps include 2D image postprocessing, which is sometimes used to mimic imaging errors such as vignetting or distortions, image augmentations, which include, among others, simple geometric transformations to introduce more variation, and domain adaptation, where image-to-image machine learning models are used to increase realism of renders. In this chapter, we present the influencing properties of the steps of the rendering pipeline (dark gray).

3.1 Objects

In industrial production, CAD models of objects or entire environments are often readily available; however, when they are missing due to incomplete product variants, lack of OEM-provided designs, or the absence of detailed semifinished product data, 3D scanning or photogrammetry can fill the gap by reconstructing an object's geometry from multiple photographs or through structured-light and laser-based measurements. Although both methods can produce accurate models, they typically require controlled conditions, specialized equipment, and postprocessing. Photogrammetry may struggle with reflective or transparent surfaces, whereas structured-light and laser scanners handle them better. When background details have minimal impact on defect appearance, omitting detailed 3D models of additional objects and using random background environment maps (e.g., HDRI images) can be a cost-effective way to vary the scene and reduce false positives outside the primary object of interest.

Besides the presence of objects and the choice of background environment maps, the level of detail of objects directly influences their appearance in rendered images. If details present on the real-world parts are not present on the virtual object, the model may not learn to distinguish between them and the defect and may recognize them as false positives.

3.2 Defect Generation

Geometry-based methods directly modify the geometry of an object, resulting in more realistic lighting and shadows when the viewpoint or illumination changes. For this approach, a defect model is either subtracted or added to the main object's geometry using Boolean operations to simulate the defect's geometric appearance. Typical examples include dents, nicks, cracks,

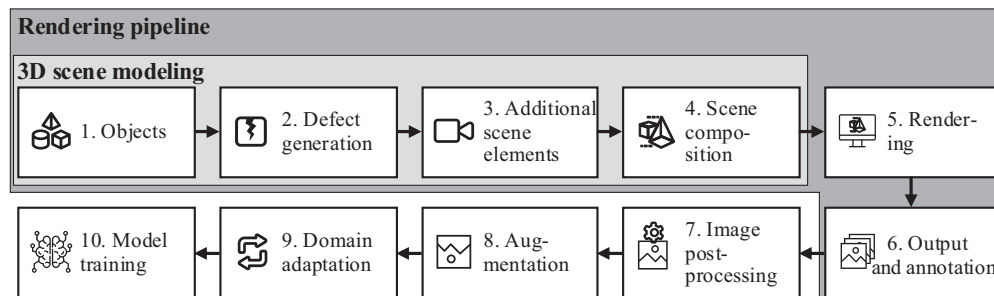


Fig. 1 Schematic representation of the synthetic data set generation workflow, incorporating image synthesis steps that ultimately influence the final model's performance in defect detection.

missing geometry, or any other defect that significantly alters the shape of an object. Object-based defect generation steps consist of the defect object's generation, positioning, manipulation, and imprinting. Defect objects may originate from manual modeling, procedural modeling, physical modeling, or 3D-scanning of real-world samples. Both the generation steps and the origin of objects significantly influence the appearance of the defect. Thus, it must be possible to examine their impact.

3.3 Additional Scene Elements

Modern rendering engines typically employ a pinhole camera model and advanced lighting simulation to replicate real-world optical conditions. These components can be configured and randomized to enhance the fidelity and variability of synthetic data sets. Virtual cameras have image resolution and focal length as the most influential properties. These parameters directly impact the size of scene objects in terms of pixels.

Accurate lighting involves identifying real-world light sources and measuring or modeling their intensity, color temperature, and light spread angle. Because light models only approximate the behavior of their real-world counterparts, systematically introducing variation to generate slightly different lighting scenarios with each image generation iteration is expected to be beneficial. Therefore, the rendering pipeline should allow for systematic examination of various intensities, color temperatures, and light spread angles.

3.4 Scene Composition

Scene composition refers to the arrangement of cameras, lighting sources, and objects within a virtual environment. It can be defined manually by replicating a known real-world setup, generated procedurally using rule-based algorithms that systematically introduce variations, or created using a combination of both approaches. Procedural methods are particularly advantageous for producing large, diverse data sets, as they automate realistic fluctuations in object placement and camera and lighting viewpoints. Scene composition highly influences a model's performance because it determines which objects are visible in a render, at what size, to what extent illuminated, and from which perspective they are viewed. Consequently, variation in scene composition can be utilized to introduce various defect appearances within a data set.

3.5 Rendering

Based on the composed scene, rendering is the final step in the rendering pipeline, producing a 2D image from a virtual 3D scene. Modern rendering algorithms commonly solve the *rendering equation*, which describes light transport through a scene. Material properties are represented using a Bidirectional Scattering Distribution Function (BSDF), whereas ray-tracing or path-tracing methods simulate how light rays interact with surfaces. A simplified pinhole camera model then captures these interactions from a specified viewpoint.

Shading computes a surface's color or intensity based on lighting, material properties, and viewing direction. Modern pipelines often use physically based rendering (PBR), which employs accurate BSDFs, microfacet distributions, Fresnel effects, and energy conservation to achieve realistic reflections and scattering. Rendering engines such as Blender Cycles, Unreal Engine, or Pixar RenderMan can efficiently produce high-resolution PBR-shaded images on consumer hardware, making PBR the favored choice for realistic synthetic data generation.

In physically based rendering (PBR), a material specifies realistic surface behavior through a few key parameters: base color (albedo), metallicity, roughness, and normal maps. These parameters ensure that surfaces exhibit believable reflectance and scattering under varying lighting and viewpoints. PBR materials remain consistent across various lighting conditions and viewing angles.

Textures refine the material's appearance by mapping values (e.g., color or roughness) across an object's surface. They may be traditional image-based textures applied using UV coordinates, prone to seams or visible tiling, or procedural textures generated algorithmically and can cover surfaces seamlessly.

To what extent realistic reflection behavior contributes to a final model's performance remains an open question. Procedural approaches offer systematic parameter variation, which may be valuable for creating highly diverse defect appearances in a synthetic data set. Therefore, a rendering pipeline should incorporate selecting different object materials or adjusting material properties

within given boundaries to systematically introduce variation in appearance and address the appearance gap by forcing the model to learn material-invariant defect features.

3.6 Output and Annotation

A central advantage of synthetic image generation is producing pixel-accurate annotations for training and evaluating machine learning models. In object detection tasks, for example, standard annotation formats are YOLO, COCO,⁶² or Pascal VOC.⁶³ The essential requirement is to assign each pixel in the render to either the background or one of the defect classes, so that bounding boxes or segmentation labels can be extracted automatically. Annotations must mark what is to be detected in the image; for example, if something is broken off due to a defect, the aim can either be to detect the break-off edge or the missing geometry. Depending on the task, the tool must provide a suitable annotation method.

4 Automated Rendering Pipeline for Defect Image Data Generation

Building on the analysis introduced in Sec. 3, we have implemented a modular, automated rendering pipeline in Blender⁶⁴ that systematically generates annotated defect image data. This pipeline encompasses geometry-based defect generation methods and supports parametric scene composition, material randomization, and automated annotation. In contrast to our prior work introduced in Ref. 58, the current pipeline is capable of automatically generating optimized defect image data sets based on a parametrized CAD model and a configuration file. This is realized with the additional modules for procedural scene composition and parametrized material randomization.

Figure 2 provides an overview of the overall workflow. In a preparatory step, the user models a *task-specific 3D scene*, including all relevant objects (e.g., the inspected part and background items), material models, virtual cameras, and illumination sources. Once this setup is complete, the pipeline automatically generates a series of unique scenes and renders corresponding images, along with pixel-accurate ground-truth annotations indicating a defect's location. Each module is controlled through user-defined configuration files, and a guiding script orchestrates their execution in sequence. Repeating this sequence yields the desired dataset size. The configuration files define how each module in the pipeline behaves, specifying parameters such as camera placement strategies, material properties, and defect characteristics. To systematically investigate how substeps affect the model's performance, the user modifies the configuration files. When working with a fixed random seed, the rest of the pipeline repeats its behaviour, only reflecting the changes in the configuration.

4.1 Scene Composer

The *Scene Composer* module begins with a manually prepared 3D scene. In this scene, users place or model all relevant objects (including the object under inspection, all pertinent background geometry, cameras, and lights) at approximate or canonical locations. This step ensures

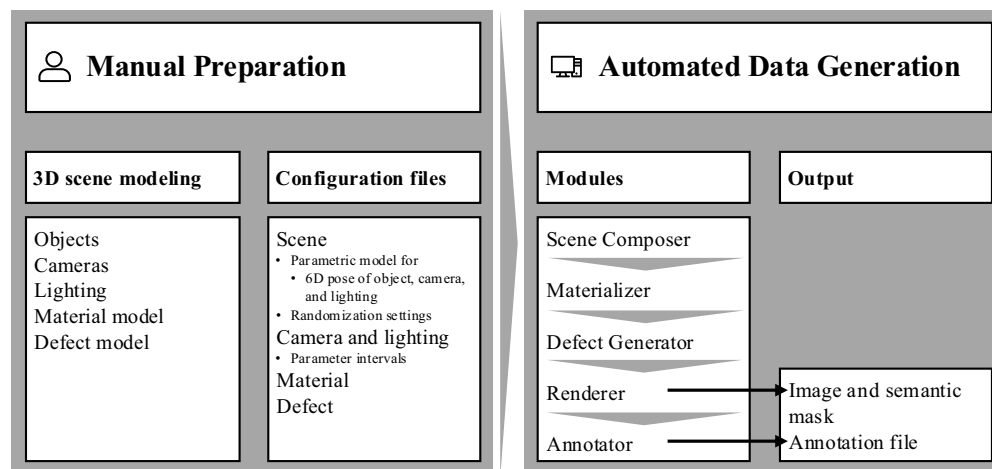


Fig. 2 Schematic overview illustrating the training data generation process.

that the digital environment broadly reflects the real-world setup, including camera resolution, focal length, and lighting configuration.

In the scene configuration file, a parametric model for all scene objects, cameras, and light sources, whose placement in the scene shall be varied, may be chosen to define a set of allowable poses. The Scene Composer module samples a translation pose from the model in each iteration to mimic sufficient coverage of real-world pose variation.

Parametric models implemented are a spherical shell model, where allowable positions are defined by the minimum and maximum limits for the spherical coordinates radius, elevation angle, and azimuth angle, and a cylindrical shell model with minimum and maximum limits for cylindrical coordinates. The module can be easily extended to support additional parametric models. Orientation of the objects can either be centered to a fixed pose or again sampled from a set of poses in the world coordinate system. Alternatively or additionally, for each object, constraints for allowable rotation around its local axes can be set for the module to sample from.

After the Scene Composer applies all specified parametric translation and orientation to the scene objects, further scene objects' parameters may be sampled within user-specified intervals. By default, this includes lighting intensities, lighting colors, camera resolutions, and camera focal lengths that can be randomly sampled within physically plausible ranges. For background variation, environment maps can be sampled with each iteration for a given set of environment maps.

The plausibility of the scene composer configuration should be visually validated in advance. With the tool, users can test and visualize positional and parameter sampling before dataset generation to tune a reasonable configuration.

4.2 Materializer

In the preparation phase, a material model must be defined or developed for each object in the scene. Our Materializer module utilizes user-defined procedural material models, which must incorporate a BSDF model and detailed geometric topology through displacement mapping. To enable variation in the material's appearance, the model must be parametrized at a high level so that users can specify parameter intervals for the module to sample from during each image generation iteration. Examples of these high-level parameters include scaling, translating, and rotating the material mapping on an object and adjusting input parameters for noise functions (e.g., Perlin noise) used to modulate color, roughness, and other material properties.

Once the scene layout is finalized, the Materializer module assigns and, if desired, samples parameters for the material models of each object in the scene. This approach facilitates systematic variation in surface finishes, reflectivities, and colors across images.

4.3 Defect Generator

For geometry-based defects, the Defect Generator begins by selecting an appropriate location on the surface of the inspected part. The user specifies a set of candidate areas, each representing a plausible defect location. Algorithm 1 details how a point is uniformly chosen from these candidate areas while ensuring it remains within the camera's field of view and is not occluded by other geometry. Similar to the orientation constraints in the Scene Composer module, the defect object's orientation relative to the surface normal at the selected location may be restricted by enforcing user-defined rotation limits along the normal axis. Figures 3(a)–3(c) illustrate defect positions on an object as red spheres, corresponding to the predefined candidate areas.

After selecting the location, the pipeline places a *defect object* at that point. During the preparation phase, the user must model the defect object, which can be manually created, procedurally generated, or derived from physical simulations or real-world measurements. To introduce variation along the break-off edge, the defect object may be subjected to user-defined displacement mapping that adds fine details.

Next, the module samples scaling factors and translation distances along the surface normal from user-defined randomization intervals [see Figs. 3(d)–3(i)]. Then, a Boolean operation merges the defect object with the main part mesh, simulating a protrusion, a shallow cavity, or significant geometry removal [see Figs. 4(a)–4(c)]. Following the Boolean operation, the module automatically creates an annotation object corresponding to the defect. For protrusions, this annotation object is the added geometry itself [see Fig. 4(d)]. For shallow cavities or moderate geometry removal, a thin-walled shell object is embedded to delineate the defect region [see Fig. 4(e)].

Algorithm 1 Uniform Point Selection on Surfaces.

Require: S : set of surfaces**Require:** δ : density criterion**Ensure:** P : list of selected points

```

1:  $P \leftarrow \emptyset$  ▷ Initialize an empty list of points
2: while  $P$  does not satisfy the point density criterion  $\delta$  do
3:    $F \leftarrow$  select a surface from  $S$  (based on weighted probability)
4:    $p \leftarrow$  sample a point uniformly on  $F$ 
5:    $P \leftarrow P \cup \{p\}$ 
6: end while
7: for all  $p \in P$  do
8:   Check the surface normal orientation at  $p$ 
9:   if normal not valid at  $p$  then
10:    continue
11:   end if
12:   Check if  $p$  lies within the camera's field of view
13:   if  $p$  not in FOV then
14:    continue
15:   end if
16:   Check for occlusion of  $p$  via raycasting
17:   if  $p$  is occluded then
18:    continue
19:   end if
20:   break ▷ All checks passed
21: end for
22: return  $P$ 

```

For significant geometry removal, the removed portion of the mesh is preserved as an annotation object [see Fig. 4(f)]. In every case, the annotation object remains hidden in the final image but is used to generate an accurate, pixel-wise semantic mask indicating the defect's location.

In summary, the module enables systematic investigation of how defect object shape, size, penetration depth, position, orientation, and topology changes affect the outcome.

4.4 Renderer and Annotator

Finally, the *Renderer and Annotator* modules render the parametric scene with Blender's *Cycles* engine. During this process, based on the annotation object, a semantic mask is rendered (Fig. 4 bottom row). This mask is saved, encoding each pixel whether it shows the background or a defect class. Optionally, the pipeline converts the semantic mask into a user-specified annotation format, such as COCO or YOLO.

5 Case Study Aircraft Engine Inspection

Beyond the substantial impact of manual 3D scene modeling, the choice of parameters and randomization intervals in the configuration file of the data generator critically influences the

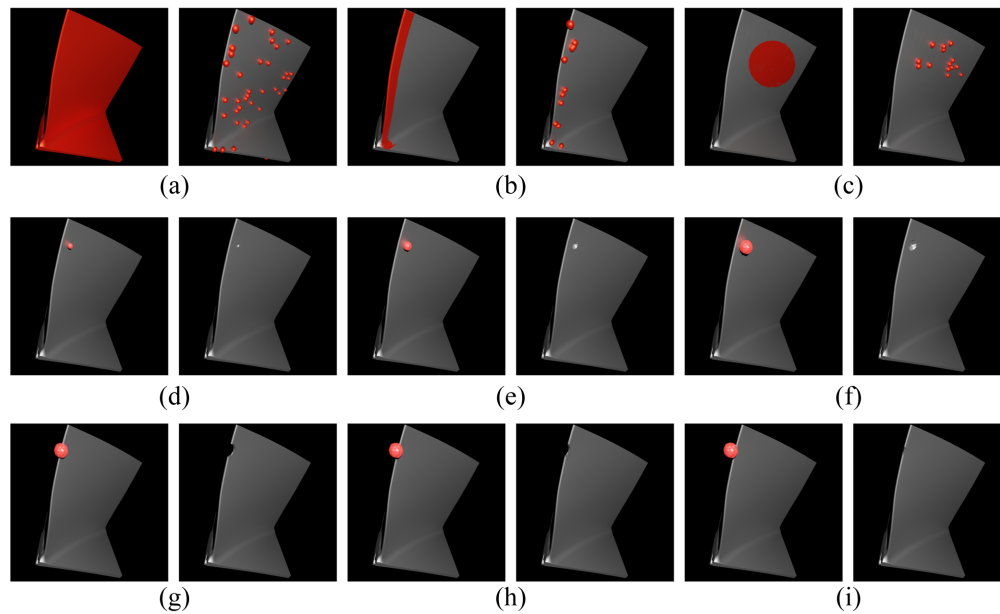


Fig. 3 Example renderings of a turbine blade designated for inspection, illustrating user-defined defect object positioning areas (top row), the effects of varying defect object sizes, and different penetration depths for a constant-size defect object displaced along the surface normal. Defect objects are shown as red spheres, whereas candidate surface areas are highlighted in red. (a) Entire object. (b) Object edge. (c) Custom-defined area. (d) Small defect size. (e) Moderate defect size. (f) Large defect size. (g) Large penetration depth. (h) Moderate penetration depth. (i) Small penetration depth.

appearance and variability of the synthetic image data. Because both physically based rendering and subsequent training and evaluation of an object detection model are computationally expensive, it is necessary to devise a strategy that yields a sufficiently performant dataset for a given application at reasonable computational cost. To this end, we propose a two-stage procedure. After manual construction of the 3D scene based on an analysis of the inspection use case, we first conduct a prescreening experiment to identify the most influential parameters and thereby reduce the number of factors to be investigated. Subsequently, the most influential parameters are tuned using a naive OFAT optimization. For each parameter, a set of candidate support points is evaluated, the best-performing value is fixed, and the procedure is repeated for the next parameter.

As a performance metric for the object detection task, we use the mean average precision (mAP) as implemented in the Ultralytics YOLO (<https://github.com/ultralytics/ultralytics>) framework. For a given intersection over union (IoU) threshold, precision and recall are computed from the numbers of true positives, false positives, and false negatives. A predicted bounding box is considered a true positive if its IoU with an as yet unmatched ground-truth bounding box of the same class is greater than or equal to the IoU threshold; otherwise, it is counted as a false positive, whereas remaining ground-truth boxes are counted as false negatives. Varying the detection confidence threshold yields a precision–recall curve, and the average precision (AP) is obtained by integrating the envelope of this curve using the 101-point interpolation procedure adopted in the COCO evaluation toolkit.⁶² We report mAP@0.5IoU, which averages AP over all classes (VOC-style metric). However, because we use a single defect class in this work, mAP is equal to AP.

In the following, we instantiate the proposed data generation pipeline for a representative use case from aircraft engine inspection. Aircraft engines are routinely subjected to borescope inspections to enable early detection of faults in safety-critical components. During such inspections, a borescope is manually inserted into dedicated inspection ports. Once the borescope is in place, the engine shaft is rotated so that the blades sweep past the borescope's field of view, allowing a stage-by-stage inspection of the engine. Human inspectors then visually detect and classify any defects observed in the recorded images.

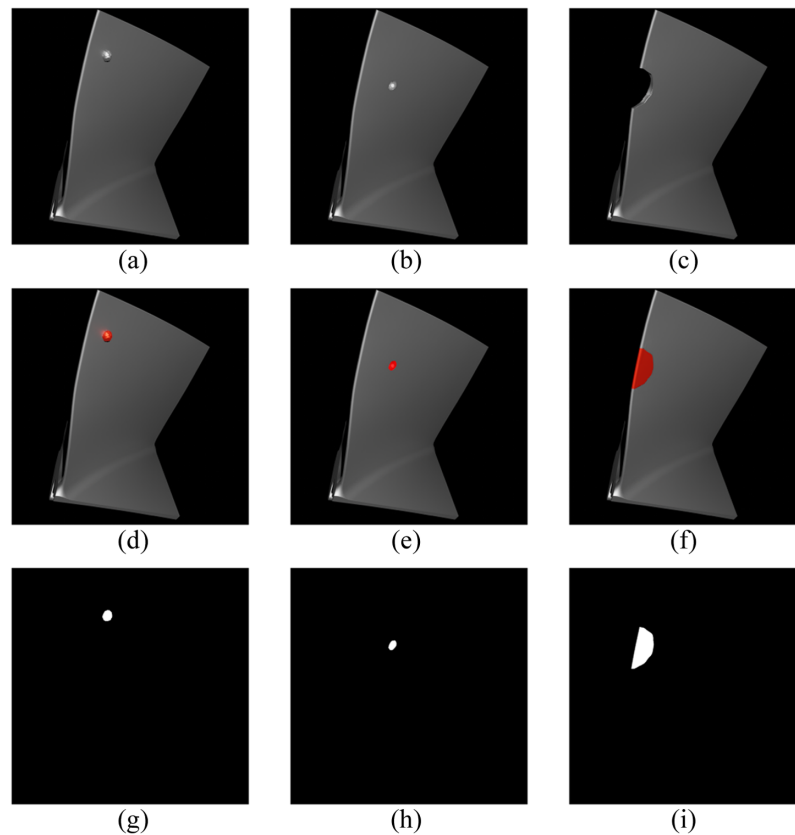


Fig. 4 Demonstration of different automatic annotation strategies. (Left) Added geometry is treated as a separate object. (Middle) A thin-walled shell object is positioned within the cavity following minor geometry removal. (Right) Removed geometry is employed as a separate object but is only rendered to generate the semantic mask (bottom row). (a) Protrusion. (b) Minor geometry removal. (c) Significant geometry removal. (d) Added geometry. (e) Shell object. (f) Removed geometry. (g)–(i) Annotation mask.

Automating defect detection and classification in this setting is challenging for several reasons. Because the borescope is manually guided, the viewpoint varies slightly between inspections, resulting in changes in perspective. The borescope is operated in proximity to the inspected surfaces, such that small translations or rotations lead to substantial differences in the captured images. Illumination provided by the integrated borescope light often causes glare and nonuniform lighting. The miniaturized sensor and optics introduce limited spatial resolution and additional imaging artefacts. Shadows cast by scene geometry further complicate image interpretation, and variations in surface finish and material appearance add another layer of complexity (see Fig. 5).

These factors have historically rendered conventional, rule-based image processing unsuitable for this application. Although deep learning-based image analysis is well-suited to handle the large intraclass and interclass variability encountered in borescope inspections, the availability of annotated training data is limited, particularly for specific defect classes and their diverse manifestations.⁶⁶ This scarcity of labeled examples hampers the development of comprehensive and robust defect detection models.

One defect class of particular relevance is *missing material*, where a blade has fractured and a macroscopic section of material is absent. For this defect class, Ref. 65 provides a dataset of 111 borescope images from the high-pressure compressor (HPC) section of an engine that were pre-annotated by domain experts (see Fig. 5). The images contain examples of missing material defects with corresponding ground-truth bounding boxes. We partially refined these bounding boxes to more accurately delineate the visible defect regions and use the resulting dataset as a real-image validation set.

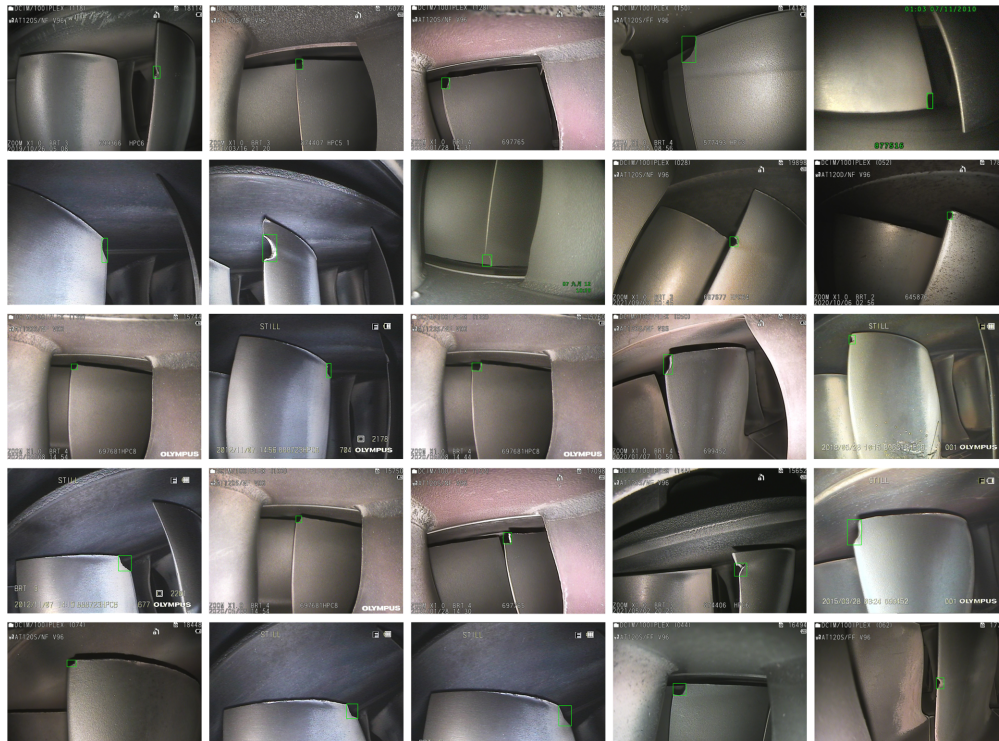


Fig. 5 Samples from the real-world dataset⁶⁵ of borescope inspection of aircraft engines. The illustrated defect category, *missing material*, depicts a blade section that has been removed. Green boxes indicate the ground truth bounding box annotations.

5.1 Synthetic Data Generation—Manual Preparation

In the first step of the synthetic data generation procedure introduced in Sec. 4, we manually construct the 3D inspection scene. CAD models of a CFM56 engine, including shafts, blades, vanes, and the housing, were provided by a local aircraft maintenance organization based on high-resolution 3D scans. These components were arranged such that all parts visible from the borescope ports of the high-pressure compressor (HPC) section are within the field of view, as illustrated in Fig. 6(a). For scene composition, the only movable component apart from the camera and illumination is the engine shaft, which is allowed to rotate about its central axis to emulate the real inspection procedure in which blades pass through the borescope's field of view. The camera translation is sampled in cylindrical coordinates to reflect the actual movement of the borescope within the port. To constrain the camera orientation, we define fixed intervals for rotation about three axes: rotation about the borescope axis and roll and pitch about the camera axis. The six-degree-of-freedom pose of the light source is fixed relative to the camera to mimic the configuration of real borescopes, and illumination is modeled using an area light with constant intensity and color.

We then introduce the defect geometry. Because this defect class is characterized by substantial material loss, we model it as a geometry-based defect and reuse the removed material as an explicit annotation object [see Fig. 4(f)]. The break-off edge is approximated using two simple geometric primitives, a sphere and a cube [Figs. 6(b) and 6(c)].

5.2 Prescreening

To efficiently identify the most influential parameters of the rendering pipeline on the performance of the industrial defect detection model, we employ a two-level Plackett–Burman screening design.⁴ PB designs are fractional factorial designs tailored for high-dimensional factor spaces: they estimate the main effects of many factors with a minimal number of runs using an orthogonal (or near-orthogonal) layout that balances factor levels across experiments. This makes them particularly suitable for our computationally expensive synthetic data generation and model training pipeline, in which the goal is to rapidly narrow down the parameter set.

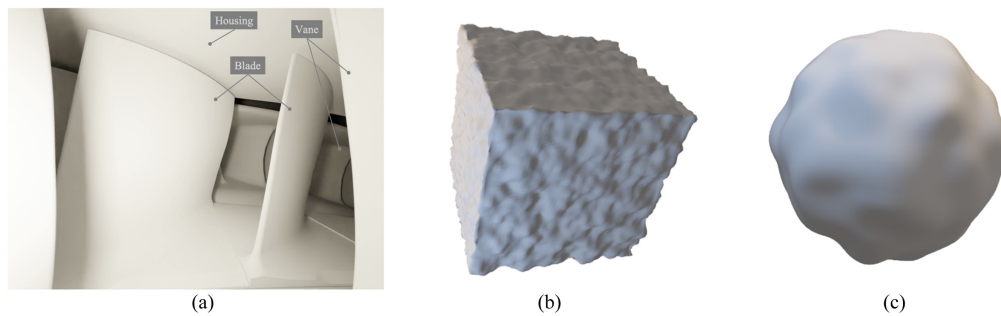


Fig. 6 (a) Virtual scene as viewed through a borescope port. (b) Cube-shaped defect object. (c) Sphere-shaped defect object.

For the present use case, we first identified a set of potentially influential rendering parameters and specified a high (+) and low (–) level for each, as listed in Table 1. Following the geometry-based defect generation scheme introduced in Sec. 4.3, we considered the following defect-related factors: the positioning area of the defect object (1), the object shape (2), the object size (3), the size randomization interval (4), the penetration depth randomization interval (5), and the displacement-mapping strength (6). As an example, the high level of the object shape corresponds to a pure sphere, whereas the low level corresponds to a randomly chosen sphere or cube. In an analogous fashion, we defined candidate levels for light, camera, optics, scene composition, and material parameters.

For the material factor, we restrict the study to procedural textures for two reasons. First, procedural textures retain a high effective resolution even when the virtual camera is positioned very close to the surface, as in the borescope inspection scenario considered here. Second, they expose a compact set of high-level parameters that can be systematically randomized. We manually designed two distinct procedural materials and used them as the low and high levels of this factor. The *plain gray* material is a homogeneous gray, matte texture without additional surface structure, see Fig. 8(b). The *random procedural* material, in contrast, combines several Perlin-noise fields to modulate the base color, roughness, metallicity, normal map, and displacement of the part and superimposes an additional procedural layer that mimics corrosion or surface debris.

Table 1 High (+) and low (–) factor settings used in the two-level Plackett-Burman screening design.

#	Factor	High level (+)	Low level (–)
Defect			
1	Positioning area	Leading edge	Corner
2	Defect object	Sphere	Sphere, cube
3	Defect object size	0.0015 m	0.004 m
4	Defect object size rand. interval	± 0.004 m	± 0 m
5	Penetration depth rand. interval	0 to 0 m	0 to 0.0001 m
6	Displacement mapping strength	0–0	0.05–0.15
Light			
7	Spot angle	150°	100°
8	Spot angle rand. interval	$\pm 0^\circ$	$\pm 30^\circ$
9	Light distance rand. interval	± 0 m	± 0.01 m

Table 1 (Continued).

#	Factor	High level (+)	Low level (–)
Defect			
10	Light angle; rand. interval	0°; $\pm 0^\circ$	0°; $\pm 10^\circ$
11	Light intensity	0.005	0.0
12	Intensity rand. interval	± 0	± 0.004
13	Color rand. interval	$H \in [0.1, 0.1]$; $S \in [0.35, 0.35]$; $V \in [1, 1]$	$H \in [0, 1]$; $S \in [0, 0.5]$; $V \in [1, 1]$
Camera			
14	Resolution	$x = 1024$ px; $y = 768$ px	$x = 512$ px; $y = 384$ px
15	Resolution rand. interval	$x = \pm 0$ px; $y = \pm 0$ px	$x = \pm 50$ px; $y = \pm 50$ px
Optics			
16	FOV	110°	85°
17	FOV rand. interval	$\pm 0^\circ$	$\pm 20^\circ$
Scene composition			
18	Spool angle rand. interval	0 rad	0.02 rad
19	Camera translation rand. intervals	$d \in [0, 0]$ m; $r \in [0, 0]$ m; $\phi \in [0, 0]$ rad	$d \in [-0.005, 0.000]$ m; $r \in [0, 0.001]$ m; $\phi \in [0, 6.28]$ rad
20	Camera orientation	$\phi \in [-0.0, 0.0]$ rad; $\theta \in [-0.0, 0.0]$ rad; $\psi \in [-0.0, 0.0]$ rad	$\phi \in [(-0.1) - 0.1]$ rad; $\theta \in [(-0.1) - 0.1]$ rad; $\psi \in [(-0.2) - 0.2]$ rad
21	Scene	Port9-LE	Port9-LE, Port7-LE, Port6-LE, Port4-LE, Port3-LE
Material			
22	Proc. Material	plain grey	random

This material is configurable via high-level parameters controlling the base colors, Perlin-noise settings, amount and type of dirt, and the strengths of the normal and displacement maps. For the random material, we specified broad randomization intervals for all of these parameters to obtain the pronounced appearance variation visible in Fig. 8(a). These intervals are defined in the material configuration file and are sampled by the Materializer module to generate a unique material instance for each rendering iteration.

With $k = 22$ candidate factor, the smallest admissible PB design requires 24 experimental runs, which allows us to assign the 22 factors plus one dummy column. The estimated effect of the dummy was used to approximate experimental noise. In each run, all factors are fixed to their predefined high (+) or low (–) level according to the PB design matrix, see Table 5. The run order is fully randomized to mitigate temporal or hardware-related drift, and all other random seeds in the rendering pipeline are kept constant. For every design point, we instantiate the corresponding configuration of the rendering pipeline and generate a synthetic training dataset of 500 images,

which is then used to fine-tune the object detector. The response variable is the mAP on the fixed real-image validation set introduced in Sec. 5, Fig. 5.

We employ a lightweight object detection model to keep the training time per synthetic dataset low. Specifically, we use the Real-Time Detection Transformer (RT-DETR)⁶⁷ as a compromise between model capacity and training duration, with the hyperparameter configuration given in Table 2. Online data augmentation is applied during training and includes HSV jitter (hue ± 0.1 , saturation ± 0.7 , value ± 0.4), random translation (up to 20%), isotropic scaling (factor [0.5, 1.5]), mild perspective distortion (0.001), random horizontal and vertical flips ($p = 0.5$ each), and compositional Mosaic ($p = 1.0$), MixUp ($p = 0.2$), and Copy-Paste ($p = 0.1$) augmentations, with rotation, shear, CutMix, and channel-order perturbations disabled.

Table 5 summarizes the experimental design and the resulting mAP values. Based on these results, we estimate the main effects via ordinary least squares on the coded (± 1) design matrix and assess their significance using Lenth's pseudostandard error (PSE) method, with the dummy factor serving as an internal noise benchmark. For each factor A, the unreplicated main effect is computed as the difference between the mean response at the high and low level, $e(A) = \bar{y}_{(A=+1)} - \bar{y}_{(A=-1)}$; larger absolute effect values indicate stronger influence on the detection performance. Lenth's method is then used to derive a simultaneous margin of error (SME) from the distribution of effects, and factors with $|e(A)| > \text{SME}$ are classified as active.

$$s_0 = 1.5 \text{ median}(|e_i|)$$

$$I = \{i \in \{1, \dots, p\} \mid |e_i| \leq 2.5s_0\}, \quad m = |I|$$

$$\text{PSE} = 1.5 \text{ median}_{i \in I}(|e_i|), \quad \text{SME} = k_2(m, \alpha) \cdot \text{PSE}$$

Among the 22 factors (plus one dummy), four exceed the SME and are therefore identified as strongly active: the material model, the positioning area of the defect object, the defect object size, and the penetration depth randomization interval. All other factors fall below the SME threshold, and the dummy factor is nonsignificant, supporting the validity of the error estimate. The ranking and relative magnitude of the effects are visualized using a Pareto plot in Fig. 7.

5.3 Optimization

Having identified the most influential factors in the screening experiment, we proceed to the optimization stage. The optimization goal is to find the optimal input parameter for the most influential factors to maximize the mAP@0.5IoU. As a starting point, we configure the training data generator by setting each factor to the level (high or low) that yielded better performance in the Plackett–Burman screening. We then carry out a simple OFAT optimization of the strongest main effects, varying one active factor at a time while keeping all others fixed at their current best level. In this step, we deliberately neglect potential second-order interaction effects between factors, because the number of available runs is limited and our primary objective is to obtain a

Table 2 Hyperparameter used for prescreening and optimization with the RT-DETR model.

Hyperparameter	Value
Optimizer	AdamW ⁶⁸
Initial LR	5×10^{-4}
LR-Schedule	Cosine decay, final LR = $0.1 * \text{initialLR}$
Batch size	16
Epochs	300
Patience	15
Training dataset samples	500
Validation dataset samples	111

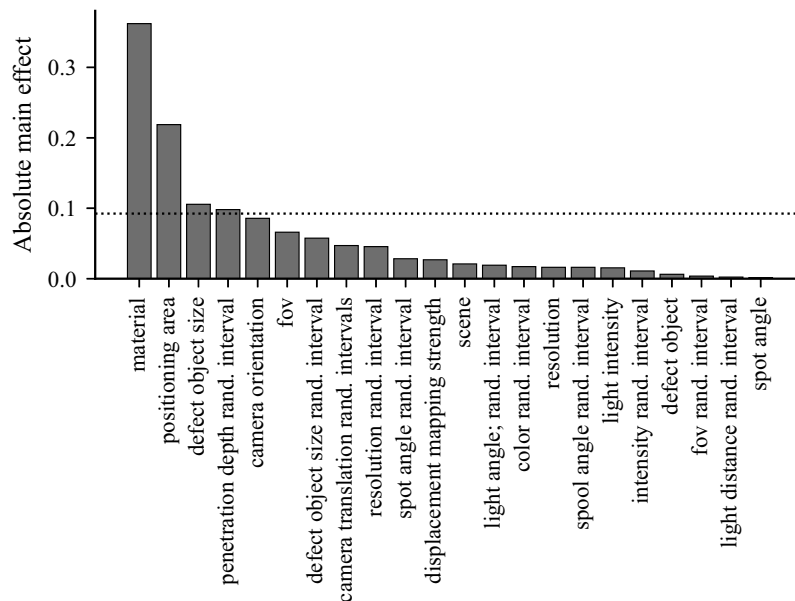


Fig. 7 Pareto chart of absolute main effects for the screening factors. Dotted line indicates the simultaneous margin of error (SME).

practically improved configuration rather than a fully characterized response surface. The best-performing dataset configuration obtained from this procedure is used as the baseline for the subsequent optimization stage. For each configuration, we train and evaluate the same object detection model as described in the previous section.

Because the material model was identified as the most active factor, we first refined this aspect in the optimization. In addition to the two procedural materials used in the prescreening (plain gray and fully randomized), we introduced two more realistic material models: a procedural metal and a procedurally randomized metal, see Figs. 8(c) and 8(d). Both models are implemented within the same procedural framework and expose high-level parameters that allow us to generate controlled appearance variation.

Next, we investigated the spatial distribution of defect objects on the blade surface, see Figs. 3(a)–3(c) for an illustration. We considered four placement options: restricting defects to the *leading edge*, restricting them to the *tip corner*, allowing placement on both the leading edge and the tip corner, and allowing placement on the entire object. We then optimized the defect object size. Because the absolute defect size is coupled to the randomization interval of the object size, we treated both quantities jointly and combined four levels of nominal defect size with three levels of size-randomization interval, see Table 3. Finally, we evaluated three different randomization intervals for the defect penetration depth.

As a last step, we revisited a factor that had not been identified as active in the prescreening and was set to zero in the baseline configuration for the optimization: displacement mapping. Because displacement mapping substantially increased the perceived realism of the synthetic defect images, we included it as an additional factor in the optimization.

Table 3 summarizes the results of the optimization. Among the material models, the fully randomized procedural material yields the highest detection performance. Restricting defect placement to both the leading edge and the tip corner of the blade consistently outperforms placing defects only on a single region or across the entire surface. Larger nominal defect sizes combined with a 25% size-randomization interval outperform smaller defects. Furthermore, extending the randomization interval for the penetration depth offers additional advantages. As expected from the qualitative assessment of image realism, enabling displacement mapping further improves the effectiveness of the synthetic dataset for training. Representative samples from the best-performing configuration are shown in Fig. 9.

To determine the appropriate size of the synthetic training dataset, we switch to the Faster R-CNN detector⁶⁹ with a ResNet-50⁷⁰ backbone pretrained on ImageNet.⁷¹ This aligns with a widely used benchmark architecture. Furthermore, the larger model promises to leverage the



Fig. 8 Samples of synthetic defects with different material models used in the optimization study. (a) Fully randomized. (b) Plain gray. (c) Metal. (d) Randomized metal.

Table 3 Results of optimizing the most active pipeline parameter, one factor at a time, are presented. The best-performing dataset (bold) serves as the baseline for the next parameter optimization.

Parameter	Value	mAP0.5IoU
Material	Fully randomized	0.611
Material	Plain gray	0.259
Material	Metal	0.422
Material	Randomized metal	0.526
Defect object positioning area	Entire object	0.401
Defect object positioning area	Tip corner	0.539
Defect object positioning area	Leading edge	0.611
Defect object positioning area	Leading edge and tip corner	0.655
Defect object size [m]; rand. interval	$s = 0.001; \Delta = s * 0.0$	0.076
Defect object size [m]; rand. interval	$s = 0.001; \Delta = s * 0.25$	0.099
Defect object size [m]; rand. interval	$s = 0.001; \Delta = s * 0.5$	0.174
Defect object size [m]; rand. interval	$s = 0.0015; \Delta = s * 0.0$	0.563
Defect object size [m]; rand. interval	$s = 0.0015; \Delta = s * 0.25$	0.474
Defect object size [m]; rand. interval	$s = 0.0015; \Delta = s * 0.5$	0.593
Defect object size [m]; rand. interval	$s = 0.0035; \Delta = s * 0.0$	0.682
Defect object size [m]; rand. interval	$s = 0.0035; \Delta = s * 0.25$	0.671
Defect object size [m]; rand. interval	$s = 0.0035; \Delta = s * 0.5$	0.682
Defect object size [m]; rand. interval	$s = 0.0045; \Delta = s * 0.0$	0.639
Defect object size [m]; rand. interval	$s = 0.0045; \Delta = s * 0.25$	0.695

Table 3 (Continued).

Parameter	Value	mAP0.5IoU
Defect object size [m]; rand. interval	$s = 0.0045$; $\Delta = s * 0.5$	0.616
Penetration depth rand. interval [m]	0–0.0001	0.695
Penetration depth rand. interval [m]	0–0.0002	0.654
Penetration depth rand. interval [m]	0–0.0005	0.678
Penetration depth rand. interval [m]	0–0.001	0.712
Penetration depth rand. interval [m]	0–0.002	0.612
Displacement mapping strength	0.05–0.15	0.799

theoretically unlimited amount of available training data and potentially yield better detection performance.

During training, we applied online data augmentation using Albumentations (<https://github.com/albumentations-team/albumentations>), including D4 dihedral transforms (all 90 deg rotations and flips, $p = 1.0$), bounding-box–safe random cropping ($p = 0.5$), random blur (motion/Gaussian/median, $p = 0.3$), additive image noise (Gaussian or ISO noise, $p = 0.4$), photometric jitter (RandomBrightnessContrast, RandomGamma, or CLAHE, $p = 0.9$), HSV jitter (hue ± 5 , saturation/value ± 10 , $p = 0.4$), lossy JPEG recompression (quality 40–100, $p = 0.5$), and mild perspective distortion (scale 0.02–0.05, $p = 0.3$). We trained with a batch size of 8 and a fixed learning rate of 0.00882, using stochastic gradient descent⁷² with momentum 0.9 and weight decay 0.0001. The resulting performance as a function of training set size is reported in Fig. 10. Starting from ~ 500 training images, the validation average precision reaches a clear plateau. The best-performing model in this study achieves an $mAP@0.5$ IoU of 0.89 on the validation set.

The reported real-data results are intended to assess the practical utility of the optimized synthetic training data generated with the proposed pipeline. They should not be interpreted as an attempt to establish a new state of the art on the benchmark, because detector architecture,



Fig. 9 Representative samples from the best-performing synthetically generated dataset using the developed rendering pipeline. These examples illustrate variations in viewpoint, defect position and shape, and material appearance.

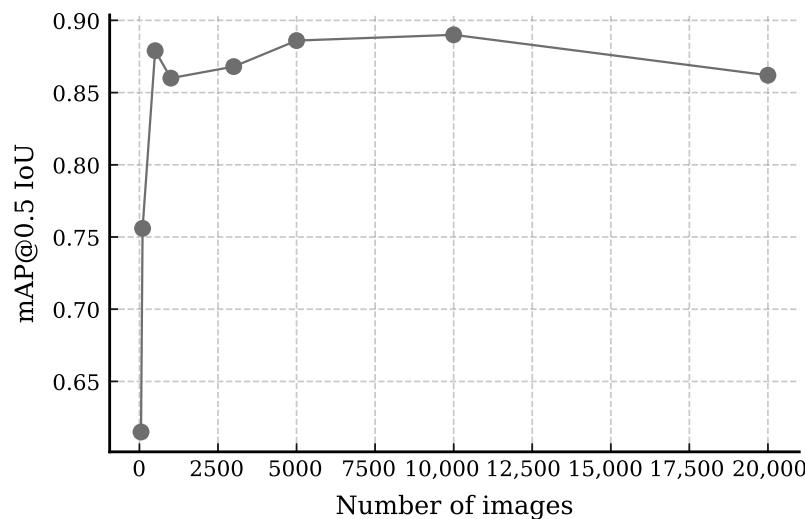


Fig. 10 Comparison of dataset size and mean average precision of the trained model.

training protocol, and the use of real versus synthetic training data differ substantially across studies. Although the preceding experiments identify a high-performing synthetic dataset configuration with respect to the available real-world benchmark, this benchmark also guides the optimization process. We therefore perform an additional independent evaluation to examine whether the resulting gains transfer to previously unseen data.

5.4 Independent Generalization Test

To assess whether the proposed optimization procedure improves transfer beyond the real-image validation set used during screening and OFAT tuning, we conducted an additional evaluation on a third, independent dataset. This experiment aims to reduce the risk that the synthetic data generation pipeline was inadvertently over-specialized to the specific real-world benchmark employed during parameter selection.

Because no alternative dataset of this defect class was available to the authors, an additional benchmark was physically reproduced. For this purpose, a high-pressure compressor (HPC) spool and real compressor blades were used. Missing material defects were manually introduced on 20 blades using files of different geometries, yielding round, oval, and angular material-removal patterns intended to approximate the defect morphology observed in the real-world borescope images shown in Fig. 5.

Subsequently, a dataset of 120 images was recorded with an endoscope from varying view-points, and all visible defects were manually annotated with bounding boxes by the authors (Fig. 11). Although this reproduced dataset does not constitute a replica of in-service borescope imagery, it provides an additional out-of-sample evaluation set with independently generated defect instances and slightly different acquisition characteristics.

For the generalization assessment, two Faster R-CNN detectors were trained using the same training protocol as in the preceding dataset-size study. The first model was trained on dataset 1 from the initial prescreening experiment (Table 5), representing an early, nonoptimized synthetic dataset configuration. The second model was trained on the best-performing optimized synthetic dataset comprising 10,000 images (Fig. 10). Both models were then evaluated on the original real-world validation set and on the reproduced dataset. The results are summarized in Table 4.

The optimized synthetic dataset improves performance not only on the original real-world benchmark, where the mAP@0.5IoU increases from 0.55 to 0.89, but also on the independently reproduced dataset, where it rises from 0.30 to 0.76. This consistent improvement across both evaluation sets suggests that the proposed screening-and-optimization procedure does not merely adapt the synthetic pipeline to the idiosyncrasies of the validation set used during development. Instead, it improves the ability of the generated training data to support detectors that transfer to previously unseen images acquired under related, but not identical, conditions.

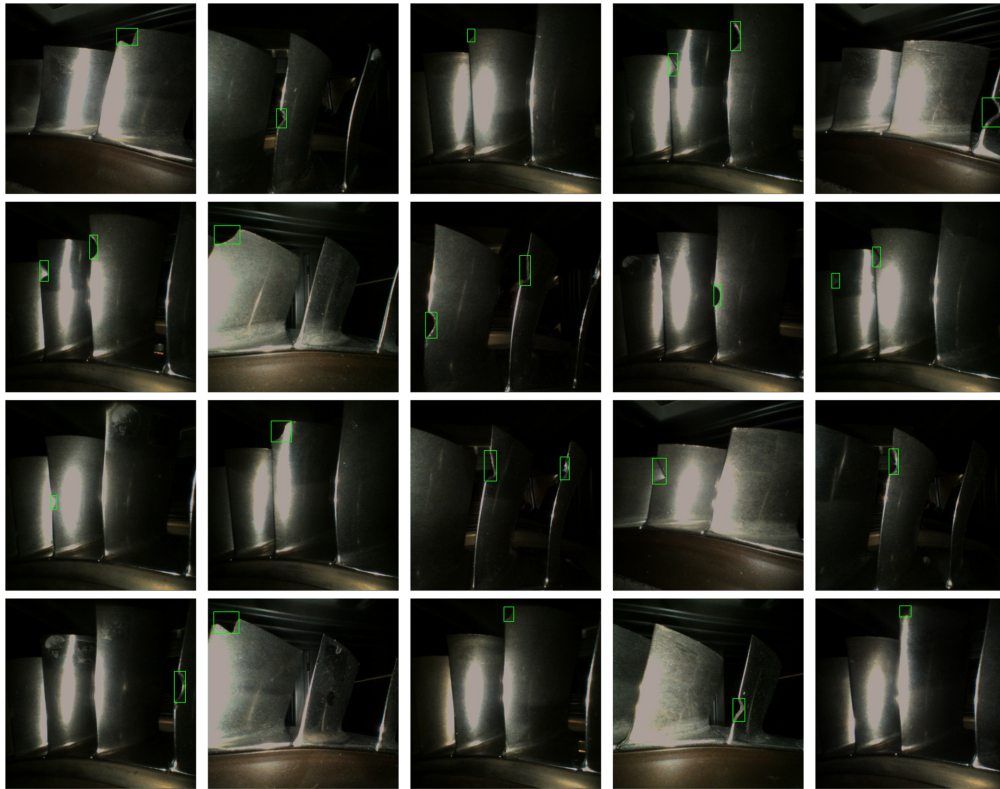


Fig. 11 Representative samples from the independently reproduced endoscopic dataset used for the generalization assessment. Missing material defects were manually introduced on real HPC blades by controlled filing and imaged from varying viewpoints.

At the same time, the reproduced dataset should be interpreted as an additional stress test of transferability rather than as a fully unbiased estimate of deployment performance. The defects were manually created to resemble the target defect class, and the acquisition setup differs from the original borescope benchmark. Nevertheless, the observed performance gain on this independent dataset provides further evidence that the proposed synthetic data optimization strategy improves generalization in a practically relevant manner.

6 Discussion

Comparing Figs. 9 and 5 qualitatively indicates that the proposed pipeline can reproduce key sources of variability observed in real inspection data, including changes in camera viewpoint and perspective, object pose, and defect shape, size, and location. The synthetic images also reflect a range of material appearances and nuisance factors that are characteristic of the borescope inspection setting.

In the quantitative study in Sec. 5, we demonstrated how the pipeline can be used to systematically improve the suitability of a synthetic dataset by first identifying influential factors and

Table 4 Comparison of detector performance on the original real-world benchmark and the independently reproduced dataset.

Train dataset	val dataset	
	Real-world dataset [mAP@0.5IoU]	Independent generalization dataset [mAP@0.5IoU]
Prescreening dataset 1	0.55	0.3
Optimized dataset	0.89	0.76

then refining them in a targeted optimization. Using only synthetic images generated with the best-performing configuration, the resulting detector reaches an mAP@0.5IoU of 0.89 on the available real-world validation set. This suggests that, for the considered use case, purely synthetic data can provide practically useful supervision for training an object detector. At the same time, the small size of the real-world benchmark limits the strength of the conclusions that can be drawn regarding expected performance in an operational inspection process. More generally, the requirement for sufficiently large real-world datasets for evaluation remains a bottleneck when synthetic data are deployed for object detection tasks: even when synthetic data reduce the need for real training images, substantial real data are still required to validate the resulting models.

The interpretability of the main optimization results is further constrained by the fact that the original real-world validation set influenced both the tuning of the synthetic data generation pipeline and the development of the detector training procedure. However, the additional reproduced dataset introduced in Sec. 5.4 provides a first independent check against over-specialization to that benchmark. The consistent performance gain observed on both evaluation sets supports the conclusion that the optimized synthetic pipeline captures transferable variation rather than merely matching the peculiarities of the original validation data. Nevertheless, the reproduced dataset is not a fully independent field test: it was manually constructed to resemble the same defect class, differs from the original benchmark in acquisition characteristics, and remains limited in size. Consequently, the reported scores should be interpreted as encouraging evidence of transferability rather than as definitive estimates of deployment performance across all engines, sensors, and inspection conditions.

The present study is also not intended to establish state-of-the-art benchmark performance through detector-specific tuning or broad comparison across model families. Instead, our objective is to examine how effectively a controllable rendering pipeline can be configured such that synthetic data alone becomes useful for training detectors that transfer to real inspection images. For the same reason, direct comparison with literature results on the same benchmark is of limited interpretability when detector architectures, training protocols, data splits, and the use of real versus synthetic training data differ substantially. Accordingly, the reported results should be understood primarily as evidence for the practical utility of optimized synthetic training data, rather than as an attempt to claim superior benchmark performance relative to prior work.

The prescreening and optimization stages introduce further methodological limitations. The Plackett–Burman design used in the prescreening covers only a subset of all potentially relevant parameters and evaluates them at two discrete levels. As such, it primarily captures main effects and cannot reliably detect higher-order interactions or nonlinear effects across the parameter space. This limitation is exemplified by the displacement mapping strength: it was not identified as strongly active in the prescreening, yet enabling displacement mapping in the later optimization had a noticeable impact on detector performance. Likewise, the subsequent OFAT refinement deliberately neglects interaction effects between factors. In the present work, this was accepted as a pragmatic compromise, because the aim was not to identify a globally optimal response surface but to obtain a sufficiently effective synthetic dataset under realistic computational and engineering constraints. In general, however, the aliasing and confounding inherent to two-level screening designs, together with the restricted scope of OFAT refinement, can lead to both false positives and false negatives in the set of active factors.

A related limitation is that the study remains primarily empirical. Although the experiments reveal which randomization parameters are most influential for the considered task, they do not yet provide a deeper theoretical explanation of why these factors dominate the sim-to-real transfer. For example, the current results do not disentangle whether certain parameters are beneficial because they improve geometric realism, enlarge appearance coverage, suppress shortcut learning, or better align the synthetic image statistics with those of the real benchmark. Accordingly, the findings should be interpreted as empirical evidence on the practical utility of structured domain-randomization studies for rendering-based synthetic data generation, rather than as a complete theoretical account of transfer mechanisms in synthetic-to-real defect detection.

From a practical perspective, the current strategy of generating a new synthetic dataset for every configuration is computationally and operationally demanding. An alternative design would be to first generate a large pool of synthetic images with systematically varied and explicitly recorded generative parameters (e.g., material model, defect size, position, and penetration depth) and then construct datasets by sampling from this pool according to the desired experimental design. Such a decoupling of image generation and dataset assembly could substantially reduce the computational cost of exploring additional configurations, introducing repeated runs, or carrying out more elaborate statistical analyses.

Finally, several design choices in our pipeline are guided by engineering intuition rather than formal optimization. The manual 3D scene setup, the selection of factors and their high/low levels in the prescreening, and the ranges chosen for the subsequent optimization are not unique; other practitioners might arrive at different settings. In particular, the material model, which was identified as one of the most influential factors, can be realized in many different ways and at varying levels of modeling effort. We argue, however, that pursuing the simplest possible models and iteratively refining only those factors that demonstrably improve task performance is a reasonable strategy to limit manual modeling effort. The principle of domain randomization is beneficial in this regard: the synthetic images do not need to be perfectly photorealistic, but must provide sufficient and task-relevant variation. As our case study suggests, variation-rich, and in part even unrealistic, synthetic datasets can be generated with comparatively simple methods, without resorting to labor-intensive procedures such as measuring the physical reflectance properties of real components.

7 Conclusion and Outlook

In this work, we presented a configurable rendering pipeline for generating synthetic training data for visual defect detection. The pipeline exposes key stages of scene setup, defect modeling, material definition, and annotation, enabling systematic variation of these factors. In combination with a structured experimental design, this allowed us to quantify how specific rendering choices affect the performance of detectors trained solely on synthetic images. For the investigated missing-material defect scenario, detectors trained on appropriately configured synthetic datasets reached competitive performance without using real defect examples in training. Moreover, the observed improvement was not limited to the real-world benchmark used during development, but was also confirmed on an independently reproduced dataset, providing additional evidence that the optimized synthetic data improve transfer under related acquisition conditions.

The study also indicates that only a subset of rendering parameters needs to be tuned with high precision to obtain effective synthetic data, whereas others have comparatively minor impact. This is encouraging for practical deployment, as it suggests that a limited number of well-chosen configuration studies can substantially reduce the sim-to-real gap. At the same time, our evaluation is restricted to a single component type, inspection setup, and defect class and relies on a comparatively small real-world dataset. As a result, quantitative conclusions about robustness under distribution shifts and across broader application domains remain unclear.

Future work will therefore concentrate on extending both the scope and automation of the approach. On the one hand, we plan to validate the pipeline for additional defect types, geometries, and sensor configurations and to incorporate larger and more diverse real datasets with dedicated test splits. On the other hand, we aim to decouple image generation from dataset assembly, enabling more flexible experimental designs on a shared pool of parameter-annotated synthetic images. In the longer term, such studies may support task-level guidelines for configuring synthetic data pipelines and contribute to reducing the effort required to deploy learning-based defect detection in industrial practice.

8 Appendix

Table 5 presents the complete two-level Plackett–Burman screening design used to evaluate the influence of the synthetic dataset generation parameters. The table lists all experimental runs, the corresponding factor settings, and the resulting mAP values used to identify the most influential factors.

Table 5 Complete two-level Plackett-Burman screening design used to identify the most influential factors in the synthetic dataset generation process. All experimental runs and their resulting mAP values are reported.

Dataset	Positioning area	Defect object	Defect size	Defect		Penetration depth rand. int.	Disp. mapping strength	Spot angle rand. int.	Spot angle rand. int.	Light distance rand. int.	Light angle; rand. int.	Light intensity	Intensity rand. int.	Color rand. interval	Resolution	Resolution rand. int.	FOV int.	FOV angle rand. int.	Spool translation rand. int.	Camera orientation	Scene	Dummy	Material	mAP
1	1	-1	-1	1	-1	-1	1	1	1	-1	-1	-1	-1	-1	1	1	-1	1	-1	-1	1	1	1	0.445
2	1	-1	-1	-1	1	1	-1	1	-1	1	1	1	1	1	-1	-1	1	-1	-1	1	1	1	1	0.138
3	1	-1	1	1	1	1	-1	-1	1	-1	-1	-1	-1	-1	1	-1	-1	1	1	1	1	1	1	0.074
4	-1	-1	1	-1	-1	1	1	-1	1	1	1	-1	1	-1	-1	1	-1	1	-1	1	1	1	-1	0.314
5	1	1	1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.651
6	-1	-1	1	-1	1	1	1	1	-1	-1	1	1	-1	1	1	-1	-1	-1	-1	1	1	1	-1	0.163
7	-1	1	1	1	-1	-1	-1	1	-1	-1	1	1	1	-1	1	1	-1	-1	-1	1	1	-1	1	0.023
8	-1	1	1	1	-1	-1	-1	1	-1	-1	1	1	-1	1	-1	-1	1	1	1	1	1	1	-1	0.370
9	-1	-1	1	-1	1	1	1	1	-1	-1	1	1	1	-1	-1	1	-1	1	1	1	-1	-1	1	0.019
10	-1	-1	-1	1	-1	-1	-1	1	1	1	1	1	-1	1	1	1	-1	1	1	1	1	-1	-1	0.572
11	1	1	1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	-1	-1	1	1	-1	1	1	-1	1	0.151
12	1	-1	1	1	1	1	-1	-1	1	-1	-1	-1	1	1	-1	1	1	-1	-1	-1	1	-1	-1	0.640
13	1	1	-1	-1	-1	-1	1	-1	1	-1	1	1	1	1	1	-1	-1	1	-1	-1	1	-1	1	0.434
14	1	1	-1	-1	-1	-1	1	-1	-1	1	-1	1	-1	-1	-1	1	1	1	1	1	1	1	-1	0.608
15	1	-1	-1	-1	1	1	-1	1	-1	1	1	1	-1	-1	1	1	1	-1	1	1	1	-1	-1	0.541
16	-1	1	-1	1	1	1	1	-1	-1	1	-1	-1	1	-1	1	-1	1	1	-1	-1	1	1	-1	0.386
17	-1	-1	-1	1	-1	-1	-1	1	-1	1	1	1	1	-1	-1	-1	1	-1	1	1	1	1	1	0.046
18	1	-1	-1	1	-1	-1	1	1	1	-1	-1	-1	1	1	-1	-1	1	-1	1	1	1	-1	-1	0.672
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.006

Table 5 (Continued).

Dataset	Positioning area	Defect object	Defect size	Defect size rand. int.	Penetration depth rand. int.	Disp. mapping strength	Spot angle rand. int.	Spot angle	Light distance rand. int.	Light angle; rand. int.	Light intensity rand. int.	Intensity rand. int.	Color rand. interval	Resolution	Resolution rand. int.	FOV	FOV rand. int.	Spool angle rand. int.	Camera translation rand. int.	Camera orientation	Scene	Dummy	Material	mAP
20	-1	1	-1	-1	1	-1	1	1	1	-1	-1	-1	1	-1	1	1	-1	1	-1	-1	-1	1	1	0.028
21	-1	1	-1	1	1	1	-1	-1	-1	1	-1	-1	1	-1	1	-1	-1	-1	1	1	1	-1	1	0.124
22	1	1	1	-1	-1	-1	1	-1	-1	1	-1	1	1	1	1	-1	-1	-1	1	-1	-1	1	-1	0.616
23	-1	1	-1	-1	1	-1	1	1	1	-1	-1	1	-1	1	-1	-1	1	-1	1	1	1	-1	-1	0.303
24	-1	-1	1	-1	-1	1	-1	1	1	1	-1	-1	1	1	-1	1	1	-1	1	-1	-1	-1	1	0.002

Disclosures

The authors declare that there are no financial interests, commercial affiliations, or other potential conflicts of interest that could have influenced the objectivity of this research or the writing of this paper. Parts of this work were previously published in the conference proceedings: Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications III. Vol. 13459. SPIE, 2025.⁷³ Large language models (LLMs) were used to improve the language and grammar of this manuscript.

Code and Data Availability

The code and the synthetic datasets generated and analyzed during the current study are available from the corresponding author upon request.

Acknowledgments

This research was funded by the German Federal Ministry for Economic Affairs and Energy (Grant No. 20Q2109D).

References

1. L. Bosnar, H. Hagen, and P. Gospodnetic, "Procedural defect modeling for virtual surface inspection environments," *IEEE Comput. Graph. Appl.* **43**, 13–22 (2023).
2. D. Schraml and G. Notni, "Synthetic training data in AI-driven quality inspection: the significance of camera, lighting, and noise parameters," *Sensors* **24**, 649 (2024).
3. F. Seiler, V. Eichinger, and I. Effenberger, "Synthetic data generation for AI-based machine vision applications," *Electron. Imaging* **36**, 276–1–276–5 (2024).
4. R. L. Plackett and J. P. Burman, "The design of optimum multifactorial experiments," *Biometrika* **33**(4), 305–325 (1946).
5. S. Niu et al., "Defect image sample generation with GAN for improving defect recognition," *IEEE Trans. Autom. Sci. Eng.* **17**(3), 1611–1622 (2020).
6. X. Wu et al., "Deep learning-based generic automatic surface defect inspection (ASDI) with pixelwise segmentation," *IEEE Trans. Instrum. Meas.* **70**, 1–10 (2021).
7. S. Jain et al., "Synthetic data augmentation for surface defect detection and classification using deep learning," *J. Intell. Manuf.* **33**, 1007–1020 (2022).
8. S. Meister et al., "Synthetic image data augmentation for fibre layup inspection processes: techniques to enhance the data set," *J. Intell. Manuf.* **32**, 1767–1789 (2021).
9. R. Wang et al., "Defect transfer GAN: diverse defect synthesis for data augmentation," arXiv:2302.08366 [cs] (2023).
10. O. De Mitri et al., "Synthetic data generation for improvement of machine learning-based optical quality control: a practical approach in the welding context," *Proc. SPIE* **12621**, 126210Y (2023).
11. Y.-J. Han and H.-J. Yu, "Fabric defect detection system using stacked convolutional denoising auto-encoders trained with synthetic defect data," *Appl. Sci.* **10**, 2511 (2020).
12. J. P. Yun et al., "Automated defect inspection system for metal surfaces based on deep learning and data augmentation," *J. Manuf. Syst.* **55**, 317–324 (2020).
13. G. Valvano et al., "Controllable image synthesis of industrial data using stable diffusion," in *IEEE/CVF Winter Conf. on Appl. of Comput. Vision (WACV)*, IEEE, Waikoloa, HI, USA, pp. 5342–5351 (2024).
14. A. Ramesh et al., "Zero-shot text-to-image generation," in *Proc. 38th Int. Conf. Mach. Learn.*, pp. 8821–8831 (2021).
15. C. Saharia et al., "Photorealistic text-to-image diffusion models with deep language understanding," *Adv. Neural Inf. Process. Syst.* pp. 36479–36494, Curran Associates, Inc. (2022).
16. R. Wang, S. Schmedding, and M. F. Huber, "Improving the effectiveness of deep generative data," in *IEEE/CVF Winter Conf. on Appl. of Comput. Vision (WACV)*, IEEE, Waikoloa, HI, USA, pp. 4910–4920 (2024).
17. G. Paulin and M. Ivacic-Kos, "Review and analysis of synthetic dataset generation methods and techniques for application in computer vision," *Artif. Intell. Rev.* **56**, 9221–9265 (2023).
18. D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: surprisingly easy synthesis for instance detection," in *IEEE Int. Conf. on Comput. Vision (ICCV)*, IEEE, Venice, pp. 1310–1319 (2017).
19. D. Mery, "Aluminum casting inspection using deep object detection methods and simulated ellipsoidal defects," *Mach. Vision Appl.* **32**, 72 (2021).
20. C.-L. Li et al., "CutPaste: self-supervised learning for anomaly detection and localization," in *IEEE/CVF Conf. Comput. Vision and Pattern Recognit. (CVPR)*, IEEE, Nashville, TN, USA, pp. 9659–9669 (2021).

21. M. Mohammadzadeh et al., “AI-driven crack detection for remanufacturing cylinder heads using deep learning and engineering-informed data augmentation,” *Automation* **5**, 578–596 (2024).
22. R. Rill-García, E. Dokladalova, and P. Dokládál, “Syncrack: improving pavement and concrete crack detection through synthetic data generation,” in *Proc. 17th Int. Joint Conf. on Comput. Vision, Imaging and Comput. Graph. Theory and Appl.*, pp. 147–158, SCITEPRESS (2022).
23. M. Haselmann and D. Gruber, “Supervised machine learning based surface inspection by synthesizing artificial defects,” in *16th IEEE Int. Conf. on Mach. Learn. and Appl. (ICMLA)*, IEEE, Cancun, Mexico, pp. 390–395 (2017).
24. M. Haselmann and D. P. Gruber, “Pixel-wise defect detection by CNNs without manually labeled training data,” *Appl. Artif. Intell.* **33**, 548–566 (2019).
25. D. Holst, O. Schmedemann, and T. Schüppstuhl, “An edge is all you need: cracking the code for generating synthetic datasets for robust crack detection models,” *Proc. CIRP* **138**, 1097–1102 (2026).
26. H. Su et al., “Render for CNN: viewpoint estimation in images using CNNs trained with rendered 3D model views,” in *IEEE Int. Conf. on Comput. Vision (ICCV)*, IEEE, Santiago, Chile, pp. 2686–2694 (2015).
27. J. Tremblay et al., “Training deep networks with synthetic data: bridging the reality gap by domain randomization,” in *IEEE/CVF Conf. Comput. Vision and Pattern Recognit. Workshops (CVPRW)*, IEEE, Salt Lake City, UT, pp. 1082–10828 (2018).
28. D. Schoepflin et al., “Synthetic training data generation for visual object identification on load carriers,” *Proc. CIRP* **104**, 1257–1262 (2021).
29. P. Martinez-Gonzalez et al., “UnrealROX: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation,” *Virtual Reality* **24**, 271–288 (2020).
30. K. Greff et al., “Kubric: a scalable dataset generator,” in *IEEE/CVF Conf. on Comput. Vision and Pattern Recognit. (CVPR)*, IEEE, New Orleans, LA, USA, pp. 3739–3751 (2022).
31. M. Denninger et al., “BlenderProc2: a procedural pipeline for photorealistic rendering,” *J. Open Source Softw.* **8**, 4901 (2023).
32. S. Moonen et al., “CAD2Render: a modular toolkit for GPU-accelerated photorealistic synthetic data generation for the manufacturing industry,” in *IEEE/CVF Winter Conf. Appl. of Comput. Vision Workshops (WACVW)*, IEEE, Waikoloa, HI, USA, pp. 583–592 (2023).
33. A. Raistrick et al., “Infinite photorealistic worlds using procedural generation,” in *IEEE/CVF Conf. on Comput. Vision and Pattern Recognit. (CVPR)*, IEEE, Vancouver, BC, Canada, pp. 12630–12641 (2023).
34. S. I. Nikolenko, “Synthetic data for deep learning,” arXiv:1909.11512 [cs] (2019).
35. Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh, “How useful is photo-realistic rendering for visual learning?” *Lect. Notes Comput. Sci.* **9915**, 202–217 (2016).
36. L. Eversberg and J. Lambrecht, “Generating images with physics-based rendering for an industrial object detection task: realism versus domain randomization,” *Sensors* **21**, 7901 (2021).
37. A. Dosovitskiy et al., “FlowNet: learning optical flow with convolutional networks,” in *IEEE Int. Conf. on Comput. Vision (ICCV)*, IEEE, Santiago, pp. 2758–2766 (2015).
38. J. Tobin et al., “Domain randomization for transferring deep neural networks from simulation to the real world,” in *IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, pp. 23–30 (2017).
39. A. Prakash et al., “Structured domain randomization: bridging the reality gap by context-aware synthetic data,” in *Int. Conf. Rob. Autom. (ICRA)*, pp. 7249–7255 (2019).
40. A. Boikov et al., “Synthetic data generation for steel defect detection and classification using deep learning,” *Symmetry* **13**, 1176 (2021).
41. A. Cordier, P. Gutierrez, and V. Plessis, “Improving generalization with synthetic training data for deep learning based quality inspection,” arXiv:2202.12818 [cs] (2022).
42. P. Gutierrez et al., “Synthetic training data generation for deep learning based quality inspection,” *Proc. SPIE* **11794**, 1179403 (2021).
43. D. Lebert et al., “Synthetic data generation for surface defect detection,” *Lect. Notes Comput. Sci.* **13446**, 198–208 (2022).
44. V. Hoskere, Y. Narazaki, and B. F. Spencer, “Physics-based graphics models in 3D synthetic environments as autonomous vision-based inspection Testbeds,” *Sensors* **22**, 532 (2022).
45. L. Bonfiglioli et al., “The eyecandies dataset for unsupervised multimodal anomaly detection and localization,” *Lect. Notes Comput. Sci.* **13845**, 459–475 (2023).
46. S. Mohanty, E. Su, and C.-C. Ho, “Reinforcement learning optimized digital twin based synthetic data generation for defect detection of titanium spacer,” *Proc. SPIE* **12623**, 1262308 (2023).
47. S. Mohanty, E. Su, and C.-C. Ho, “Enhancing titanium spacer defect detection through reinforcement learning-optimized digital twin and synthetic data generation,” *J. Electron. Imaging* **33**, 013021 (2024).
48. I. Effenberger, F. Seiler, and V. Eichinger, “Synthetische Daten für die Automatisierung mit KI/Synthetic data for AI-based automation,” *Werkstattstechnik online* **114**(9), 490–496 (2024).
49. C. Schorr et al., “A scalable synthetic data creation pipeline for AI-based automated optical quality control,” in *Proc. 14th Int. Conf. on Simul. and Model. Methodol., Technol. and Appl.*, SCITEPRESS, Dijon, France, pp. 37–46 (2024).

50. A. R. Singh et al., “Generating synthetic training images to detect split defects in stamped components,” *IEEE Trans. Ind. Inform.* **20**, 4816–4827 (2024).
51. A. Martinet et al., “Procedural modeling of cracks and fractures,” in *Proc. Shape Model. Appl.*, IEEE, Genova, Italy, pp. 346–349 (2004).
52. M.-G. Retzlaff et al., “Potential and challenges of using computer graphics for the simulation of optical measurement systems,” in *Tagungsband*, pp. 322–329, AMA Service GmbH, Nürnberg, Germany (2016).
53. M.-G. Retzlaff, M. Richter, and T. Langle, “Combining synthetic image acquisition and machine learning: accelerated design and deployment of sorting systems,” *Forum Bildverarbeitung*, p. 49 (2016).
54. M.-G. Retzlaff et al., “Synthesizing images using parameterized models for automated optical inspection (AOI),” *Technisches Messen* **82**, 251–261 (2015).
55. T. Dahmen et al., “Digital reality: a model-based approach to supervised learning from synthetic data,” *AI Perspect.* **1**, 2 (2019).
56. L. Bosnar et al., “Image synthesis pipeline for surface inspection,” in *Proc. LEVIA 20: Leipzig Symp. Vis. Appl.* (2020).
57. L. Bosnar et al., “Texture synthesis for surface inspection,” in *LEVIA’22 : Leipzig Symp. on Visualization in Appl.*, Leipzig University (2022).
58. O. Schmedemann et al., “Procedural synthetic training data generation for AI-based defect detection in industrial surface inspection,” *Proc. CIRP* **107**, 1101–1106 (2022).
59. J. Fulir et al., “Synthetic data for defect segmentation on complex metal surfaces,” in *IEEE/CVF Conf. Comput. Vision and Pattern Recognit. Workshops (CVPRW)*, IEEE, Vancouver, BC, Canada, pp. 4424–4434 (2023).
60. J. Fulir et al., “SYNOPSIS: image synthesis pipeline for machine vision in metal surface inspection,” arXiv:2410.14844 [cs] (2024).
61. R. L. Cook, “Shade trees,” *ACM SIGGRAPH Comput. Graph.* **18**, 223–231 (1984).
62. T.-Y. Lin et al., “Microsoft COCO: common objects in context,” arXiv:1405.0312 [cs] (2015).
63. M. Everingham et al., “The Pascal visual object classes (VOC) challenge,” *Int. J. Comput. Vision* **88**, 303–338 (2010).
64. Blender Foundation, “Blender,” <https://www.blender.org/> (accessed 12 June 2026).
65. X. Li et al., “Deep learning-based defects detection of certain aero-engine blades and vanes with DDSC-YOLOv5s,” *Sci. Rep.* **12**, 13067 (2022).
66. H. Shang et al., “Global prior transformer network in intelligent borescope inspection for surface damage detection of aeroengine blade,” *IEEE Trans. Ind. Inform.* **19**, 8865–8877 (2023).
67. Y. Zhao et al., “DETRs beat YOLOs on real-time object detection,” arXiv:2304.08069 [cs] (2024).
68. I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” arXiv:1711.05101 [cs] (2019).
69. S. Ren et al., “Faster R-CNN: towards real-time object detection with region proposal networks,” arXiv:1506.01497 [cs] (2016).
70. K. He et al., “Deep residual learning for image recognition,” in *IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, pp. 770–778 (2016).
71. J. Deng et al., “ImageNet: a large-scale hierarchical image database,” in *IEEE Conf. on Comput. Vision and Pattern Recognit.*, IEEE, Miami, FL, pp. 248–255 (2009).
72. S. Ruder, “An overview of gradient descent optimization algorithms,” arXiv:1609.04747 [cs] (2017).
73. O. Schmedemann et al., “Introducing a tool for synthetic defect image data generation: enhancing industrial surface inspection,” *Proc. SPIE* **13459**, 134590E (2025).

Ole Schmedemann works as a research associate at the Institute of Aircraft Production Technology at Hamburg University of Technology (TUHH). He received his BS and MS degrees in mechanical engineering from TUHH in 2016 and 2019, respectively, and is currently pursuing his PhD. His current research interests include synthetic training data generation for visual inspection based on deep learning. He is a member of SPIE.

Biographies of the other authors are not available.