1828
AKADÉMIAI KIADÓ
UNIVERSITY of DEBRECEN
International Review of Applied Sciences and Engineering
DOI:
10.1556/1848.2025.01005

ORIGINAL RESEARCH PAPER

Application of image translation-based approach in industrial manufacturing system for object identification

Tibor Péter Kapusi[1], Timotei István Erdei[2*], Géza Husi[2] and András Hajdu[1]

[1] Department of Data Science and Visualization, Faculty of Informatics, University of Debrecen, Hungary

[2] Department of Vehicles Engineering, Faculty of Engineering, University of Debrecen, Hungary

Received: February 3, 2025 • Accepted: May 10, 2025

ABSTRACT

Older multi-axis industrial robots used in industry do not have the computational power to use neural networks to perform complex manufacturing tasks. Therefore, a method combining 3D CAD modeling with data synthesis has been developed. The pix2pixHD approach allows the synthesis of photo-realistic images to generate datasets. The resulting datasets can be used to train deep learning-based detectors, which can later be applied to improve the object detection accuracy of older machine units. After evaluation of tests with YOLOv3 models, it is shown that only the datasets generated using the synthesized approach can detect an object with high detection accuracy. The generated datasets were also measured using the Complex wavelet structural similarity. The developed innovative method can be a practical and cost-effective strategy for smaller laboratories and small and medium enterprises, thus providing an opportunity to upgrade older machines.

KEYWORDS

cyber-system, image-to-image, conveyor, deep learning, neural network, industry 4.0, object detection

1. INTRODUCTION

Industry 5.0 opens new opportunities for industrial automation. It presents a new challenge for engineers, requiring systemic thinking. During production, a production line generates data that needs to be analyzed. Large amounts of data can be interpreted and evaluated using artificial intelligence. The problem is that there are many specific artificial intelligence models. Therefore, selecting and training the correct model with the proper data set is a challenging task.

In industry, many old machine units are used whose functions can be extended by modernization. However, these older machines do not have modern standards, so they cannot communicate over a network [1].

Real-time trajectory generation and collision avoidance are extremely complex and multi-factorial [2–4]. The usage of deep neural networks proves helpful for classifying key points. Upgrading outdated systems is challenging because the physical hardware cannot be upgraded.

Neural networks have been used successfully in 2D robot path planning [2, 3]. Work-pieces in the machine's work area were also identified, as well as critical points [4].

Redesigning robotic systems is challenging because embedded controllers have limited configuration options. For this reason, image analysis cannot be performed only via the robot controller.

*Corresponding author.
E-mail: timoteierdei@eng.unideb.com

AKJournals

Innovative solutions can overcome this problem and extend the machines' life cycles. The paper shows how deep learning can be added to older robots and how to train models.

In the project, we apply a DL-based (Deep Learning) technique without the need for major overhauls and implement artificial intelligence to detect the cube on the conveyor belt. Previously, we also discussed the possibilities for extending the functionality of old robots [5]. Now, in this case, the whole environment, including reflections of illumination and color, will also be examined during image synthesis.

The structure of the paper: Section 2 will describe the basics and parameters of the material handling machine. Section 3 provides the typical phenomenal problems of the real-world image scenes taken.

Section 4 presents the details of the modeling procedure. Section 5 provides the idea behind generating a dataset with Image-to-Image translation and describes the selected architecture.

Section six explains the details of chosen detector algorithms and section 7 summarizes the environment, along with the changing parameters in the training process. In section 8, we evaluated the effectiveness of deep learning-based algorithms independent of image quality, and Section 9 draws the conclusions of the paper.

2. KUKA KR5 ROBOT UNIT & FLEXLINK XK

In our CPS laboratory, a robotic cell is being set up to carry out various research and development tasks. The developments will focus on the application of robotics and artificial intelligence within the lab. A KUKA KR5 robot with serial kinematics has been installed in the lab. The robot itself has six degrees of freedom, which allows for a relatively large working envelope of 8.4 m^3. In industry, the machine unit is mainly used for welding tasks thanks to its high precision, with a repeatability of ± 0.04 mm. For material handling tasks, it is only suitable with restrictions as the payload is limited to 5 kg. The robot cell around the KUKA KR5 (see Fig. 1) is equipped with three platforms for material handling tasks. The robot cell is made of standard aluminum profiles. The main unit in the machine's work cell is the conveyor belt, which ensures the positioning of the various test objects.

The KUKA KR5, as mentioned above, is a serial kinematics unit with 6 DoF. It can handle complicated operations. However, the range of motion is greatly affected by the limitations of each axis:

- Rotational A1: ±155°
- Rotational A2: +65°/−180°
- Rotational A3: +158°/−15°
- Rotational A4: ±350°
- Rotational A5: ±130°
- Rotational A6: ±350° [6].

Workspace of KUKA KR5 robot arm can be seen in Fig. 2.

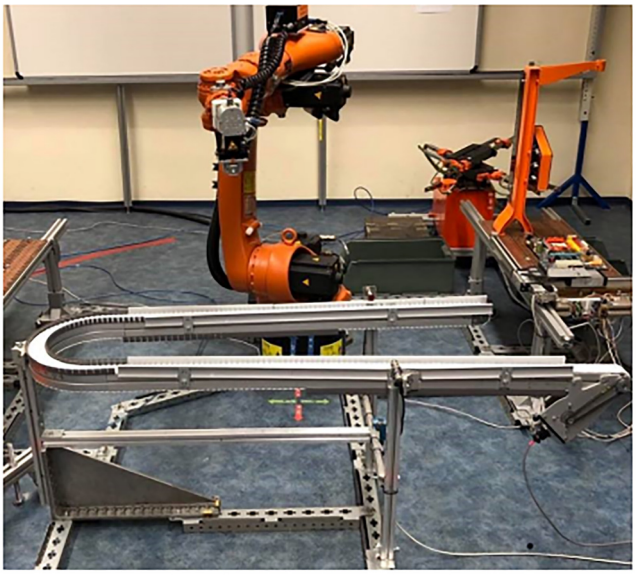

Fig. 1. KUKA KR5 Arc 'Own source'

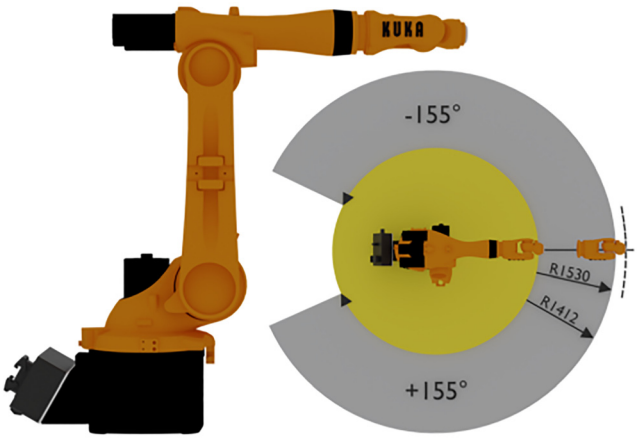

Fig. 2. The KUKA KR5 workspace 'Own source'

The total weight is 127 kg, and the payload is only 5 kg. The load capacity of the KUKA KR5 affects its dynamics, which in turn influences the robot's movement performance (see Fig. 3).

Figure 4 shows the maximum allowable load. In general, a load of 5 kg is allowed from 0 to 100 mm, but above 500 mm only 2 kg is allowed.

The dynamic parameters of the robot can also change depending on the load, which must be considered when performing a given movement. In our case, the rated payload is the maximum allowable mass that the machine unit can still handle under normal conditions.

The values show that the lower axes are significantly slower, and the reason for this is the robot's design because motors on these axes have to handle higher inertia. The opposite is also true for the upper joints of the robot, which are able to move more accurately and faster due to the lower load to be moved.

Figure 4 shows the speed profile of the KUKA KR5 robot arm axes as a function of time. The point P1 represents the

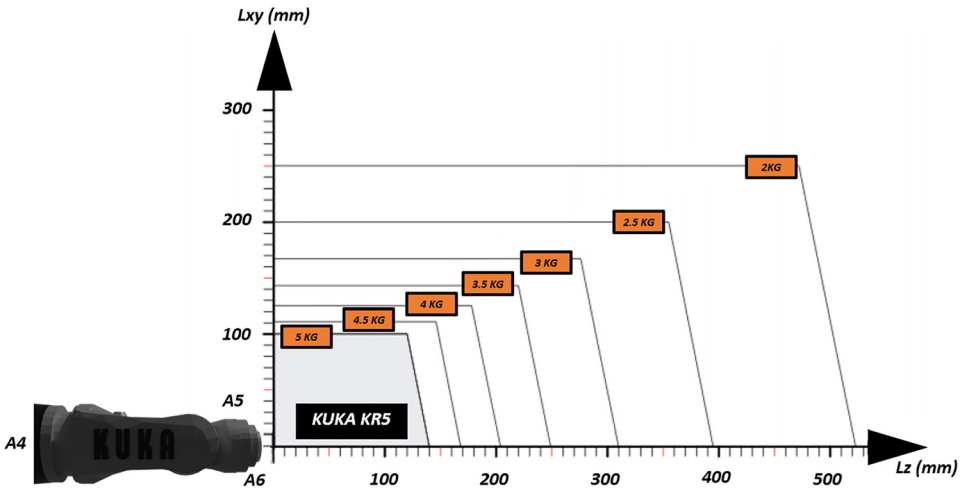

Fig. 3. The KUKA KR5 payload diagram 'Own source'

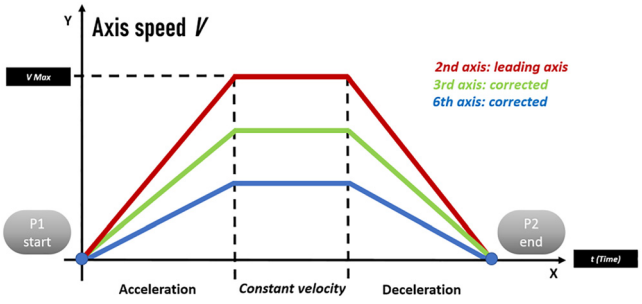

Fig. 4. The axis speed diagram of KUKA KR5 'Own source'

starting position and P2 is the target position in the robot workspace.

The dynamics of the robot's movement can be broken down into three phases. The first one is the acceleration phase, where the robot's speed increases gradually until it reaches the allowed max velocity. The slope on the graph characterizes the rate of acceleration. The constant velocity phase means that the machine unit has reached its allowed speed, i.e. the rate of acceleration is zero. In the braking phase, the machine speed gradually decreases until it reaches zero.

The graph shows that the A2 axis is the leading axis, which moves the fastest. The A3 and A6 axes accelerate or decelerate according to their movement in relation to the leading axis.

The Denavit-Hartenberg method was used for the kinematic description of KUKA KR5 robot arm. The method describes the positioning of links and joints in space in a standardized way (see Fig. 5 and Table 1).

The DH parameters are:

- θ (theta): Rotation around the z-axis to parallel the x-axis.
- d: Offset along the z-axis to the intersection of the common normal of the x-axes.
- a: Offset along the x-axis to the intersection of the common normal of the z-axes.
- α (alfa): Rotation about the x-axis to parallel the z-axes [7].

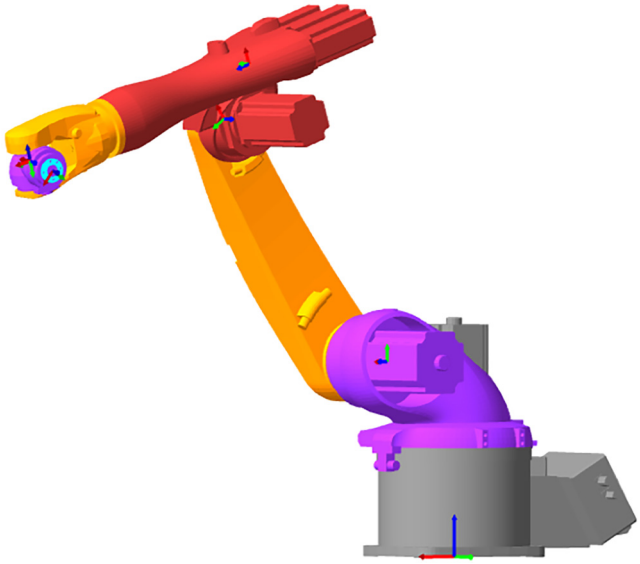

Fig. 5. The KUKA KR5 payload diagram 'Own source'

To determine the required Denavit-Hartenberg parameters [7], we used the free robot simulation software called RoboAnalyzer, which provides 3D graphics to determine the joint parameters of the KUKA KR5 robot arm. It is important to point out that the program is primarily designed for older Windows operating systems that can run well (.Net 2.0 framework). In our case, it was run on a virtual operating system, so system requirements were met.

The industrial assembly line within the robotics laboratory was used for the project. The specified objects are moving on Flexlink conveyor which has been installed into the workspace of the robot. A robot cell with a Flexlink conveyor belt was constructed out of aluminum profiles around the robot arm.

The conveyor can carry a weight of 10 kg and is powered by a NEMA23 ST5918L4508-B stepper motor. FlexLink is 5200 mm long and 45 mm wide. The KUKA KR5 arm unit has a Gmbh gripper to be able to transport objects if required.

Table 1. Denavit-Hartenberg parameters of KUKA KR5

Joint number	Axis type	Angle of the joint (θ)	Offset in the joint (b)	Length of the link (a)	Twist angle (α) radian	Default value (JV) degree
A1	Rotary joints	Variable	0.5	0.17	π/2	10
A2	Rotary joints	Variable	0.136	0.5	π	56
A3	Rotary joints	Variable	0.136	0.16	–π/2	−16
A4	Rotary joints	Variable	0.63	0.1	π/2	85
A5	Rotary joints	Variable	0.12	0.1	–π/2	75
A6	Rotary joints	Variable	0.115	0.1	0	42

3. BACKGROUND ON THE DESCRIPTIVENESS OF REAL-WORLD IMAGES

This section gives an overview of the description of real-world images, future applications for data synthesis, and deep learning frameworks, with the aim of producing as realistic images as possible. With the developed methodology, we will use computer-synthesized image files to create dataset, which will be used to ML-based (machine learning) models. The synthesized images will be rendered based on a 3D CAD model of your real object, which will include RGB colors. Reflections and possible color changes were also taken into account, and other light effects are among the specific phenomena that can be seen in the real world.

Building a physical-mathematical model for each system is quite challenging in the latter scenario [8, 9]. Although many manufacturing processes can be optimized from an illumination perspective [10], another paper written by Martinez noted that optical sensors may be limited in the component to be produced [11]. In order to investigate and offer a solid solution to the issues that arise, it is important to describe the phenomena found in the actual image. The actual image (see Fig. 6) is described by the following equation [12]:

$$I_{real} = I_{normal} + I_{phenonema} + N(0, \sigma_2) \quad (1)$$

where I_{normal} denotes the phenomenon-free image component, $I_{phenonema}$ contains only the phenomenon components, and N means a random Gaussian noise whose variance is sampled from a scaled chi-square distribution as $\sigma^2 \sim 0.01\chi^2$.

The components that contain individual phenomenon can be divided into two fundamental categories: flares and reflections. Reflections on the camera lenses are created by extreme lighting conditions. The architecture of the lens system largely dictates their shape and frequency. Depending on their type, they can be either diffused or reflective. This effect can interfere with object detection in industrial processes.

When this happens, pixels burn out, significantly affecting the algorithm's performance and reducing detection accuracy [13].

Reflections can be caused by the material and surface of the object being produced. According to formal definitions, reflections are the sum of spectral energy distributions of illuminations and surface reflections and can be expressed by the following equation [14]:

$$R(x) = R_s(x) + R_d(x) \quad (2)$$

Fig. 6. Components of real-world image descriptiveness 'Own source'

4. 3D CAD MODELING OF THE CONVEYOR & RENDERING DATASET IMAGES

It should be noted that, by default, the industrial robotic systems are not able to determine the spatial positions of the raw materials. However, using the images from IP cameras, objects moving on the conveyor belt can be clearly identified.

The procedure is time-consuming because the machine learning algorithms require large amounts of input data, which must be generated under different conditions. However, if the full 3D CAD model of the Flexlink XK is available, any number of image files can be generated based on virtual rendering.

3D CAD model has been designed in SketchUp program, and this model was used only as a schematic design, which will be the input of the selected Image-to-Image translation-based technique. In this method, Flexlink XK 3D CAD is expected to be a detailed model consisting of 2000 separate part models. The polygon number of the complete model was 109,692 (one hundred and nine thousand, six hundred and ninety-two). The modular design of the conveyor belt results in a high model number.

Later, the exported ".OBJ" file was imported into Blender. The color values of the 3D elements are given in the specific color wheel (HSV).

There are several ways to generate an input image dataset with the appropriate number of items. Blender provides several built-in rendering options.

In our case, the scene was created in Blender. The scene itself was a virtual scanner with keyframes. The object was placed at the origin of the virtual 3D space, and the scanning camera moved along a regular circular path (see Fig. 7). The virtual camera renders 360 images in a single round, which can be repeated as many times as necessary.

The cube object has also been rendered to pass along the Flexlink XK conveyor belt.

After the rendering process, a pix2pixHD-based deep learning network applied texture, lighting, and other environmental phenomena.

5. THE IMAGE-TO-IMAGE TRANSLATION-BASED APPROACH

This section provides a more in-depth explanation of the selected image translation technique to create photorealistic pictures.

As a result of the rendering techniques, we now have the necessary images for training models.

Recently, several applications have been developed in the deep learning field. Branytskyi et al. presented a novel use of GAN (generative adversarial networks) which was a significantly altered neural network [15]. This architecture included a digital visual processing layer [16]. Eversberg and Lambrecht explored how physics-based rendering (PBR) and domain randomization (DR) techniques influence the accuracy of industrial object vision tasks using synthetically generated data [17]. Li et al. demonstrated that models trained using domain randomization perform effectively in diverse industrial environments, especially where "real-world" data collection is limited or costly [18]. Yang et al. introduce a method combining domain randomization with style transfer. The proposed approach enhances object detection accuracy in real environments and reduces the need for extensive real-world data collection, thus making synthetic data more effective for training deep learning models [19]. Hu et al. introduced an interesting approach, which is based on an improved Cycle-GAN architecture integrated with Residual Dense Blocks (RDB) and the Structural similarity index measure [20]. This approach provides a cost-effective solution for augmenting training datasets in industrial object detection tasks.

Compared to previous approaches in industrial robotics, our proposed methodology uniquely integrates high-

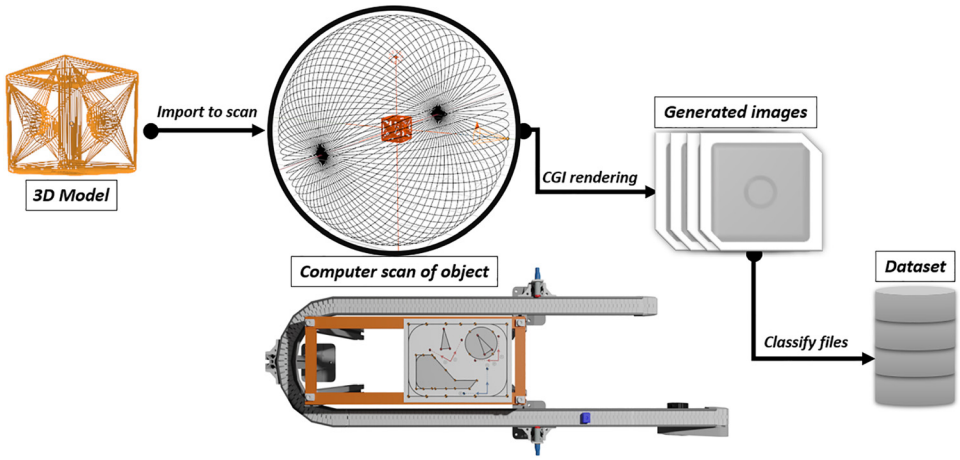

Fig. 7. The methodology of generating a dataset for deep learning 'Own source'

resolution, photorealistic synthetic image generation and explicitly targets older legacy robotic systems. However, to generate ultra-realistic images, we also focused on image domain translation, where we can learn the characteristics and shapes of photorealistic environments, including phenomena and materials.

In this work, we used a multilayer Gaussian image pyramid-based convolutional denoising and automatic coding neural network model. This model can synthesize detection results by recreating picture patches at a given resolution. Also pertinent is the work by Kaji and Kida [21] on the suitability of translation-based methods to resolve problems with reconstruction, denoising and achieve high resolution in medical image analyzation.

In our case, it is necessary to apply domain translation, which is based on learning the G: X → Y image domain mapping.

It is crucial to note that the data set {*ai*, *bi*} is used to determine whether the images *ai* are associated with this pair of images *bi*. Therefore, we will use the paired cases, as our collection is organized, and each image has a corresponding image pair.

Many deep learning architectures can now be used to accomplish image-to-image translation [22, 23]. From these, we selected the pix2pixHD-based one. The reason behind that is that it produces high-resolution and photorealistic images with exceptional accuracy.

To create a collection of images for training the detectors, the rendered images need to be converted, as shown in Fig. 8.

The selected pix2pixHD technique transforms photos using the following improved loss function [22]:

$$\min_{G}\left(\left(\max_{D_1,D_2,D_3}\mathscr{L}_{GAN}(G,D_k)\right)+\lambda\sum_{k=1,2,3}\mathscr{L}_{FM}(G,D_k)\right) \tag{3}$$

where

$$\mathscr{L}_{FM}=\mathbb{E}_{(s,x)}\sum_{i=1}^{T}\frac{1}{N_i}\left[\| D_k^{(i)}(s,x)-D_k^{(i)}(s,G(s))\|_1\right]. \tag{4}$$

We performed several training and tests based on the CycleGAN [23] architecture. Compared with the pix2pixHD approach, we obtained significantly less precise results.

6. THE DETECTOR ALGORITHMS USED IN DEEP LEARNING

As a part of this section, we explain our real-time detector's architecture used to train the model unit.

Fig. 8. The translation of the images 'Own source'

For deep learning, an architecture called YOLO (You Only Look Once) was used as visual inspection, which was described by Zamora et al. [24]. Using FPGA acceleration and an implementation based on the YOLO V3, Yu et al. ave suggested a machine vision-based defect detection procedure [25]. Another study by Zhou et al. presented a mixed/hybrid method based on YOLO Version 4 and MobileNetv2 to increase the precision of recognizing noticeably small items/objects in photographs [26].

For our method we chose the YOLO architecture [27] for real-time detection because it is capable of rapid implementation in various tasks and the precision level for detection is high.

The backbone and the detection of components are the two main architectural elements of YOLO detectors. The core layers, which are in charge of extracting the main image features, include the convolutional, batch-normalization, maxpool, and leakyRELU layers. These layers make the backbone block. The bounding box is important because it makes predictions about the objects in the machine's area.

Our scenario has primarily embedded detectors, for which we have chosen four different architectures. The detectors are: YOLO version3-5l, YOLO Version3 Spatial Pyramid Pooling (SPP) and two tiny versions of YOLOV3 (YOLOV3-tiny + YOLOV3-tiny-3l). Instead of making significant architectural changes, the detector blocks have been optimized by us for the dataset produced via data synthesis. Our primary objective is to confirm the functionality and correctness of the detectors in the light of the imagery.

The quantity of convolutional filters that came before each YOLO layer was expressed by [28]:

$$F=3\cdot(C+5) \tag{5}$$

where C denotes the quantity of the detectable classes. In our case, this value is given as 18 because we have a single object.

As a final step, we recomputed the corresponding anchor values to ensure the best detection performance attainable using the built-in clustering algorithm called k-means++ [29] (see Table 2, which contains the new values).

The detectors' loss function is given (see [30, 31]):

$$loss = clsLoss + locLoss + confLoss \tag{6}$$

where *clsLoss* denotes the classification, *locLoss* the localization, and *confLoss* the confidence *loss*, respectively.

7. TRAINING AND VALIDATION PROCESS

This section contains more details and parameters of the training and validation procedure. Each neural network was trained on a computer with a Ryzen 5900x CPU and an NVIDIA RTX 2070 GPU.

A. *pix2pxHD*

We selected the following deep learning network configuration and hyper-parameters to create the synthesized photos. The Adam algorithm [32] was employed as an

Table 2. The computed anchor values of each detector

Anchor index	YOLO Version3-Spatial Pyramid Pooling	YOLO Version3-5l	YOLO Version3-tiny-3l	YOLO Version3-tiny
0	12 × 7	10 × 6	12 × 7	10 × 9
1	9 × 10	10 × 10	9 × 10	13 × 8
2	10 × 10	9 × 10	10 × 10	10 × 10
3	11 × 10	10 × 10	11 × 10	11 × 10
4	11 × 10	10 × 10	11 × 10	12 × 10
5	11 × 10	11 × 10	11 × 10	14 × 10
6	13 × 10	11 × 10	13 × 10	–
7	14 × 9	11 × 10	14 × 9	–
8	15 × 11	13 × 8	15 × 11	–
9	–	12 × 10	–	–
10	–	11 × 10	–	–
11	–	12 × 10	–	–
12	–	13 × 10	–	–
13	–	14 × 10	–	–
14	–	15 × 12	–	–

optimizer with 0.0005 learning rate value that remained constant for the first 40 epochs before decreasing linearly.

The generator component contains the multi-scale residual network model, and the discriminator architecture is based on multi-scale GAN neural networks [22]. The most challenging part was finding the best combination of hyperparameters of the network. Therefore, we continuously trained the architecture based on datasets, and we also modified several architecture parameters, such as the number of discriminators from 2 to 5, the down-sampling levels from 3 to 9, and the feature clusters from 10 to 16. Based on these selected values, the training process converged better, and we obtained the best results.

The algorithm was trained on image pairs (400) with 1024 × 576 pixels without pixel labels. The chosen epoch value was 200. The batch size was minimized, so 20 batches were selected. For the real-world photos, we took pairs of images at the actual location under normal factory environmental conditions. During training, we exclusively used rotation and mirror augmentation. The entire network training took about 26 h.

B. *YOLO-based detectors*

We produced the initial dataset in the first step, which is separated into training and validation components. Only 100 actual captured photos were included in the validation part, while the 400 images in the training dataset were also synthetized using our translation algorithm. The image resolution in both instances was 512 × 256.

The most challenging part in the learning progress was to avoid the over-training. Therefore, we also used other picture augmentation algorithms, such as hue shifting, scaling image scenes, and adjusting saturation and exposure [33].

Table 3 lists the related techniques together with the corresponding parameters. Considering the parameters of the augmentation process, Adam [32] was the best optimizer algorithm.

Table 3. The chosen augmentation methods with the used parameters

Name of the algorithm	Used parameters
Saturation value	1.5
Exposure value	1.5
Resizing value	1.5
Hue shifting value	0.3

Table 4. shows the parameter setting of Adam (Decay and Momentum).

Additionally, it was necessary to configure several parameters of the learning rate, such as the scheduler and initial learning value. These could provide us with a proper learning rate curve in every training iteration. The chosen exponential scheduler provided by [34]:

$$l_r = l_{rinitial} \cdot exp \frac{-k}{\#epochs} \tag{7}$$

where $l_{rinitial}$ denotes the initial value of learning rate, k means a hyper-parameter, and epochs parameter contains the maximum number of training cycles.

We conducted several experiments to determine the optimal settings for the initial values and the parameter k. The number $k = 1e - 1$ is thus obtained by applying a logarithmic search on the parameter intervallum. Table 4 contains the best learning rate values that were discovered.

The learning parameters, including the burnt-in and maximum epoch value, were further adjusted in the following stage. Since the burnt-in parameter is in charge of establishing convergence and stability of the training process, selecting this value carefully is crucial. For each detector, the optimal value in our situation is 100.

As a part of the final step, we had to determine the highest epoch number that can be used by the following equation [5, 28, 31]:

$$E = 2000 \cdot C \tag{8}$$

where E is the maximum epoch number, and C is the object classes. In the case of this application, we have only one single object to detect, and the set value is given as 2000, respectively.

The training time depended on the selected architecture. Table 4 summarizes the individual timing values of each YOLO detector. In the case of smaller detectors such as

Table 4. The detector parameters and training times of YOLO architectures

Name of detector	Value of learning rate	Value of momentum	Value decay	Training time (h)
YOLO Version3-tiny	0.007	0.92	0.0003	7.82
YOLO Version3-tiny-3l	0.007	0.92	0.0003	8.45
YOLOV3- Spatial Pyramid Pooling	0.0005	0.92	0.0003	30.32
YOLO Version3-5l	0.0002	0.92	0.0003	32.25

YOLO Version3-tiny and YOLO Version3-tiny-3l we set the minibatch value to eight. In the case of huge networks (YOLOV3- Spatial Pyramid Pooling and YOLO Version3-5l), we could choose only the smaller value, which was the fourth one.

8. RESULTS OF TRAINING

A. *pix2pixHD*

Table 5 displays the outcomes of the chosen pix2pixHD method. Figure 9 displays the sampling images that were acquired throughout the training procedure. Additional metrics have also been chosen, which we applied to the entire image and object regions.

To determine whether the current training is successful, we used a robust approach known as complex wavelet structural similarity (CW-SSIM) [35], since it is quite helpful in our situation and invariant to different transformations like rotation and resizing. We also tested the classical structural similarity but found that it was not necessarily applicable in our case because the disparity in object and image sizes may cause inaccuracies.

In our application case, we conducted several training and case tests, such as different lighting conditions and rearranged environmental backgrounds. Because the information content is crucial for the training, we also computed the selected metric with a standard deviation for object regions, not only the entire image.

We included the CW-SSIM as a metric in the training process and calculated its value at the end of each training period. At the beginning, we experienced that the quality of the entire image region reached a high level relatively quickly within a few iterations. However, the object regions and their content were not discernible until the tenth iteration.

It is important to note that the success of the training depends mainly on the nature of the problem. However, with the help of CW-SSIM, we have definitively established the critical level below which the object cannot be

Table 5. Image quality results of the synthetically generated images during the training

#epochs	$\overline{CWSSIM_I}$	$\sigma CWSSIM_I$	$minCWSSIM_I$	$maxCWSSIM_I$	$\overline{CWSSIM_{obj}}$	$\sigma CWSSIM_{obj}$	$minCWSSIM_{obj}$	$maxCWSSIM_{obj}$
1	0.6477	7.0878e-5	0.6457	0.6493	0.4085	0.0761	0.2502	0.7217
2	0.7281	7.6168e-4	0.7259	0.7296	0.4401	0.0868	0.2258	0.7459
3	0.7576	6.7552e-4	0.7560	0.7591	0.4574	0.0896	0.2282	0.7562
4	0.7816	6.8852e-4	0.7796	0.7827	0.4653	0.0897	0.2295	0.7578
5	0.8006	6.6298e-4	0.7990	0.8019	0.4879	0.0933	0.2083	0.7674
6	0.8113	6.6731e-4	0.8094	0.8128	0.5056	0.0910	0.2252	0.7810
7	0.8207	5.8425e-4	0.8193	0.8218	0.4967	0.0924	0.2262	0.7662
8	0.8256	6.2135e-4	0.8242	0.8272	0.5176	0.0872	0.2485	0.7962
9	0.8282	6.5893e-4	0.8257	0.8295	0.5334	0.0817	0.3025	0.7884
10	0.8409	5.9780e-4	0.8423	0.8392	0.5779	0.068	0.3366	0.7789
20	0.8583	5.9699e-4	0.8594	0.8569	0.6039	0.0651	0.3469	0.7716
30	0.8687	4.9685e-4	0.8696	0.8673	0.6017	0.0646	0.3631	0.7688
40	0.8738	5.1662e-4	0.8749	0.8723	0.5986	0.0626	0.3573	0.7862
50	0.8772	5.1611e-4	0.8782	0.8756	0.6089	0.0659	0.3390	0.8095
60	0.8784	4.5833e-4	0.8793	0.8772	0.6113	0.0602	0.3962	0.7962
70	0.8804	4.7160e-4	0.8814	0.8793	0.6099	0.0625	0.3465	0.7810
80	0.8815	4.5871e-4	0.8823	0.8803	0.6113	0.0601	0.3817	0.7972
90	0.8809	4.8511e-4	0.8821	0.8797	0.6214	0.0612	0.3857	0.7896
100	0.8825	3.9562e-4	0.8833	0.8815	0.6206	0.0631	0.3676	0.7804
140	0.8855	3.5346e-4	0.8864	0.8845	0.6291	0.0582	0.3991	0.7858
200	0.8827	3.9410e-4	0.8835	0.8817	0.6225	0.0572	0.4041	0.7579

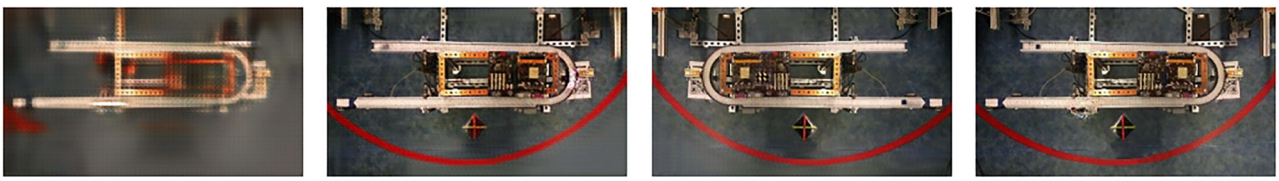

Fig. 9. The outputs of the pix2pixHD algorithm during the training of synthesized image scenes: (a) sampled image at the beginning of the training, (b) sampled image at a similarity value of about 0.5, (c) sampled image at a similarity value of above 0.6, (d) sampled image at the end of the training 'Own source'

Table 6. Detector results depend on the quality of the synthesized images

$\overline{CWSSIM}_{obj}$	YOLOV3- Spatial Pyramid Pooling				YOLO Version3-5l				YOLO Version3-tiny-3l				YOLO Version3-tiny			
	mAP	precision	recall	f1-score	mAP	precision	recall	f1-score	mAP	precision	recall	f1-score	mAP	precision	recall	f1-score
1.0000	8.4234e-1	9.0031e-1	8.0122e-1	8.5995e-1	9.5985e-1	1.0000	8.8881e-1	9.3158e-1	8.1158e-1	8.3458e-1	8.5401e-1	8.4045e-1	9.1123e-1	9.2423e-1	9.0024e-1	9.1486e-1
4.0851e-1	4.4491e-1	5.0251e-1	5.2021e-1	5.1214e-1	4.8391e-1	6.2475e-1	1.2245e-1	2.1269e-1	8.6136e-2	0.0000	0.0000	0.0000	2.4102e-1	4.6364e-1	3.0051e-1	3.6325e-1
4.4014e-1	4.9163e-1	5.7156e-1	5.2694e-1	5.5156e-1	5.9263e-1	8.2236e-1	3.5698e-1	4.9548e-1	1.9999e-1	5.6458e-1	1.2213e-1	2.0091e-1	3.477e-1	6.2584e-1	2.0622e-1	3.0192e-1
4.5746e-1	5.6062 -1	7.3112e-1	4.0156e-1	5.2116e-1	6.7912e-1	8.6775e-1	4.7446e-1	6.1106e-1	3.2888e-1	4.2112e-1	4.0025e-1	4.1996e-1	4.474e-1	5.9156e-1	4.3364e-1	4.9094e-1
4.9677e-1	5.5381e-1	7.7478e-1	2.5569e-1	3.8478e-1	7.5814e-1	8.3446e-1	7.5325e-1	7.9478e-1	4.1861e-1	8.8487e-1	1.7144e-1	2.9487e-1	4.881e-1	6.7168e-1	3.0658e-1	4.1495e-1
5.1762e-1	5.5664e-1	6.6458e-1	5.7332e-1	6.1569e-1	7.8736e-1	9.6458e-1	5.7021e-1	7.2745e-1	6.2956e-1	9.4023e-1	3.8136e-1	5.4354e-1	6.487e-1	8.7658e-1	6.5354e-1	7.4569e-1
5.7796e-1	7.5028e-1	7.2323e-1	7.7125e-1	7.5158e-1	9.4694e-1	1.0000	9.0158e-1	9.5458e-1	6.6854e-1	7.2489e-1	6.5158e-1	6.8358e-1	6.448e-1	8.2125e-1	6.8368e-1	7.4698e-1
6.0397e-1	8.189e-1	7.8435e-1	8.0681e-1	7.9784e-1	9.1564e-1	9.5264e-1	8.8154e-1	9.1102e-1	7.3612e-1	8.4265e-1	6.5695e-1	7.3448e-1	8.750e-1	9.2365e-1	8.8354e-1	9.0222e-1
6.2916e-1	7.9335e-1	7.2912e-1	7.8654e-1	7.4145e-1	9.3946e-1	9.2145e-1	8.8245e-1	9.0254e-1	7.8928e-1	8.3489e-1	7.3154e-1	7.7487e-1	8.950e-1	8.6125e-1	9.0105e-1	8.8182e-1

distinguished from the background. In all test cases, we found this critical value at 0.5.

By calculating the dataset's standard values (best and worst outcomes), we were able to determine that, although the object sections had significantly larger values, the overall images had very few variances. In the second case, the value of similarity can highly impact the performance of the detectors, depending on the ratio of the regions' quality.

B. *The YOLO detectors*

Table 6 contains the results of trained YOLO detectors. The results of F1-score, mean average precision, recall and precision also listed in it.

As a reference, we also trained the detectors on real-world images, and the first row contains the corresponding values. The dataset generated was relatively large and consisted of 256,000 augmented images and 400 real-world images for validation.

As mentioned in the previous chapter, the obtained CW-SSIM value is decisive for detectors, which can also be considered a synthetic threshold in a technical sense. From the values, we concluded that when the complex wavelet structural similarity dropped below 0.5, the detectors' accuracy dramatically declined because the object to be perceived blended into the background.

The evaluation also found that the performance of the detectors approximates the reference results trained on real photos from a similarity value of at least 0.6.

The reason for this is that at 0.5, the object is already detectably separated from the background, but the information loss is still significant enough to reduce the performance of the detectors. It is also important to note that determining the level at which YOLO network almost reaches the accuracy values measured on the original real-world images depends largely on the detail of the objects to be detected and the robustness of the chosen detector. In our case, this value was measured at 0.6. However, this may vary, and in some cases, it may even increase for more complex objects and detectors with different architectures.

In conclusion, the presented method can be considered successful if we can synthesize images in which the object region can be noticeably distinguished from the background.

Additionally, several tests were carried out to examine the generalization capability with the current synthetic data for various real-world scenarios. YOLO detectors were tested and trained with synthetic data in an industrial environment with different lighting and background conditions. In conclusion, the difference in mAP was only 3–5%, which is an acceptable result.

9. CONCLUSION

The Vehicle Manufacturing laboratory has many industrial robot units. It is an opportunity to add new functionality to older machines. In manufacturing, it is now expected that machines used in industry will have some level of machine learning in addition to basic standard operation. The Vehicle

Manufacturing Lab was created in the spirit of testing individual control programs before they are actually integrated into the production line. In the laboratory, artificial intelligence is constantly being incorporated into machines, so our future production cell model will be based on new foundations. The project detailed above is another step in this progress.

The training of neural networks using data synthesis was done in this work. 3D modeling and image-to-image translation were used in the implementation of data synthesis. The complex wavelet structural similarity measure was used to estimate the quality and similarity of the synthetically generated images, and YOLO detectors were trained using these results. This is the largest discrepancy between the artificially created data set and the actual image data set. We found the threshold value at which detector training accuracy is comparable to reference training accuracy.

With the help of the solution, we were able to install the deep learning neural network on an existing industrial robot unit without having to replace its control system entirely or incur any further expenses.

Conflict of interest: Géza Husi is a member of the Editorial Board of the journal, therefore they did not take part in the review process in any capacity and the submission was handled by a different member of the editorial board. The submission was subject to the same process as any other manuscript and editorial board membership had no influence on editorial consideration and the final decision.

REFERENCES

[1] T. Erdei, Z. Molnár, N. Obinna, and G. Husi, "A novel design of an augmented reality based navigation system & its industrial applications," *ACTA IMEKO*, vol. 7, p. 57, 04 2018.

[2] Z. Xu, X. Zhou, H. Wu, X. Li, and S. Li, "Motion planning of manipulators for simultaneous obstacle avoidance and target tracking: an rnn approach with guaranteed performance," *IEEE Trans. Ind. Electronics*, vol. 99, pp. 1–1, 04 2021.

[3] D. Al-Alimi, Y. Shao, R. Feng, M. A. A. Al-qaness, M. A. Elaziz, and S. Kim, "Multi-scale geospatial object detection based on shallow-deep feature extraction," *Remote Sensing*, vol. 11, no. 21, 2019. [Online]. Available: https://www.mdpi.com/2072-4292/11/21/2525.

[4] Z. Xu, X. Zhou, and S. Li, "Deep recurrent neural networks based obstacle avoidance control for redundant manipulators," *Front. Neurorobotics*, vol. 13, 07 2019.

[5] T. P. Kapusi, T. I. Erdei, G. Husi, and A. Hajdu, "Application of deep learning in the deployment of an industrial scara machine for real-time object detection," *Robotics*, vol. 11, no. 4, 2022. [Online]. Available: https://www.mdpi.com/2218-6581/11/4/69.

[6] KUKA Robot Group. KR 5 arc specification; KUKA Roboter GmbH D-86165 Augsburg; Germany issued: 21.03.2011.

[7] A. Talli and A. Giriyapur, "Kinematic analysis and simulation of industrial robot based on RoboAnalyzer," 2021. https://doi.org/10.1007/978-981-33-4176-0_40

[8] M. Hullin, E. Eisemann, H.-P. Seidel, and S. Lee, "Physically-based real-time lens flare rendering," *ACM Trans. Graph.*, vol. 30, p. 108, 07 2011.

[9] S. Lee and E. Eisemann, "Practical real-time lens-flare rendering," *Computer Graphics Forum*, vol. 32, 07 2013.

[10] D. Seland, "An industry demanding more: intelligent illumination and expansive measurement volume sets the new helix apart from other 3-d metrology solutions," vol. 50, no. 8, p. 22+, Aug 2011. Article. [Online]. Available: https://link.gale.com/apps/doc/A264581412/AONE.

[11] P. Martinez, D. R. Ahmad, and M. Al-Hussein, "A vision-based system for pre-inspection of steel frame manufacturing," *Autom. Constr.*, vol. 97, pp. 151–63, 01 2019.

[12] Y. Wu, Q. He, T. Xue, R. Garg, J. Chen, A. Veeraraghavan, and J. T. Barron, "How to train neural networks for flare removal," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 2239–47.

[13] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau, "Robust physical adversarial attack on faster R-CNN object detector," *ArXiv*, vols abs/1804.05810, 2018.

[14] T. P. Kapusi, L. Kovács, and A. Hajdu, "Deep learning-based anomaly detection for imaging in autonomous vehicles," in *2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS)*, 2022, pp. 142–7.

[15] V. Branytskyi, M. Golovianko, D. Malyk, and V. Terziyan, "Generative adversarial networks with bio-inspired primary visual cortex for industry 4.0," *Proced. Computer Sci.*, vol. 200, pp. 418–27, 01 2022.

[16] S. Mei, W. Yudan, and G. Wen, "Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model," *Sensors*, vol. 18, p. 1064, 04 2018.

[17] L. Eversberg and J. Lambrecht, "Generating images with physics-based rendering for an industrial object detection task: realism versus domain randomization," *Sensors*, vol. 21, p. 7901, 2021. https://doi.org/10.3390/s21237901

[18] J. Li, P.-L. Götvall, J. Provost, and K. Åkesson, "Training convolutional neural networks with synthesized data for object recognition in industrial manufacturing," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain, 2019, pp. 1544–7. https://doi.org/10.1109/ETFA.2019.8869484

[19] X. Yang, X. Fan, J. Wang, and K. Lee, "Image translation based synthetic data generation for industrial object detection and pose estimation," *IEEE Robotics Autom. Lett.*, vol. 7, no. 3, pp. 7201–8, July 2022. https://doi.org/10.1109/LRA.2022.3180403

[20] J. Hu, F. Xiao, Q. Jin, G. Zhao, and P. Lou, "Synthetic data generation based on RDB-CycleGAN for industrial object detection," *Mathematics*, vol. 11, p. 4588, 2023. https://doi.org/10.3390/math11224588

[21] S. Kaji and S. Kida, "Overview of image-to-image translation by use of deep neural networks: denoising, super-resolution, modality conversion, and reconstruction in medical imaging," *Radiological Phys. Technol.*, vol. 12, 06 2019.

[22] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," pp. 8798–807, 06 2018.

[23] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[24] M. Zamora, J. A. Castro Vargas, J. Azorin-Lopez, and J. Rodrıiguez, "Deep learning-based visual control assistant for assembly in industry 4.0," *Comput. Industry*, vol. 131, p. 103485, 10 2021.

[25] L. Yu, J. Zhu, Q. Zhao, and Z. Wang, "An effi cient yolo algorithm with an attention mechanism for vision-based defect inspection deployed on fpga," *Micromachines*, vol. 13, no. 7, 2022. [Online]. Available: https://www.mdpi.com/2072-666X/13/7/1058.

[26] X. Zhou, X. Xu, W. Liang, Z. Zeng, S. Shimizu, L. T. Yang, and Q. Jin, "Intelligent small object detection for digital twin in smart manufacturing with industrial cyber-physical systems," *IEEE Trans. Ind. Inform.*, vol. 18, no. 2, pp. 1377–86, 2022.

[27] P. Adarsh, P. Rathi, and M. Kumar, "Yolo v3-tiny: object detection and recognition using one stage improved model," 03 2020, pp. 687–94.

[28] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: optimal speed and accuracy of object detection," *CoRR*, vols abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934.

[29] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding," vol. 8, pp. 1027–35, 01 2007.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," pp. 779–88, 06 2016.

[31] J. Redmon and A. Farhadi, "Yolov3: an incremental improvement," *CoRR*, vols abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767.

[32] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980.

[33] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *CoRR*, vols abs/1712.04621, 2017. [Online]. Available: http://arxiv.org/abs/1712.04621.

[34] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," *CoRR*, vols abs/1910.07454, 2019. [Online]. Available: http://arxiv.org/abs/1910.07454.

[35] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, "Complex wavelet structural similarity: a new image similarity index," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2385–401, 2009.