Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 134 (2025) 259–264

www.elsevier.com/locate/procedia

Proceedings of the 58th CIRP Conference on Manufacturing Systems 2025

Enhancing Machine Vision Training with Digital Twins: A Toolchain for Optimized Image Categorization Using Synthetic Trainingsets

Bernhard Wallner[a,*], Mihnea Aleman[a], Fabian Singer[a], Laurenz Pickel[a], Jürgen Donabaum[a], Friedrich Bleicher[a], Thomas Trautner[a]

[a]*TU Wien, Institute of Production Engineering and Photonic Technologies, Getreidemarkt 9/311, 1060 Wien, Austria*

* Corresponding author. Tel.: +43-1-58801-31120 ; *E-mail address:* wallner@ift.at

Abstract

In dynamic production environments, frequent changes in products and setups demand efficient machine vision systems to verify part configurations accurately. This study introduces a digital twin-enabled toolchain designed to generate synthetic images for machine vision training, thereby reducing dependence on large, real-world image datasets. By employing pretrained deep learning models and an extractor-based classification method, the toolchain significantly minimizes the required number of training images without compromising accuracy. An industrial case study reveals the effectiveness of this approach, achieving reliable performance with fewer images and reducing training time, offering a cost-effective solution for adaptable, resilient manufacturing systems.

Keywords: digital twin; synthetic training images; machine vision; manufacturing cell; deep learning; case study

1. Introduction

Production companies face volatility through mass customization, fast market changes or global disturbances leading to more frequently changing production and smaller lot sizes. Those changes often require adapting the parts' machining setup and clamping situation within a CAM-System. Workers usually clamp the parts, making the process vulnerable to human error. Wrong clamping setup can lead to scrap production, tool breakage or machine crashes, production interruptions, and additional costs. Machine vision applications are used to check if parts are correctly clamped with the suitable clamping device to counter manual errors in the clamping process.

Machine vision applications must be trained with image data, requiring training and test sets of the actual application. This procedure is cost-intensive because (i) images often have to be taken in the production environment, leading to downtimes, and (ii) it has to be performed by machine vision experts, making it not feasible for small lot sizes.

Digital Twins (DTs) are a promising concept for tackling those issues as they offer a virtual representation of the production environment, allowing simulation, visualization, and tests without disrupting production. We propose a DT toolchain for training machine vision applications with synthetical images by extending the framework of Alexopoulos et al. [1]. Our attempt reduces the requirements for synthetical images by using open-source pretrained deep-learning models. We compare eight different algorithms and benchmark them against a commercial solution within an industrial case study.

This DT approach enables the automation of the toolchain from CAM planning to machine vision, which (i) minimizes the risk of human errors and (ii) makes small lot sizes feasible through cost reduction. Therefore, we can increase the resilience of the manufacturing system.

This paper is structured as follows: In section 2, we briefly overview related work. In section 3, we propose the CAM2Vision toolchain and elaborate on the critical machine learning functionality within a case study in section 4. Then, we discuss and summarise our findings in section 5.

2. Related work

2.1. Digital Twin frameworks and standardization

DTs propose seamless integration between the physical and virtual worlds. Besides the consensus in academia that a DT consists of a digital and a physical representation of an artefact,

10.1016/j.procir.2025.03.031

many different concepts, standards, and frameworks exist. The majority follows either the 3-dimension model developed by Grieves [2] and characterized by Kritzinger et al. [3] or the extended 5-dimension model by Tao et al. [4]. As summarized by Latsou et al. [5] "..., a detailed methodology and standardization are needed to address the absence of a common framework, facilitating the creation of DTs ...". Their proposed framework focuses on reusability and scalability, e.g. by using ontologies.

One standard is ISO 23247 [6], offering distinct functional elements. Wallner et al. [7] applied this standard in a manufacturing cell and introduced a feature set categorization by rate-of-changes. In our approach, we focus on tackling the medium-frequency changes, characterized by changes "...that occur during the production of the default portfolio ..." [7]. Although the ISO 23247 still lacks essential features, e.g. verification, validation [5], or proper life-cycle coverage [7], it offers essential modularity within the framework to build DTs in the manufacturing domain.

Alexopoulos et al. [1] propose a DT-framework to train machine learning applications on synthetical images. They highlight the advantages and needs of automatic image generation and labeling and give an insight into the challenges of creating photo-realistic renderings. Their approach requires a high setup time for the rendering environment (1 hour), medium-sized data sets (300 images) and expert knowledge for image generation. Although we follow the approach basically, we try reducing the requirements for the test image generation by using basic settings and CAM-inbuilt rendering options. Also, we extend the approach by proposing an ontology-based reasoning for the image evaluation.

The importance of DTs for resilience is shown by Bakopoulos et al. [8]. They propose a resilience manufacturing framework using Asset Administration Shell (AAS), data spaces, resilience assessment, and reconfiguration services.

2.2. Machine Learning for Machine Vision

Although there are approaches that focus on part localization for robot applications [9, 10, 1] or defect detection for quality control [11, 12] classification of real-world images with artificial counterparts lacks discussion. Approaches like metric learning with Siamese networks [13] optimize image similarity, while CycleGANs [14] improve synthetic image realism to mitigate domain shifts. Both methods facilitate classification but require large datasets for optimal performance.

Transfer learning [15] presents a more data-efficient option by fine-tuning models pretrained on large datasets, also highlighted by [10]. This method leverages existing representations to boost classification accuracy in real-world contexts without extensive retraining. These pretrained models can extract features from images, which can subsequently be classified using a variety of machine-learning algorithms. To leverage this capability, we implement a two-part architecture consisting of a feature extractor and a classifier. The feature extractor utilizes deep learning models pretrained on large datasets, while the classifier applies machine learning methods to process the extracted features. We will evaluate these deep learning models as a baseline compared to traditional computer vision techniques.

3. The CAM2Vision Toolchain

3.1. Digital Twin Enabled Decision Making

As proposed by the related work, it is beneficial to modularise DTs and develop individual functions. Similar to Alexopoulos et al. [1], we propose an attempt that uses synthetically generated images to train the machine vision application. Figure 1 shows our proposed toolchain: In the CAM-System, the setup situation is defined. The main parameters are the required clamping device (workpiece pallet), raw material dimensions, and material position. The image evaluator later uses those parameters for decision-making. The CAM-System also defines the camera settings for the creation of synthetic images. Those images then get passed together with the parameters to the machine learning pipeline, which will be described in subsection 3.3. In the setup station, the orders are prepared and set up according to the information from the CAM-System. After the prepared workpiece is released to the manufacturing cell, an image gets taken and provided to the image-evaluator.

Using machine learning and an underlying ontology, the image evaluator determines the appropriate next steps for processing the clamped workpiece. We define three scenarios: (i) *Reclamp:* the deviations are too big, and the setup process needs to be repeated. Reclamping is necessary when the wrong clamping device is used or the raw material is mounted outside tolerances and can't be machined. (ii) *Replan:* the raw material is clamped slightly outside the tolerances, and the CAM-System needs to verify the NC-Code (iii) *Offset:* the raw material is clamped within the tolerances, and the offset (e.g. G54) is passed to the CNC-Machine, and the production starts. In this publication, we focus on the *Reclamp*-functionality by detecting three different clamping situations.

3.2. Synthetic Image Creation

The synthetic image creation aims to create proper input for the training algorithm without needing the physical setup. Many CAM-Software offer inbuilt rendering solutions covering part material, light sources, perspective, and background. The primary aim of those rendering tools is to create photo-realistic pictures for marketing purposes, requiring high setup time and tweaking for proper settings, e.g. material or lighting. Our approach is to use basic settings as much as possible, reducing setup effort and rendering time.

A crucial step is automating this image rendering pipeline. In the end, the camera settings of the actual camera should match the settings used in the rendering process. One way to achieve this is by using a camera that offers an AAS and using an interface, e.g. REST API, to change the settings inside the Software accordingly.

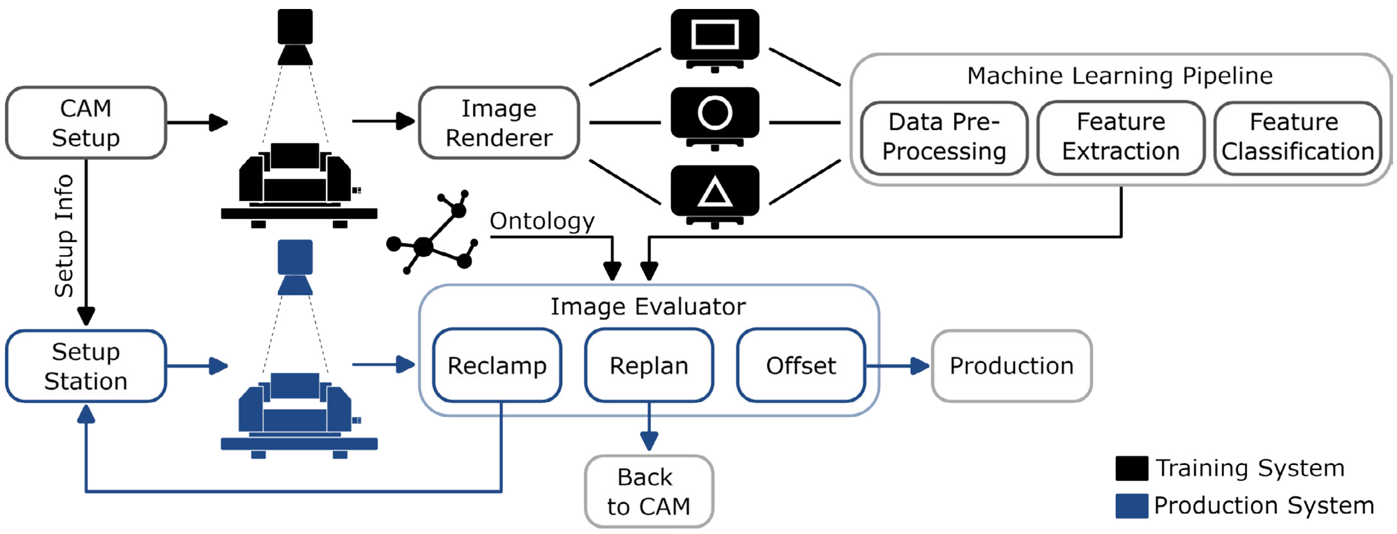


Fig. 1: Architecture of CAM2Vision Toolchain

3.3. Machine Learning Pipeline

This section presents our machine learning pipeline, which involves a series of key steps: preprocessing, feature extraction, and feature classification. The complete architecture of this approach is depicted in Figure 1. The source code of the machine learning pipeline is available on GitHub [23].

The preprocessing stage focuses on preparing images to enhance the effectiveness of subsequent feature extraction algorithms. This process includes resizing to a uniform size, normalizing, and scaling the images. These steps ensure consistent and optimally formatted input data.

We employ various methods for feature extraction as seen in Table 1. When using SIFT, we apply the Bag of Visual Words model, which is particularly effective for representing local image features. The Color Histogram approach, on the other hand, focuses on capturing the distribution of colors within an image. It involves computing a histogram representing how pixel values are distributed across different color ranges in the RGB space. For the deep learning models, including SqueezeNet1_1, Inceptionv3, DenseNet161, ResNet152, ResNext101_64x4d, and ViT-H/14, we leverage pretrained weights from the ImageNet [24] dataset. These pretrained models allow us to extract high-level features directly from the images. The features extracted from these models capture complex patterns and semantic information, making them highly effective for image analysis. Finally, the extracted features are utilized by a feature classifier to categorize the images. We have selected a Support Vector Machine (SVM) classifier for this task. SVMs are well-suited for image classification because they handle high-dimensional feature spaces, which is common when working with complex image features.

Table 1: Comparison of model characteristics and performance metrics.

Model	Memory (MB)	Feature Count	Accuracy	Precision	Recall	F1 Score
vit_h_14 [16]	**2416.74**	1280	0.95	0.96	0.95	0.95
resnext101_64x4d [17]	319.6	2048	**1.0**	**1.0**	**1.0**	**1.0**
resnet152 [18]	230.7	2048	**1.0**	**1.0**	**1.0**	**1.0**
densenet161 [19]	110.7	**2208**	**1.0**	**1.0**	**1.0**	**1.0**
inceptionv3 [20]	104.12	2048	**1.0**	**1.0**	**1.0**	**1.0**
squeezenet1_1 [21]	4.78	512	**1.0**	**1.0**	**1.0**	**1.0**
colorhistogram	0.119	768	**0.33**	**0.11**	**0.33**	**0.17**
sift [22]	**0.066**	**100**	0.68	0.66	0.68	0.65

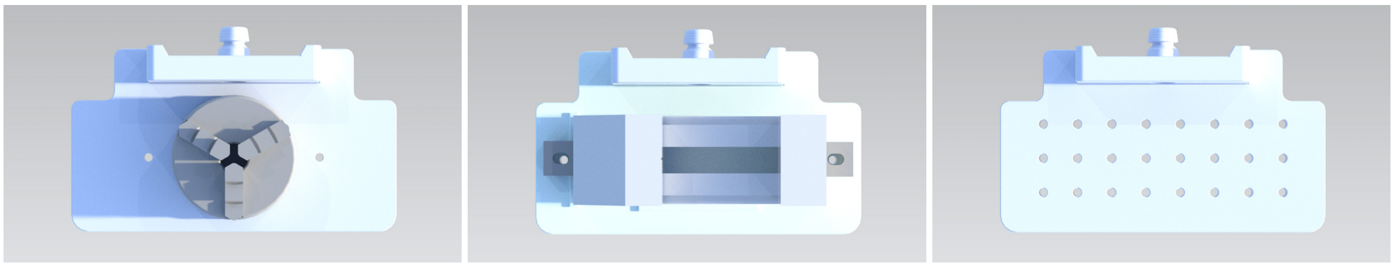
3.4. Evaluation Metrics

We evaluate our approach through a comprehensive set of metrics that capture various aspects. Our evaluation is organized into three essential parts: (i) performance, (ii) memory efficiency and inference speed, and (iii) quality of the features extracted.

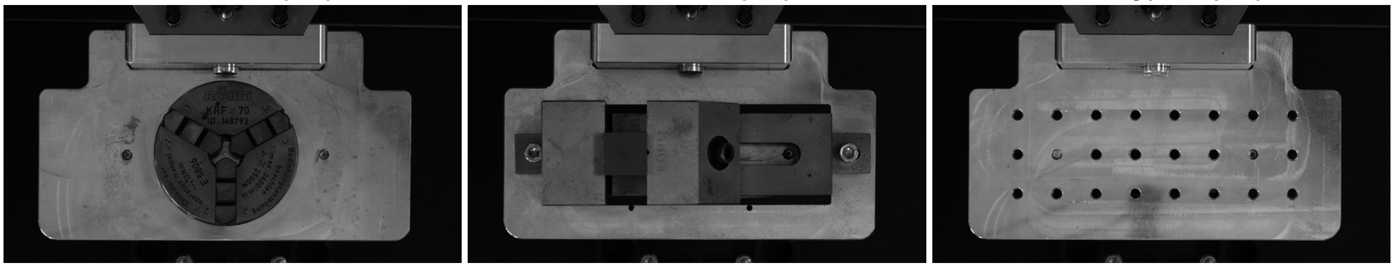
The selected *performance metrics* include accuracy, precision, recall, and F1-score. Accuracy gauges the overall correctness of the model, while precision reflects the proportion of true positives among predicted positives, helping to reduce false positives. Recall measures the model's capability to identify all relevant instances. The F1-score, as a harmonic mean of precision and recall, provides a balanced measure of the model's performance, mainly when dealing with imbalanced datasets.

We also assess *memory efficiency and inference speed*. Inference speed is essential for real-time applications, indicating how quickly the model can predict. Memory efficiency measures the memory required during inference, which is crucial for deployment in resource-constrained environments. Precisely, we measure the time it takes for the model to predict a single image and the memory space the algorithm occupies.

Finally, we assess the *quality of the extracted features* by examining the formation of clusters in 2D space. Well-defined, distinct clusters corresponding to different classes indicate that the feature extractor effectively captures discriminative features, while overlapping clusters may suggest lower feature quality or insufficient class separation.



(a) Cat. chuck *training image* (b) Cat. vice *training image* (c) Cat. empty *training image*



(d) Cat. chuck *test image* (e) Cat. vice *test image* (f) Cat. empty *test image*

Fig. 2: Image categories for training (a-c) and testing (d-f)

4. Case Study

4.1. Cell setup

The manufacturing cell in focus follows the same setup as already explained by [7]. In addition to the EMCO CM55 machine tool and the ABB IRB 120 robot, we integrated a camera system as described in subsection 4.2. The case study focuses on categorizing the clamping devices for supporting the *Reclamp*-functionality of the toolchain. As CAM-Software, we use SIEMENS NX in version 1969. Three different clamping pallets are available in the case study as shown in Figure 2: *Cat. chuck* for cylindrical raw material, *Cat. vice* for prismatic raw material, and *Cat. empty* pallet for individual clamping setup.

4.2. Optical Setup

This experimental setup evaluates a deep learning model trained solely on SIEMENS NX CAM renderings. The objective is to validate whether this model can accurately recognize actual images of the same parts captured with the setup described below. The camera is mounted using profile technology, positioned approximately 1 meter above the workspace and oriented vertically downward to capture a clear, top-down view of each clamping device. Ambient ceiling lighting is used throughout the setup, increasing exposure times to compensate for lower light levels. In real-world applications, additional lighting is beneficial to reduce exposure time and prevent motion blur, but this setup achieved sufficient quality for the controlled experiments.

We use a 5-megapixel Opto Engineering COE-050-C-POE-050-IR-C color camera (Sony IMX264 sensor, 2448 x 2048 resolution, 2/3" format) configured for monochrome imaging in our setup. It features a global shutter for sharp captures without motion blur and achieves a frame rate of 23.5 fps. The camera uses an IR cut filter, ensuring image quality under ambient light. The 25 mm fixed-focus lens (Opto Engineering EN2MP2514) offers a sharp image at the required working distance. The aperture, adjustable from f/1.4, allows for depth-of-field control on a 2/3" sensor with minimal distortion (0.27%) and supports a large image circle of 11 mm.

4.3. Training and Test Sets

The training set was intentionally kept simple, with basic materials used for rendering and relying on built-in functionality. We aimed to match the virtual camera settings to the physical camera, though not all camera parameters are available in the CAM-Software. The images were rendered manually, each taking approximately 1-2 minutes on a standard laptop. We focused on a small training set of 21 images to keep the process efficient. The test set consists of 375 grayscale images to evaluate the model's performance. Figure 2 shows the visual differences between the training set (2a-2c) and the test set (2d-2f).

4.4. Machine Learning Method

Memory Usage and Efficiency: When comparing machine learning models, significant disparities in memory consumption and feature extraction capabilities emerge, reflecting their architectural complexities and intended applications. Regarding memory consumption, SIFT and Color Histogram are the most lightweight models, making them efficient for low-resource scenarios. SqueezeNet1_1 also demonstrates minimal memory overhead, while more sophisticated models, such as ResNet50, InceptionV3, and DenseNet161, require significantly more memory but remain manageable. Conversely, models like ResNeXt101_64x4d and ViT-H/14 necessitate much larger memory allocations, with the latter being particularly demanding. Table 1 shows the memory consumption of each model.

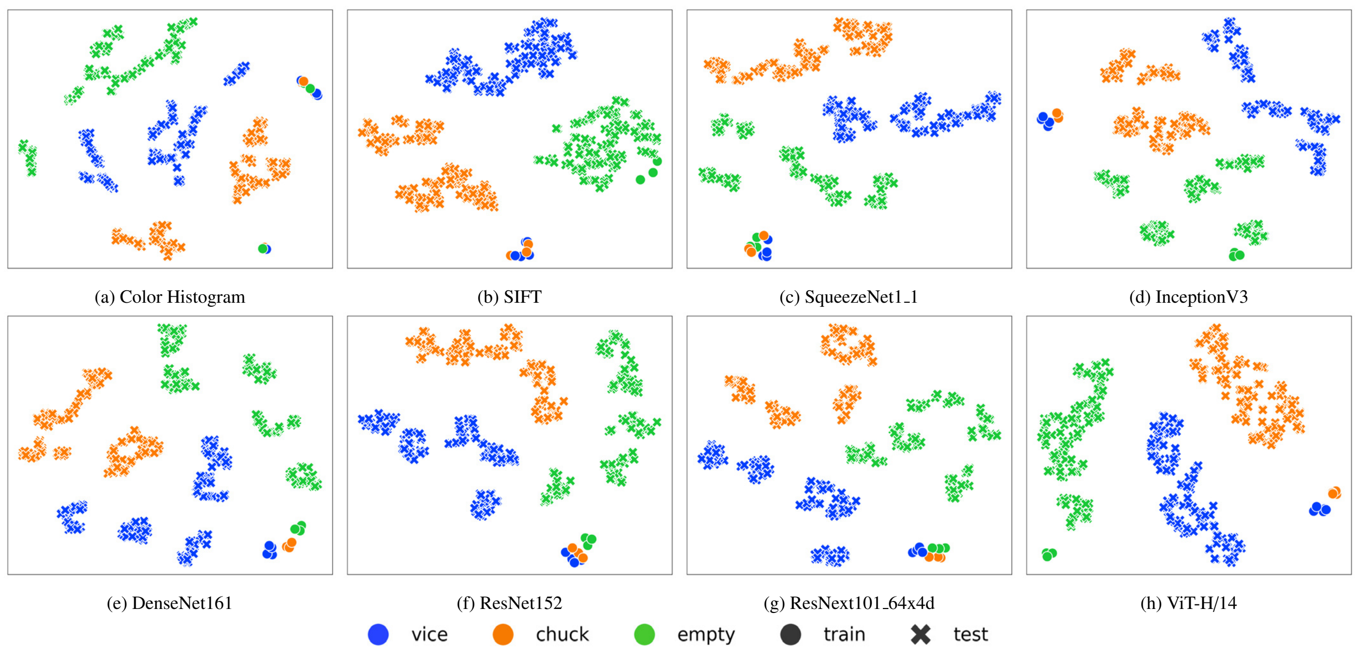


Fig. 3: t-SNE visualisation of features extracted by different models.

Feature extraction capabilities also vary considerably among these models. DenseNet161, ResNet152, ResNeXt101_64x4d, and InceptionV3 showcase abilities to capture intricate data patterns. In contrast, SqueezeNet1_1 is suitable for resource-constrained environments, while traditional methods like SIFT produce only 100 features, potentially limiting their performance in complex tasks. The transformer-based ViT-H/14 outputs 1280 features, balancing complexity and efficiency. This comparison shows the inherent trade-offs between feature richness and memory efficiency. While lightweight models are more memory-efficient, they may compromise predictive capabilities relative to their larger, more complex counterparts.

Performance: Table 1 illustrates the performance comparison of the machine learning models, revealing significant disparities in their effectiveness as indicated by accuracy, precision, recall, and F1 score. Models like ResNet152, ResNeXt101_64x4d, DenseNet161, InceptionV3, and SqueezeNet1_1 all achieve perfect scores across all metrics, showcasing their robust capabilities in accurately identifying and classifying data. In contrast, the Color Histogram model shows much poorer performance, with an accuracy of only 0.33 and a particularly low precision of 0.11, indicating limited effectiveness in classification tasks. The SIFT model performs moderately well, with an accuracy of 0.68 and balanced precision and recall scores. In contrast, the ViT-H/14 model exhibits high performance with an accuracy of 0.95 and excellent precision and recall (0.96 and 0.95). Overall, the results highlight that while deep learning models excel in performance metrics, traditional methods like Color Histogram and SIFT lag significantly behind, emphasizing the advantages of advanced architectures in machine learning applications.

Quality of the extracted features: Figure 3 presents visual representations of the features extracted by each model we tested, using t-SNE [25]. T-SNE is a technique that projects data into a lower-dimensional space, typically for visualization. In each figure, circles represent synthetical images from the training dataset, while “x” symbols denote real-world images. Additionally, when using t-SNE for dimensionality reduction, some information from the original high-dimensional data is inevitably lost, as t-SNE cannot perfectly preserve all relationships from the original space when projecting to fewer dimensions. As a result, while t-SNE can highlight patterns and clusters for easier visual interpretation, it may not fully capture the complexity of the original data, leading to a potential loss of certain nuances in the data structure. The initial observation is that each model effectively clustered the test set, demonstrating a clear visual separation. However, of greater significance is the observation that the models faced greater difficulty in clustering artificial images alongside real-world images.

While deep learning models can extract significant features and prioritize the most relevant ones, it is crucial to emphasize preprocessing when training with synthetical images. We can minimize visual discrepancies by ensuring these images closely mimic real-world counterparts through effective preprocessing, ultimately enhancing model performance during testing. The analysis of the color histogram is shown in Figure 3a, where it struggles to differentiate between the training categories. SIFT’s effectiveness in aligning images from *Cat. empty*, though limited in distinguishing between similar types like *Cat. vice* and *Cat. chuck*, is illustrated in Figure 3b. SqueezeNet1_1 lacks clear visual separation in t-SNE plots, as seen in Figure 3c. More complex models like InceptionV3, DenseNet161, ResNet152, and ResNext101_64x4d demonstrate improved separation of artificial image categories and better alignment with real-world data, shown in Figures 3d-3g, respectively. The ViT-H/14 model best separates categories and closely aligns artificial images with real test images, as illustrated in Figure 3h.

4.5. Commercial Benchmark

We used the proprietary image processing library Zebra Aurora Vision to determine whether the results achieved in subsection 4.4 could also be obtained with a commercial software solution. The same training and test datasets were employed. Similar to the presented results of the open-source solution, an extractor classification approach was utilized. This two-step process consists of a *Detect Features* filter and a *Classify Object* filter. The *Detect Features* filter is used to locate the relevant clamping pallets in the image, segment the corresponding regions, and define these as Regions of Interest for the *Classify Object* filter. This process significantly improved the model's reliability. All images in the test set were correctly classified, albeit with varying confidence levels. On average, correct classification was achieved with confidence of 99.9% for *Cat. chuck*, 62.5% for *Cat. vice*, and 100% for *empty*. The low confidence for *Cat. vice* is due to high variety of clamping positions in the test set and the simple training set.

5. Discussion and Conclusions

This paper proposes a DT-toolchain for generating synthetical images and training machine vision applications based on a CAM-Setup. It could be shown that using pretrained deep learning models provides robust performance across various architectures, making this a reliable and practical approach. It shows that solving this problem is relatively general and flexible, as it doesn't require the development of a highly specialized or overly complex model. Instead, even standard pretrained models can achieve substantial results, indicating that the problem can be addressed without custom-built, intricate solutions. Since most models showed good accuracy, precision, and recall, we favor SqueezeNet1_1 due to its high memory efficiency. Furthermore, we showed that basic rendering functionality and small training sets also allow suitable model training.

Our future research will focus on the automation of the remaining toolchain by integrating the Siemens NXOpen interface for renderings, AAS for camera configuration and extending the *image evaluator* to address the *Replan* and *Offset* functionality.

Acknowledgements

This research has been partially funded by the Horizon Europe FLEX4RES project with Grant Agreement Number 101091903 and by the project fund "Arbeit 4.0" of AK Niederösterreich within the MillOne project.

References

[1] K. Alexopoulos, N. Nikolakis, G. Chryssolouris, Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing, International Journal of Computer Integrated Manufacturing 33 (5 2020). doi:10.1080/0951192X.2020.1747642.

[2] M. Grieves, Digital twin: Manufacturing excellence through virtual factory replication (2014).

[3] W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn, Digital Twin in manufacturing: A categorical literature review and classification, IFAC-PapersOnLine 51 (11) (2018). doi:10.1016/j.ifacol.2018.08.474.

[4] F. Tao, H. Zhang, A. Liu, A. Y. Nee, Digital twin in industry: State-of-the-art, IEEE Transactions on Industrial Informatics 15 (2019) 2405–2415. doi:10.1109/TII.2018.2873186.

[5] C. Latsou, D. Ariansyah, L. Salome, J. A. Erkoyuncu, J. Sibson, J. Dunville, A unified framework for digital twin development in manufacturing, Advanced Engineering Informatics 62 (2024) 102567. doi:10.1016/J.AEI.2024.102567.

[6] ISO 23247-1 Automation systems and integration-Digital twin framework for manufacturing-Part 1: Overview and general principles (2021).

[7] B. Wallner, B. Zwölfer, T. Trautner, F. Bleicher, Digital twin development and operation of a flexible manufacturing cell using iso 23247, Procedia CIRP 120 (2023) 1149–1154. doi:10.1016/J.PROCIR.2023.09.140.

[8] E. Bakopoulos, K. Sipsas, N. Nikolakis, K. Alexopoulos, A digital twin and data spaces framework towards resilient manufacturing value chains, IFAC-PapersOnLine (2024). doi:10.1016/J.IFACOL.2024.09.129.

[9] M. Choi, D. Kim, J. Moon, M. Kim, J. Um, Digital Twin System Framework for Gripping Deformable Objects with Synthetic Data and Reinforcement Learning, Springer Nature Switzerland, 2024. doi:10.1007/978-3-031-74482-2_12.

[10] C. Manettas, N. Nikolakis, K. Alexopoulos, Synthetic datasets for Deep Learning in computer-vision assisted tasks in manufacturing, Procedia CIRP 103 (2021) 237–242. doi:10.1016/j.procir.2021.10.038.

[11] C. Schorr, S. Hocke, T. Masiak, P. Trampert, A Scalable Synthetic Data Creation Pipeline for AI-Based Automated Optical Quality Control, 2024, pp. 37–46. doi:10.5220/0012717400003758.

[12] S. Mohanty, E. Su, C.-C. Ho, Enhancing titanium spacer defect detection through reinforcement learning-optimized digital twin and synthetic data generation, Journal of Electronic Imaging 33 (2024). doi:10.1117/1.JEI.33.1.013021.

[13] G. Koch, R. Zemel, R. Salakhutdinov, et al., Siamese neural networks for one-shot image recognition, in: ICML deep learning workshop, Vol. 2, Lille, 2015, pp. 1–30.

[14] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2223–2232.

[15] J. Plested, T. Gedeon, Deep transfer learning for image classification: a survey, arXiv preprint arXiv:2205.09904 (2022).

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale (2021). arXiv:2010.11929.

[17] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition, 2017. doi:10.1109/CVPR.2017.634.

[18] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[19] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, IEEE Conference on Computer Vision and Pattern Recognition (2016). doi:10.1109/CVPR.2016.308.

[21] F. N. Iandola, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size, arXiv preprint arXiv:1602.07360 (2016).

[22] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2004) 91–110.

[23] A. Mihnea, Feature extraction and classification, https://github.com/SpeedyGonzales949/feature-extraction-classification.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[25] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (11) (2008).