INTERNATIONAL DESIGN CONFERENCE – DESIGN 2024
https://doi.org/10.1017/pds.2024.226

Towards a process for the creation of synthetic training data for AI-computer vision models utilizing engineering data

Sebastian Schwoch ✉, Maximilian Peter Dammann, Johannes Georg Bartl, Maximilian Kretzschmar, Bernhard Saske and Kristin Paetzold-Byhain

Technische Universität Dresden, Germany

✉ sebastian.schwoch@tu-dresden.de

Abstract

Artificial Intelligence-based Computer Vision models (AI-CV models) for object detection can support various applications over the entire lifecycle of machines and plants such as monitoring or maintenance tasks. Despite ongoing research on using engineering data to synthesize training data for AI-CV model development, there is a lack of process guidelines for the creation of such data. This paper proposes a synthetic training data creation process tailored to the particularities of an engineering context addressing challenges such as the domain gap and methods like domain randomization.

Keywords: artificial intelligence (AI), synthetic training data, annotation automation, domain gap, process improvement

1. Introduction

The utilization of Artificial Intelligence-based Computer Vision models (AI-CV model) can support a wide range of processes and applications along the life cycle of technical systems such as machines and plants by detecting components in photo and video data (Zhou *et al.*, 2023). Current research on AI-CV models in engineering focuses on the detection of object categories like screws, bearings or pipes by utilizing training data sets with generic categories (Drost *et al.*, 2017). In most industry use cases not the category but a specific component of a technical system is of interest. Hence, training data sets are needed which replace and extend the generic categories with identifiers for the individual system components. If these identifiers are article IDs of a product data management (PDM) system, the AI-CV models can be linked to the entire data backend of the components (see Figure 1).

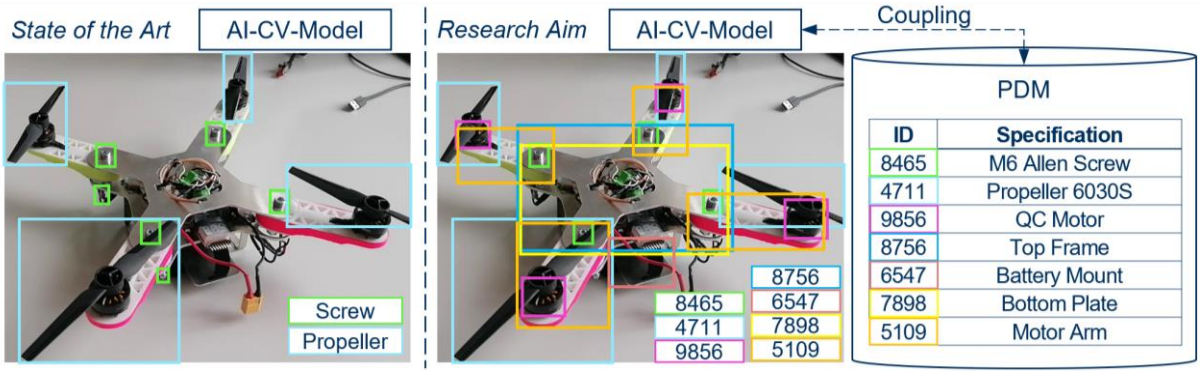

Figure 1. Left: Category detection; Right: Component detection with PDM coupling

In order to provide an extensive, high quality training data set as basis for AI-CV model development (Jain *et al.*, 2020), currently thousands of photographs are manually annotated with the object locations and object categories in a time-consuming and error-prone process (Valtchev and Wu, 2021; Assadzadeh *et al.*, 2022). The annotation process consumes a large part of the time needed to create training data sets for AI-CV. For example, Ono et al. state an annotation time of approx. two weeks for 7.000 images by five annotators (Ono *et al.*, 2023).
In an industrial context, this means that a physical instance of the technical system must exist before the time-consuming process of capturing and manually annotating photographs can take place. Hence, the creation of training data is delayed until near-final prototypes are available late in the development phase. In specialized machine and plant engineering where prototypes are uncommon, AI-CV development only starts post construction, preventing their early use in these fields (see Figure 2).

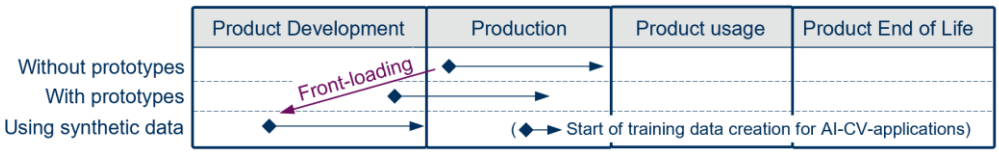

Figure 2. Front-loading of AI-CV-Model development when using synthetic training data

The use of synthetic training data - virtually generated images and annotations through rendering - is ideal for the front-loading of AI-CV model development (see Figure 2). Particularly in the context of engineering, the geometry data of technical systems generated during development provide an excellent basis for deriving synthetic training data (Kohtala and Steinert, 2021) and automating renderings and annotations saves additional time.
However, no current framework supports the creation of synthetic training data sets which enable AI-CV models to detect individual components. Therefore, this research aims to develop a framework that maintains the link between engineering data (geometry data and article IDs) and training data throughout the entire AI-CV model development process. This paper focuses on the definition of a process for generating synthetic training data sets in an engineering context and answers the following research question:

> *How does a process have to be designed that generates comprehensive and high-quality synthetic training data sets for AI-CV models from engineering data?*

Future research will complement and support the proposed process by investigating related methods, tools and an accompanying data model.

Figure 3. The research objects of a framework for utilization of synthetic training data for AI-CV applications in engineering

2. State of the art

2.1. AI development processes

Modern guidelines for the development of AI-CV models increasingly originate from industry and view the ML-based application development as a circular process that is intended to enable greater agility. These modern guidelines include the Microsoft Team Data Science-Process (TDSP) (Microsoft, 2023) and the DIN SPEC 13266 (DIN Deutsches Institut für Normung e.V., 2020). While the TDSP describes the general development of ML models, DIN SPEC 13266 focuses on deep learning-based CV models.

https://doi.org/10.1017/pds.2024.226 Published online by Cambridge University Press

Other approaches include the Data Science Process Model (DASC-PM) (Schulz *et al.*, 2022) and Engineering Data Driven Applications (EDDA) (Hesenius *et al.*, 2019).
Another focus is the development of software frameworks for the practical implementation of ML and AI-CV in particular (Orhei *et al.*, 2021; Stephen Gould, 2012). These software implementations are usually based on generalized processes or frameworks that facilitate ML application and provide necessary tools for AI development with notable examples being scikit-learn and TensorFlow.
Despite numerous guidelines, there's a lack of adapted approaches for developing AI-CV models in engineering product development and notably in handling synthetic training data for these applications.

2.2. Synthetic training data

While high-quality data sets for everyday objects are readily available (Lin *et al.*, 2014; Downs *et al.*, 2022; He *et al.*, 2022), there is a deficit for industry-specific applications. In such cases, data sets have to be manually created from scratch, which poses a significant challenge for implementing ML-based object detection models in industrial applications (Wong *et al.*, 2019).
Consequently, there is increasing interest in using synthetic training data derived from the geometry models of components and generated in a controlled environment. This approach enables precise control of synthesis parameters and the automatic generation of pixel-precise annotations (Mayer *et al.*, 2018), which allows for faster generation of extensive data sets avoiding human inaccuracies (Ono *et al.*, 2023).
Synthetic data has a promising role in the realm of product development (Kohtala and Steinert, 2021). Initially, the lack of physical components in the early stages means there is no opportunity to capture photos for training data. Traditionally used tools, such as CAD systems and the geometric data created with them open up a novel pathway for creating synthetic training data sets (Pasanisi *et al.*, 2023). While this method is scalable due to its highly automatable nature and independence from a physical prototype, effectively utilizing this data is not trivial due to the often-encountered distribution mismatch between training and real test data, which leads to a significant performance drop (Chen *et al.*, 2018). In response to this challenge several techniques have been explored to improve the generalization capability of models trained on predominantly synthetic data, which are discussed in the following sections. These approaches support the creation of training data sets that are not constrained by the limitations of real-world data, enabling highly scalable model development early in the development process. Additionally, they simplify and accelerate the process of adapting or expanding these datasets to specific problems while also enabling the simulation of events that are rare or impossible to replicate in reality.

2.2.1. Domain gap

Leveraging synthetic data introduces unique considerations in the AI development process. Render engines, which are grounded in mathematical models, employ assumptions which simplify complex real-world phenomena. The divergence between model-driven synthetic data and photos of real environments is termed the domain gap. If overlooked during the data generation process, this gap can diminish the generalization ability of CV models trained purely on synthetic data. (Park *et al.*, 2020)

2.2.2. Domain randomization and domain adaptation

Domain randomization is a method for synthetic data generation that artificially expands the parameter space of the training data set to overcome the domain gap. The method systematically varies parameters such as object pose, lighting, textures and backgrounds in non-realistic ways in order to guide the neural network to grasp the fundamental features of objects. Thereby the generalization ability of the CV model is improved (Salas *et al.*, 2020; Tremblay *et al.*, 2018).
A further method to overcome the domain gap is domain adaptation. In this approach, a model trained predominantly on synthetic data is enhanced by incorporating a subset of real-world data during its training phase. Multiple studies have demonstrated that adding a relatively small set of real training images into a largely synthetic data set can significantly improve the models ability to generalize in the target domain (Horváth *et al.*, 2023; Tremblay *et al.*, 2018). However, using realistic renderings to handle the domain gap is also investigated in literature (Denninger *et al.*, 2019).

https://doi.org/10.1017/pds.2024.226 Published online by Cambridge University Press

3. Constraints and requirements for the process

In this chapter constraints and requirements for the process are derived from the requirements and the availability of engineering data - additionally typical requirements for engineering AI systems are taken into account from literature (Ahmad *et al.*, 2023; Schulz *et al.*, 2022).

The specific context in which the AI-CV model will be used needs to be considered for the implementation of the process. One example is the support of maintenance work with augmented reality devices. In this task, an AI-CV model only needs to be able to identify the components required for the respective maintenance procedure. The proposed process must support both such a delimited problem definition as well as the creation of a training data set for all components of a system.

Also, different AI-CV models require different formatting of the generated training data set. For example, Mask R-CNN requires a training data set in the COCO format (He *et al.*, 2017), while YOLO models define their own formatting (Hussain, 2023). Furthermore, the desired type of object identification also influences which annotations must be included in the training data set, ranging from simply naming the category of the objects on image/video data to pixel-precise differentiation of several instances of an object category (Lin *et al.*, 2014).

Finally, the engineering context defines various constraints on the process. It can be assumed that a machine-readable data source of the components exists, for which an AI-CV model should be developed. This database needs to contain a representative geometric description of the components to be recognized (e.g., CAD files) and a unique identification of these components (e.g., article ID). Examples of suitable data sources are PDM systems.

Table 1 outlines the requirements for a comprehensive AI-CV model creation process based on available data in engineering given the mentioned constraints.

Table 1. Process requirements

Process
Adaptation of existing guidelines for the development of AI-(CV) systems.
The process must be fully automatable.
Synthetic training data
The domain gap must be taken into account and measures for dealing with it have to be integrated.
The generation of synthetic training data includes the generation of rendered images and annotation data.
The generated synthetic training data must be reproducible.
Integration of engineering methods and tools
The input is geometry information and product structure information from engineering data sources (PDM).
The continuous link between the engineering data and the synthetic training data must be ensured.
The generated synthetic training data and parameters of the process must be documented and archived.
Integration of AI methods and tools
Modelling data can be formatted according to any data sets (e.g., YOLO, COCO, etc.).
Data quality
Generated data sets must contain complete, up to date, sufficient and non-redundant data (class symmetry).
The correct and unambiguous assignment of synthetic images and annotation data must be ensured.
The annotation must be pixel-precise, exceeding the quality of manual annotations.

4. A process for the creation of synthetic training data utilizing engineering data

4.1. Process overview

A process for generating a synthetic *Training Data Set* shown in Figure 4 explicitly takes into account the requirements listed in Table 1.

https://doi.org/10.1017/pds.2024.226 Published online by Cambridge University Press

It consists of four process steps: "Data Acquisition", "Generation of Synthetic Images", "Generation of Annotation Data" and "Formatting of Training Data Set". While Figure 4 depicts a linear process, iterations within the process are allowed.

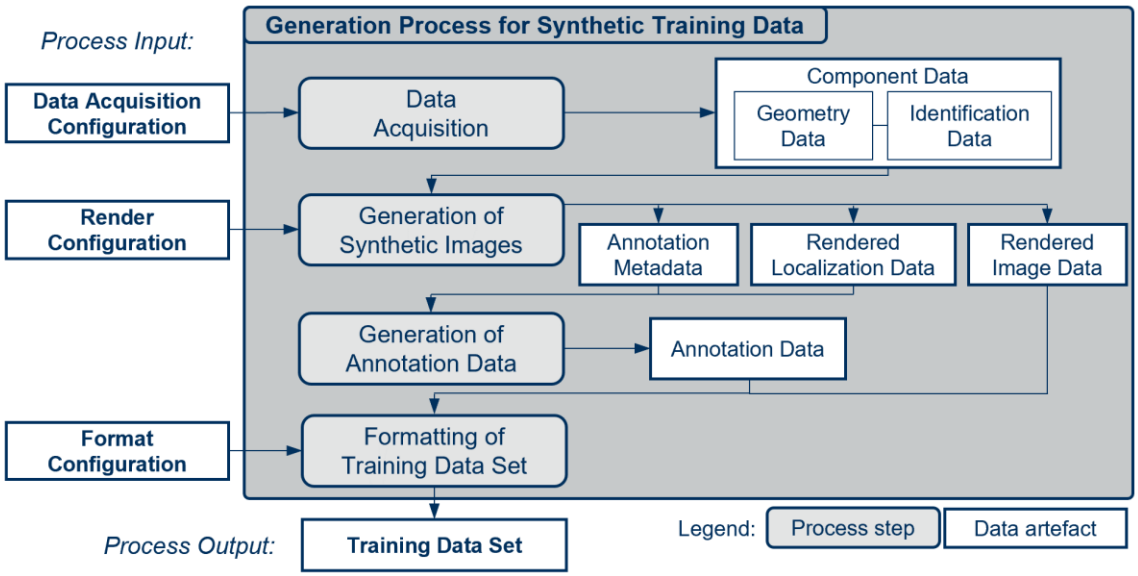

Figure 4. Process overview with process input (left), process output (bottom), process steps (middle) and preliminary data artefacts (right)

The "Data Acquisition" process step includes selecting, extracting and preparing the data from engineering data sources specified within the *Data Acquisition Configuration*. The output, *Component Data*, includes the *Geometry Data* of all components needed for the *Training Data Set* as well as the *Identification Data* related to this geometry. The *Identification Data* containing the ID of each component in the data backend. A tight coupling between *Geometry Data* and *Identification Data* is needed to maintain these links throughout the following process steps. Within the "Generation of Synthetic Images" process step images are rendered using the *Component Data*. The rendering settings are specified in the *Rendering Configuration*. After the "Generation of Synthetic Images" *Rendered Image Data*, *Rendered Localization Data* and *Annotation Metadata* are created. The *Rendered Image Data* are the rendered pixel-based images of the system used for the training of the AI-CV-models. The *Rendered Localization Data* are also images rendered with the same camera settings as the *Rendered Image Data* but using a clear color-rendering of the components to pixel-precisely distinguish the components (see Chapter 5, Figure 9). The *Annotation Metadata* stores information about the components visible on the *Rendered Image Data* and the color-coding used for the *Rendered Localization Data*. The *Annotation Metadata* and the *Rendered Localization Data* are inputs for the third process step "Generation of Annotation Data". Here, the pixel-precise color-coding of the *Rendered Localization Data* is computed into annotations, which can be used by the training algorithms of the AI-CV-models. This includes the derivation of bounding box and mask coordinates for each component in every *Rendered Localization Data*, resulting in the *Annotation Data*. The last process step "Formatting of Training Data" converts the *Annotation Data* and *Rendered Image Data* to one or more formats of training data, specified by the *Format Configuration*.
The process results are one or more complete, consistent, high-quality *Training Data Sets* that contain the IDs of the individual components of the system as categories and the required format for the subsequent training algorithm. The *Training Data Sets* are designed for direct use with the corresponding training algorithm. The individual process steps are described in detail below.

4.2. Process step “Data Acquisition”

In the "Data Acquisition" process step shown in Figure 5, *Component Data* is created, which encompasses both the geometry and a unique identification of the components.

https://doi.org/10.1017/pds.2024.226 Published online by Cambridge University Press

The input *Data Acquisition Configuration* includes information on accessing the data sources from which the *Component Data* is extracted (e.g., PDM systems or overarching concepts as in Schwoch et al. (2023)) and specifies which components will be included in the *Training Data Set*.

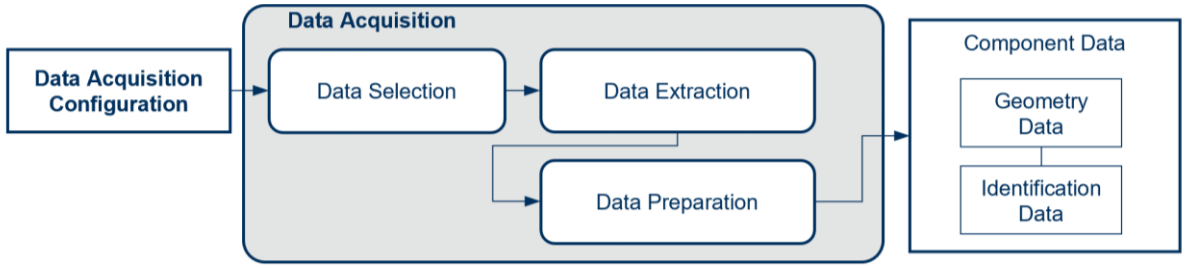

Figure 5. Input, activities and output of process step "Data Acquisition"

Within the "Data Selection" activity all included components and their structural relationships are analyzed and selected in the specified data sources. During "Data Extraction", the geometry and identification information of the components are loaded from the data sources. In "Data Preparation", the geometry is tessellated and formatted in a format that can be read by the rendering tool and the hierarchy and transformation matrices are extracted (Dammann *et al.*, 2022). During all activities, it is essential to maintain the connection between the geometry and identification information in the data sources.

The output is the *Component Data*, consisting of *Geometry Data* and *Identification Data*. The *Geometry Data* includes the geometric description of the components, their hierarchy and transformation matrices. The *Identification Data* contains the respective unique identifier of the data sources for each component (e.g., the article ID in the PDM system). *Geometry Data* and *Identification Data* are closely linked, allowing images, masks and annotations to be associated with the respective IDs of the data backend.

4.3. Process step “Generation of Synthetic Images”

In the "Generation of Synthetic Images" process step (see Figure 6), the *Component Data* is first imported into the used rendering environment. The import includes both the *Geometry Data* in the form of the tessellated geometries, as the basis for rendering, and the *Identification Data*.

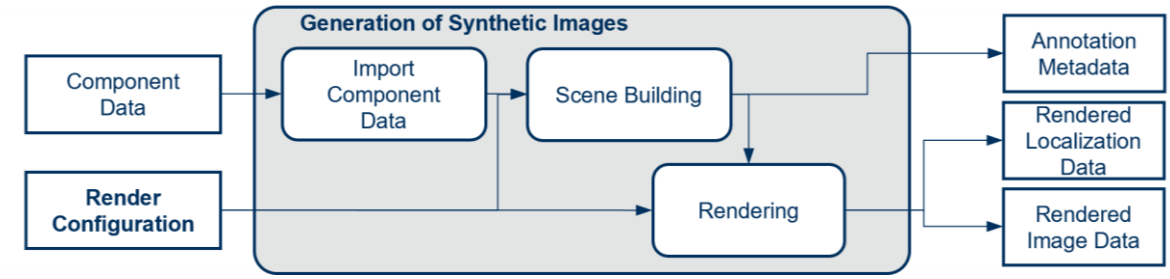

Figure 6. Input, activities and output of process step "Generation of Synthetic Images"

The scene graph is then built up. The *Geometry Data* is used to reproduce the component hierarchy in the scene graph and to assign the correct geometry information and transformation matrices to the objects in the scene graph. By using the unique identifiers of the *Identification Data*, it is possible to link the objects in the scene graph with the component information in external systems like PDM (Stelzer *et al.*, 2012). Information from the *Render Configuration*, such as camera transformations, lighting or materials is also used to create the scene graph. The *Render Configuration* has a central influence on the generalization of the AI-CV models trained on the generated *Training Data Set* to real environments (e.g., by using domain randomization or realistic renderings). After "Scene Building", the first output is the *Annotation Metadata* that stores information about the visible components in the *Rendered Image Data* and the *Rendered Localization Data*, which are both created in the subsequent "Rendering" activity. In "Rendering", information from the *Render Configuration* can also be used to create post-processing effects such as glare, overexposure, vignetting or blurring.

https://doi.org/10.1017/pds.2024.226 Published online by Cambridge University Press

4.4. Process step “Generation of Annotation Data”

The third process step is called "Generation of Annotation Data", shown in Figure 7.

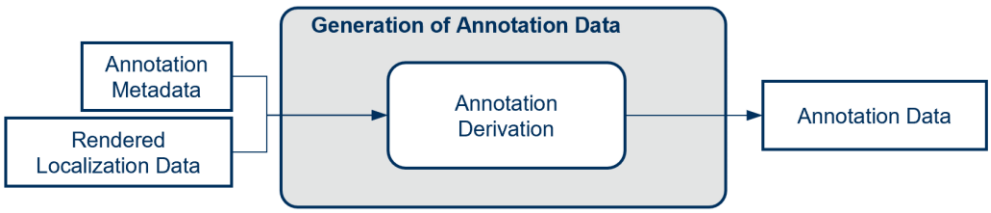

Figure 7. Input, activities and output of process step "Generation of Annotation Data"

In the process step, the annotations are derived from the *Annotation Metadata* and the *Rendered Localization Data*. This is done in the "Annotation Derivation" activity. Alternate mathematical descriptions can be generated from the *Rendered Localization Data* for the annotation information available in pixel images. For segmentation masks, for example, this can be a polygon or spline description, while reference points and dimensions are generated for bounding boxes. Finally, the component identifiers of the *Annotation Metadata* and the annotation descriptions derived from the *Rendered Localization Data* are combined in the *Annotation Data*.

4.5. Process step “Formatting of Training Data Set”

The final *Training Data Set* is created in the last process step "Formatting of Training Data Set" which can be used directly to train corresponding AI-CV models (see Figure 8).

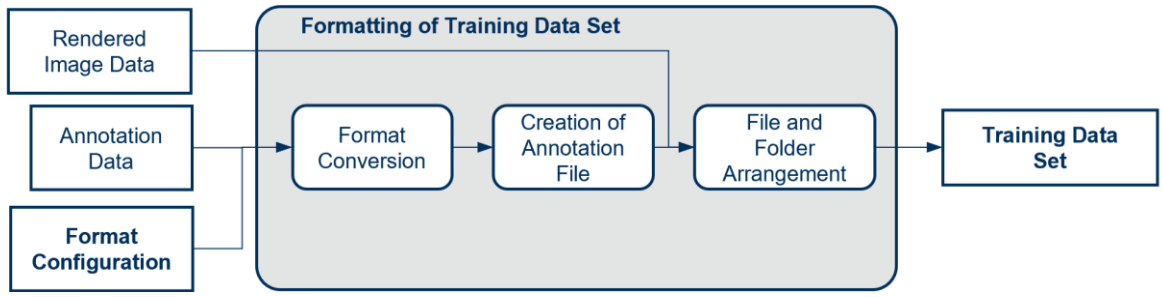

Figure 8. Input, activities and output of process step "Formatting of Training Data Set"

The input *Format Configuration* defines one or more formats of the *Training Data Set* to be generated. Examples of training data formats are the PASCAL VOC XML format (Everingham *et al.*, 2010), the COCO format (Lin *et al.*, 2014) and the YOLOv5 format (Jocher and Waxmann, 2023). It is possible to repeat this last process step with different format definitions. The different formats not only vary in terms of the file types of the annotations but also in the encoding of the various annotation information, like bounding box and mask definitions. In YOLOv5 format, bounding boxes are defined using relative center coordinates and width in both x and y directions, while COCO specifies the upper-left corner and widths. Similar variations apply to segmentation masks. Therefore, the annotation information is first converted into the specified format in the "Format Conversion" activity.

In the "Creation of Annotation File" activity the converted *Annotation Data* is put into one or more annotation files, depending on the chosen target format. Annotation files vary in file types (.csv, .json, .yaml-files) and inner structure, including category encoding differences. Nevertheless, all annotation files contain at least the following information: the path to the *Rendered Image Data* which the annotations describe, the category of the object delimited by the annotations and the form of the annotation regarding its type (bounding box, mask, etc.).

The last activity "File and Folder Arrangement" includes the arrangement of all files and folders of the *Training Data Set*. It needs to be ensured that paths from the *Annotation Data* to the *Rendered Image Data* are consistent to the resulting folder structure. Finally, the format consistent *Training Data Set* can be transferred to the machine or cloud application which performs the training.

https://doi.org/10.1017/pds.2024.226 Published online by Cambridge University Press

5. Exemplary implementation of the process

Figure 9 shows an exemplary creation of a *Training Data Set* for identifying quadcopter components.

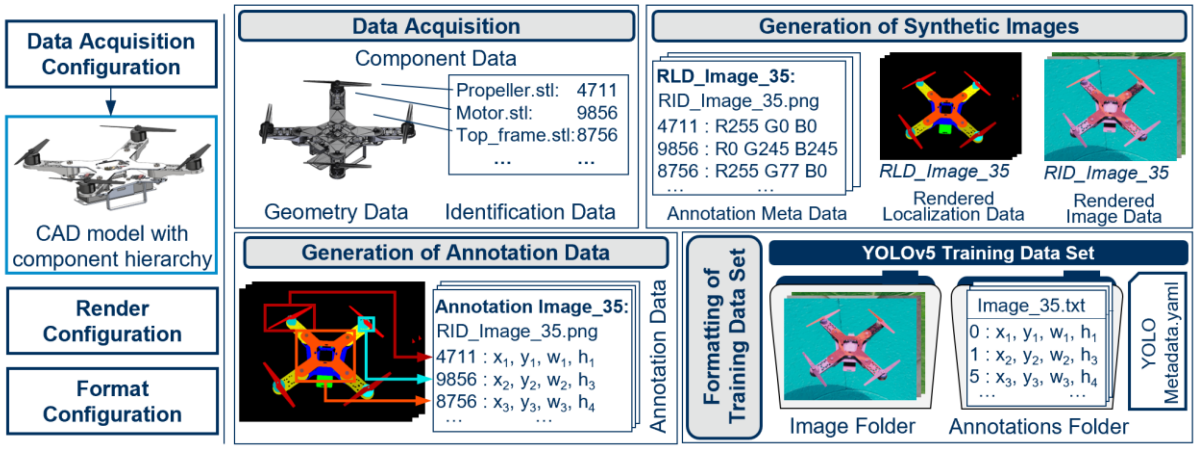

Figure 9. Creation of a training data set for the identification of quadcopter components

The *Data Acquisition Configuration* defines the use of a CAD-assembly of the quadcopter and the information, that all of its components should be included in the training data set. After the "Data Acquisition" the CAD-geometry is available as .stl-files representing the individual parts and assemblies. The *Identification Data* links each .stl.-file to a unique ID. After "Generation of Synthetic Images" multiple *Rendered Image Data* of the components are rendered according to the information within the *Render Configuration*. In this example domain randomization is used to overcome the domain gap. For each *Rendered Image Data* instance (e.g., *RID_Image_35* in Figure 9) exists a corresponding *Rendered Localization Data* (e.g., *RLD_Image_35* in Figure 9) which is rendered using identical camera settings but highlights each component with a unique color. *Annotation Metadata* stores the links between *Rendered Image Data* and *Rendered Localization Data* as well as the color-coding for the components in each *Rendered Localization Data* instance. After "Generation of Annotation Data" the color encoded component locations in the *Rendered Localization Data* are converted to segmentation masks and bounding box information. The final process step "Formatting of Training Data Set" results in a YOLOv5 format conform file and folder structure including the *Rendered Image Data* and corresponding bounding box annotations. In YOLOv5 format, categories are encoded numerically within annotation files while an additional metadata file (YOLO *metadata.yaml* in Figure 9) maps the numbers to their respective category (here 0 maps to category 4711, 1 to 9856 and 5 to 8756).

The main tools used to conduct the exemplary implementation of the process are the 3D graphics software Blender combined with a python API to automate the generation of synthetic images using domain randomization methods to diminish the domain gap between training data and a physical prototype. The training data set comprises 500 images, featuring 300 variations in camera perspective of the main assembly, along with varying lighting color and intensity, 25 unique object textures, 25 different backgrounds and distractors for simulating occlusion. The selection of parameters is guided by a series of experiments that investigated their impact on the performance of the object detection model, details of which will be presented in future work.

The resulting YOLOv5 Training Data Set is used to train a YOLOv7 model with an mAP@0.5 of 71,21% on real world images which corresponds to given literature (Tremblay *et al.*, 2018; Ono *et al.*, 2023). Figure 10 shows the real-time detection with the trained YOLOv7 model in video data.

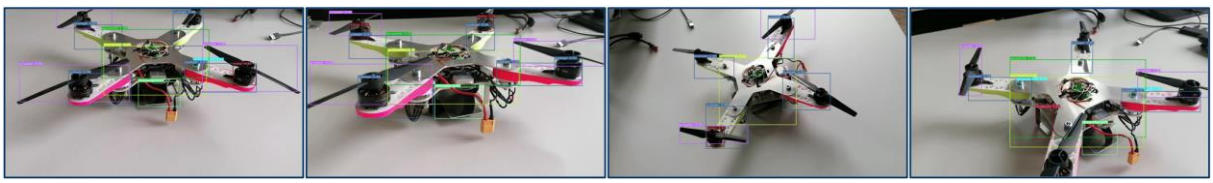

Figure 10. Example of component detection with the trained YOLOv7 model on video data

https://doi.org/10.1017/pds.2024.226 Published online by Cambridge University Press

6. Summary and outlook

Detecting specific components of a technical system using AI-CV models requires training data sets that include distinct identifiers and annotations for each component. The availability of geometric models in engineering encourages the automated generation of synthetic training data via methods like domain randomization. However, current research gives no support for the creation process of synthetic training data in an engineering context.

This contribution provides a support for engineers in the form of a consistent process derived from literature review and the analysis of the requirements of an engineering context. The process activities, their relationships and inputs/outputs are presented and specified. Finally, the successful application of the process is demonstrated with a YOLOv5 formatted training data set for the identification of quadcopter components. The application of domain randomization shows great potential for synthetic training data sets with high detection rates (mAP@0.5 of 71,21%) utilizing a YOLOv7 detection model on real world images. The described process has potential for full automation, thus eliminating tedious manual work and reducing the time needed to develop AI-CV models in engineering.

Further research aims are the development of a data model to accompany the process and the integration of the framework into the entire product life cycle focusing on product development. Further methods and tools are investigated for the targeted control of domain randomization and the systematic investigation of the influence of the parameters of domain randomization on model generalization.

References

Ahmad, K., Abdelrazek, M., Arora, C., Bano, M. and Grundy, J. (2023), "Requirements engineering for artificial intelligence systems: A systematic mapping study", *Information and Software Technology*, Vol. 158, p. 107176. https://dx.doi.org/10.1016/j.infsof.2023.107176

Assadzadeh, A., Arashpour, M., Brilakis, I., Ngo, T. and Konstantinou, E. (2022), "Vision-based excavator pose estimation using synthetically generated datasets with domain randomization", *Automation in Construction*, Vol. 134, p. 104089. https://dx.doi.org/10.1016/j.autcon.2021.104089

Chen, Y., Li, W., Sakaridis, C., Dai, D. and van Gool, L. (2018), *Domain Adaptive Faster R-CNN for Object Detection in the Wild*. https://dx.doi.org/10.48550/arXiv.1803.03243

Dammann, M.P., Steger, W. and Stelzer, R. (2022), "Automated and Adaptive Geometry Preparation for AR/VR-Applications", *Journal of Computing and Information Science in Engineering*, Vol. 22 No. 3. https://dx.doi.org/10.1115/1.4053327

Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y. and Olefir, D., et al. (2019), *BlenderProc*. https://dx.doi.org/10.48550/arXiv.1911.01911

DIN Deutsches Institut für Normung e.V. (2020), *DIN SPEC 13266:2020-04, Leitfaden für die Entwicklung von Deep-Learning-Bilderkennungssystemen*, Beuth Verlag GmbH, Berlin. https://dx.doi.org/10.31030/3134557

Downs, L., Francis, A., Koenig, N., Kinman, B. and Hickman, R., et al. (2022), *Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items*. https://dx.doi.org/10.48550/arXiv.2204.11918

Drost, B., Ulrich, M., Bergmann, P., Hartinger, P. and Steger, C. (2017), "Introducing MVTec ITODD — A Dataset for 3D Object Recognition in Industry", paper presented at 2017 IEEE International Conference on Computer Vision Workshop (ICCVW), 22.10.-29.10.2017, Venice, Italy. https://dx.doi.org/10.1109/ICCVW.2017.257

Everingham, M., van Gool, L., Williams, C.K.I., Winn, J. and Zisserman, A. (2010), "The Pascal Visual Object Classes (VOC) Challenge", *International Journal of Computer Vision*, Vol. 88 No. 2, pp. 303–338. https://dx.doi.org/10.1007/s11263-009-0275-4

He, J., Yang, S., Yang, S., Kortylewski, A. and Yuan, X., et al. (2022), "PartImageNet: A Large, High-Quality Dataset of Parts", in Avidan, S., Brostow, G., Cissé, M., Farinella, G.M. and Hassner, T. (Eds.), *Computer Vision – ECCV* 2022, *Lecture notes in computer science*, Vol. 13668, Springer Nature Switzerland, Cham, pp. 128–145. https://dx.doi.org/10.48550/arXiv.2112.00933

He, K., Gkioxari, G., Dollár, P. and Girshick, R. (2017), *Mask R-CNN*. https://dx.doi.org/10.48550/arXiv.1703.06870

Hesenius, M., Schwenzfeier, N., Meyer, O., Koop, W. and Gruhn, V. (2019), "Towards a Software Engineering Process for Developing Data-Driven Applications", in *2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, 5/28/-5/28/2019, Montreal, QC, Canada*, IEEE, Piscataway, NJ, pp. 35–41. https://dx.doi.org/10.1109/RAISE.2019.00014

Horváth, D., Erdős, G., Istenes, Z., Horváth, T. and Földi, S. (2023), "Object Detection Using Sim2Real Domain Randomization for Robotic Applications", *IEEE Transactions on Robotics*, Vol. 39 No. 2, pp. 1225–1243. https://dx.doi.org/10.1109/tro.2022.3207619

https://doi.org/10.1017/pds.2024.226 Published online by Cambridge University Press

Hussain, M. (2023), "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection", *Machines*, Vol. 11 No. 7, p. 677. https://dx.doi.org/10.3390/machines11070677

Jain, A., Patel, H., Nagalapatti, L., Gupta, N. and Mehta, S., et al. (2020), "Overview and Importance of Data Quality for Machine Learning Tasks", in Gupta, R., Liu, Y., Shah, M., Rajan, S., Tang, J. and Prakash, B.A. (Eds.), *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020/07/06-10, USA*, ACM, New York, USA, pp. 3561–3562. https://dx.doi.org/10.1145/3394486.3406477

Jocher, G. and Waxmann, S. (2023), "Ultralytics YOLOv8 Docs. Datasets Overview", available at: https://docs.ultralytics.com/datasets/ (accessed 13 February 2024)

Kohtala, S. and Steinert, M. (2021), "Leveraging synthetic data from CAD models for training object detection models – a VR industry application case", *Procedia CIRP*, Vol. 100, pp. 714–719. https://dx.doi.org/10.1016/j.procir.2021.05.092

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L. and Girshick, R., et al. (2014), *Microsoft COCO: Common Objects in Context*. https://dx.doi.org/10.48550/arXiv.1405.0312

Mayer, N., Ilg, E., Fischer, P., Hazirbas, C. and Cremers, D., et al. (2018), "What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?", *International Journal of Computer Vision*, Vol. 126 No. 9, pp. 942–960. https://dx.doi.org/10.1007/s11263-018-1082-6

Microsoft (2023), "What is the Team Data Science Process?", available at: https://learn.microsoft.com/en-us/azure/architecture/data-science-process/overview (accessed 13 February 2024)

Ono, T., Suzuki, A. and Tamukoh, H. (2023), "An Effective Method for Minimizing Domain Gap in Sim2Real Object Recognition Using Domain Randomization", *Proceedings of International Conference on Artificial Life and Robotics*, Vol. 28, pp. 420–424. https://dx.doi.org/10.5954/icarob.2023.os17-6

Orhei, C., Vert, S., Mocofan, M. and Vasiu, R. (2021), "End-To-End Computer Vision Framework: An Open-Source Platform for Research and Education", *Sensors (Basel, Switzerland)*, Vol. 21 No. 11. https://dx.doi.org/10.3390/s21113691

Park, K., Lee, H., Yang, H. and Oh, S.-Y. (2020), "Improving Instance Segmentation using Synthetic Data with Artificial Distractors", in *ICCAS 2020: 2020 20th International Conference on Control, Automation and Systems proceedings October 13 (Tue)-16 (Fri), 2020, BEXCO, Busan, Korea, 10/13/2020 - 10/16/2020, Busan, Korea (South)*, IEEE, Piscataway, NJ, pp. 22–26. https://dx.doi.org/10.23919/ICCAS50221.2020.9268390

Pasanisi, D., Rota, E., Ermidoro, M. and Fasanotti, L. (2023), "On Domain Randomization for Object Detection in real industrial scenarios using Synthetic Images", *Procedia Computer Science*, Vol. 217, pp. 816–825. https://dx.doi.org/10.1016/j.procs.2022.12.278

Salas, A.J.C., Meza-Lovon, G., Fernandez, M.E.L. and Raposo, A. (2020), "Training with synthetic images for object detection and segmentation in real machinery images", in 2020 *33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 07.11.2020 - 10.11.2020, Recife/Porto de Galinhas, Brazil*, IEEE, pp. 226–233. https://dx.doi.org/10.1109/SIBGRAPI51738.2020.00038

Schulz, M., Neuhaus, U., Kaufmann, J., Kühnel, S. and Alekozai, E.M., et al. (2022), *DASC-PM v1.1: A process model for data science projects*, Nordakademie gAG Hochschule der Wirtschaft; Universitäts- und Landesbibliothek Sachsen-Anhalt, Elmshorn, Halle (Saale)

Schwoch, S., Leidich, J., Layer, M., Saske, B. and Paetzold-Byhain, K., et al. (2023), "A conceptual framework for information linkage and exchange throughout the lifecycle of process plants", in *DS 125: Proceedings of the 34th Symposium Design for X (DFX2023), 14th and 15th September* 2023, The Design Society, pp. 245–256. https://dx.doi.org/10.35199/dfx2023.25

Stelzer, R., Steger, W. and Petermann, D. (2012), "The VR Session Manager: A Tool to Co-Ordinate a Collaborative Product Development Process in a Virtual Environment", in *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - 2012, 8/12/2012 - 8/15/2012, Chicago, Illinois, USA*, ASME, New York, NY, pp. 1517–1525. https://dx.doi.org/10.1115/DETC2012-70998

Stephen Gould (2012), "DARWIN: A Framework for Machine Learning and Computer Vision Research and Development", *Journal of Machine Learning Research*, Vol. 13 No. 113, pp. 3533–3537

Tremblay, J., Prakash, A., Acuna, D., Brophy, M. and Jampani, V., et al. (2018), *Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization*. https://dx.doi.org/10.48550/arXiv.1804.06516

Valtchev, S.Z. and Wu, J. (2021), "Domain randomization for neural network classification", *Journal of big data*, Vol. 8 No. 1, p. 94. https://dx.doi.org/10.1186/s40537-021-00455-5

Wong, M.Z., Kunii, K., Baylis, M., Ong, W.H., Kroupa, P. and Koller, S. (2019), "Synthetic dataset generation for object-to-model deep learning in industrial applications", *PeerJ. Computer science*, Vol. 5, e222. https://dx.doi.org/10.7717/peerj-cs.222

Zhou, L., Zhang, L. and Konz, N. (2023), "Computer Vision Techniques in Manufacturing", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 53 No. 1, pp. 105–117. https://dx.doi.org/10.1109/TSMC.2022.3166397