Adapting synthetic training data in deep learning-based visual surface inspection to improve transferability of simulations to real-world environments

Ole Schmedemann[a,*], Simon Schlodinski[a], Dirk Holst[a], and Thorsten Schüppstuhl[a]

[a]Hamburg University of Technology, TUHH, Institute of Aircraft Production Technology, Denickestr. 17, 21073 Hamburg, Germany

ABSTRACT

Learning models from synthetic image data rendered from 3D models and applying them to real-world applications can reduce costs and improve performance when using deep learning for image processing in automated visual inspection tasks. However, sufficient generalisation from synthetic to real-world data is challenging, because synthetic samples only approximate the inherent structure of real-world images and lack image properties present in real-world data, a phenomenon called domain gap. In this work, we propose to combine synthetic generation approaches with CycleGAN, a style transfer method based on Generative Adversarial Networks (GANs). CycleGAN learns the inherent structure from real-world samples and adapts the synthetic data accordingly. We investigate how synthetic data can be adapted for a use case of visual inspection of automotive cast iron parts, and show that supervised deep object detectors trained on the adapted data can successfully generalise to real-world data and outperform object detectors trained on synthetic data alone. This demonstrates that generative domain adaptation helps to leverage synthetic data in deep learning-assisted inspection systems for automated visual inspection tasks.

Keywords: deep learning, automated visual inspection, synthetic training data, domain adaptation, surface inspection, non-destructive testing, industrial endoscope inspection, borescope

1. INTRODUCTION

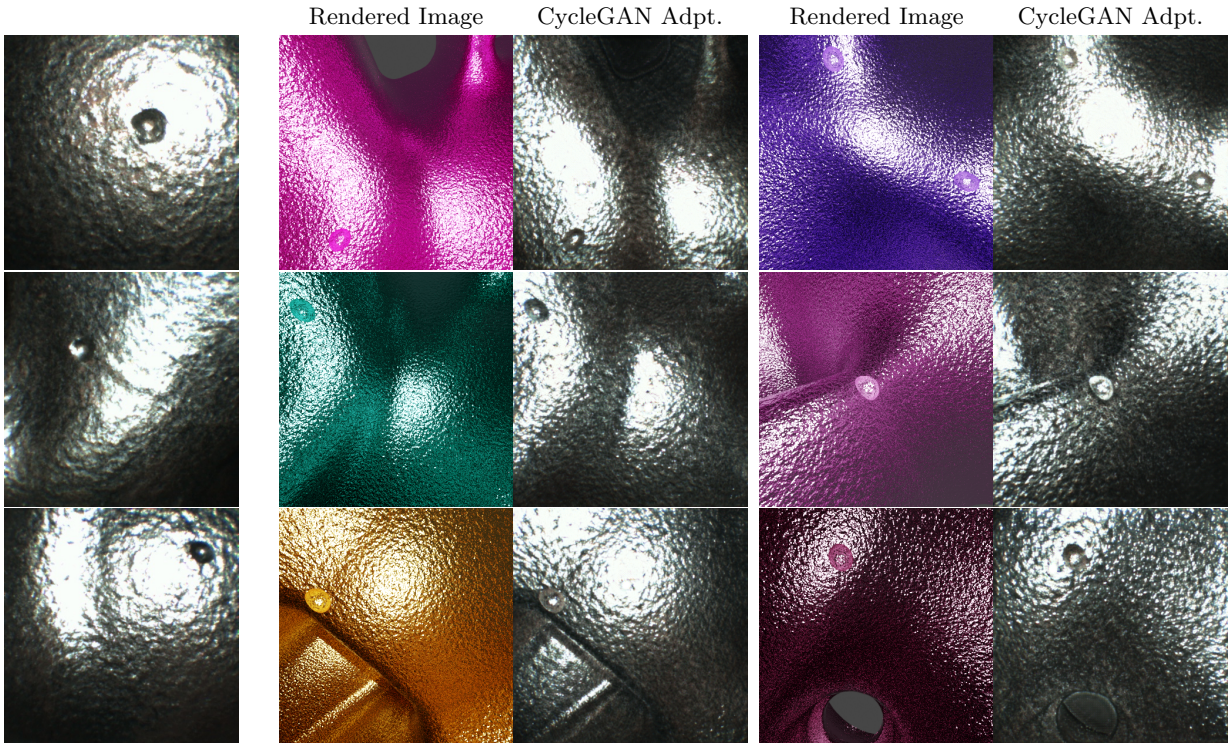
Deep learning-based visual surface inspection relies on data to train AI models. In an industrial context sufficient data is usually sparse. An alternative is to use synthetically generated data, e.g. through rendering or image synthesis. However, models trained with synthetic data alone often struggle to transfer to real-world applications. This is mainly because rendering algorithms are only an approximation of real-world cameras, objects, and illumination and therefore rendered images lack realism or differ from real-world images in low-level image statistics. This observation is known as domain gap[1], domain shift[2], reality gap[3], or sim2real gap[4].

Recent work has shown success in using advanced domain adaptation techniques to translate synthesised images into real-world domains[5,6]. Generative Adversarial Networks (GANs) learn the inherent structure from real-world samples and adapt the rendered images accordingly. However, it is uncertain whether state-of-the-art deep domain adaptation techniques can be utilised in the field of automated visual inspection to translate synthetic image data to the real-world domain and whether the translated data can be used as a surrogate for real-world data to train deep learning-based defect detection models.

We expect the coupling of image rendering with a posteriori domain adaptation techniques to be highly beneficial in industrial applications. Two main benefits are expected. Firstly, the effort and therefore the time and cost of generating synthetic data will be significantly reduced because less modelling effort is needed to set up 3D models, textures, illumination-, and camera parameter for the rendering pipeline, because less sensor realism is needed since the realism is introduced a posteriori with the image translation process. This is particularly beneficial as modelling effort increases exponentially with increasing realism. Secondly, the performance of

Further author information:
* E-mail: ole.schmedemann@tuhh.de

Automated Visual Inspection and Machine Vision V, edited by Jürgen Beyerer, Michael Heizmann,
Proc. of SPIE Vol. 12623, 1262307 · © 2023 SPIE · 0277-786X · doi: 10.1117/12.2673857



(a) Real world samples (b) Rendered images before and after domain adaptation with CycleGAN

Figure 1: The real-world samples (a) show defects on the surface of an object. The images are taken with an industrial borescope inside the object's cavity. We model the object, defects, textures, camera, and illumination parameters and render synthetic images. We train CycleGAN[7] to transfer images from the synthetic rendered domain to the real-world domain (domain adaptation) to increase realism and and increase the suitability of the images for use as training data.

defect detection models trained on translated image data will be improved by introducing image properties such as sensor noise that are difficult or impossible to account for in the rendering pipeline alone.

In this work, we couple domain adaptation and synthetic data generation for deep learning-based visual surface inspection. In our use case, the cavities of cast iron parts need to be inspected with industrial borescopes after demoulding. Fig. 1a shows example images from the use case. We created a dataset of rendered images with a procedural data generation pipeline[8]. Next we implemented the most common method for deep domain adaptation of unpaired image data, CycleGAN[7]. We trained CycleGAN on both real-world samples and rendered images and then used it to translate the rendered images. Fig. 1b shows our qualitative results by comparing rendered images with the corresponding translated image. Next, we trained a deep object detection model with different datasets and empirically compared the performance of the model on a real test dataset. We show that the translated data increases the detection rate of the model when used as training data compared to using rendered data alone.

2. RELATED WORK

Discrepancies between rendered and original data occur due to the inherent limitations of rendering engines to accurately represent the real world. Rendering engines that aim to create realistic images or animations, rely on approximations to simulate the interaction of light with surfaces. In addition, 3D objects within virtual scenes are only approximations of their real-world counterparts. Furthermore, virtual scenes are not the same as real scenes, as they involve simplified textures and models.

A specific limitation is that the rendering process uses a pinhole camera model, which does not fully capture the intricacies of real camera sensors. This mismatch can result in differences between the rendered image and the original data captured by the camera. Rendering engines approximate the complex interactions between light and surfaces, resulting in discrepancies between the rendered output and the original data. Simplified representations of 3D objects and textures, along with the inadequate description of real camera sensors, contribute to the variations observed in the rendering process.

This deviation of data distributions between two domains, a training domain (source domain) and a related test domain (target domain), is reffered to as a domain gap[1]. In terms of AI application, the domain gap results in a lack of transferability of knowledge learned in one domain to the other. In the case of a synthetically trained defect detection AI, the domain gap reduces the defect detection performance in the real-world domain.

2.1 Image Domain Adaptation

Domain adaptation techniques are used when the domain gap between two domains needs to be reduced. In general domain adaptation techniques are used when there is little or no data available in the target domain while large annotated datasets exist or can be be inexpensively created in another, related domain, such as rendered image data[9].

Domain adaptation techniques for image-to-image translation can be divided into "shallow" and "deep" adaptation methods. "Shallow" architectures include domain adaptation approaches with only one hidden layer, while "deep" AI architectures have multiple hidden layers. "Deep" AI architectures can be further subdivided into:

- "Shallow methods with deep features": A deep KNN extracts features from the input. The extracted features are then processed by a shallow domain adaptation architecture.
- "Fine tuning of deep AI architectures": A convolutional neural network is trained on data from one domain. After the training, it is fine-tuned with data from domain B.
- "Deep Domain Adaptation": Multilayer convolutional neural networks are used for domain adaptation. We focus on Deep Domain Adaptation methods because they offer a significant performance improvement over previous state-of-the-art shallow methods[10].

Deep domain adaptation techniques differ depending on the availability of class labels in the source and target domains. Class labels are present in both the rendered and the real-world domains. Numerically, the amount of rendered data is only limited by the rendering time per image while the amount of labelled real-world samples depends strongly on the use case. In addition, there are no image pairs from both domains that show a scene in both domains from the same viewpoint and that would describe an exact one-to-one translation from the one domain to the other. Therefore, the domain adaptation technique must take unpaired image data as an input.

The most prominent deep domain adaptation techniques for unpaired image data are CycleGAN[7], UNIT[11], DRIT[12], or SimGAN[5], which demonstrate exceptional image translation capabilities usually demonstrated for everyday objects, such as humans, animals, paintings, etc., but do not focus on images from industrial visual inspection tasks. These approaches are based on Generative Adversarial Networks (GANs). An overview of a variety of GAN-based state-of-the-art domain adaptation methods is provided by Toldo et al.[13] and by Wang and Deng[10].

2.2 Domain Adaptation for Rendered Images

For the application of an image-to-image translation between a synthetic and a real domain, GAN-based methods have already been investigated in some application areas. Isola et al.[14] use a conditional GAN to translate segmentation masks into real-looking street images. Furthermore, Bousmalis et al.[6] use a GAN-based network, called PixelDA. This network is used to transform synthetic images of 3D models representing everyday objects (e.g. hammer, light, telephone) into realistic looking photographic images. The transformed images are then used to train a network to classify the objects contained in the image and determine their pose. Shrivastava et al.[5] use the SimGAN model to increase the realism of synthetic modelling of human eye parts using real

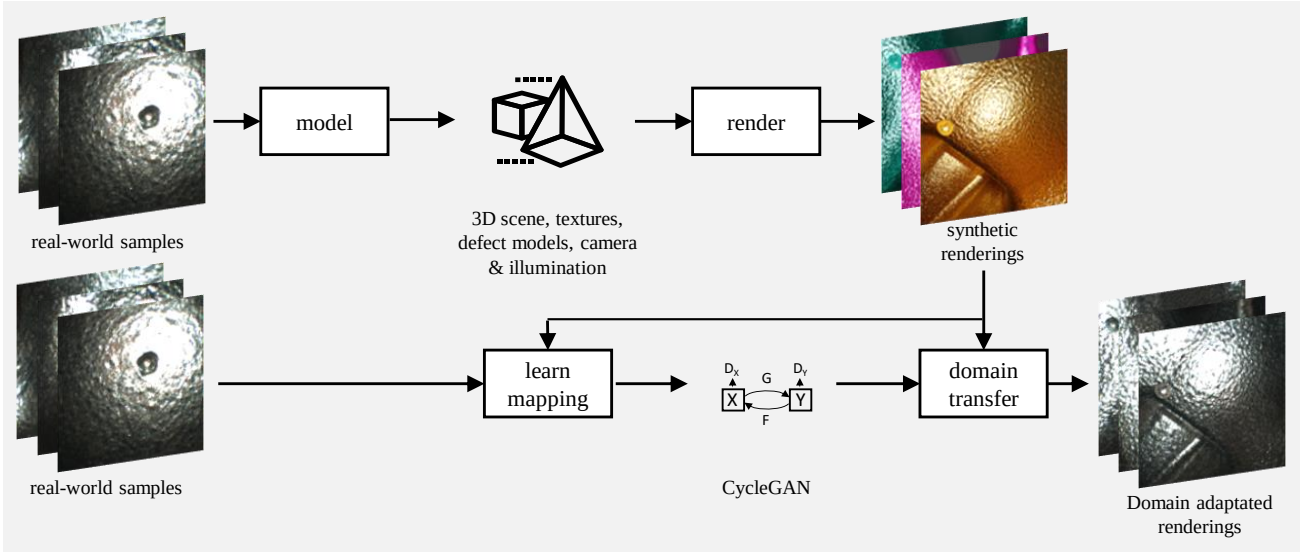


Figure 2: Pipeline for coupling image synthesis with deep domain adaptation used in this work.

reference data. The more realistic appearing synthetic data is then used to train gaze direction algorithm. The use of a so-called CyCADA model is presented by Hoffman et al.[15]. This model transforms street images from a computer game into real-looking images.

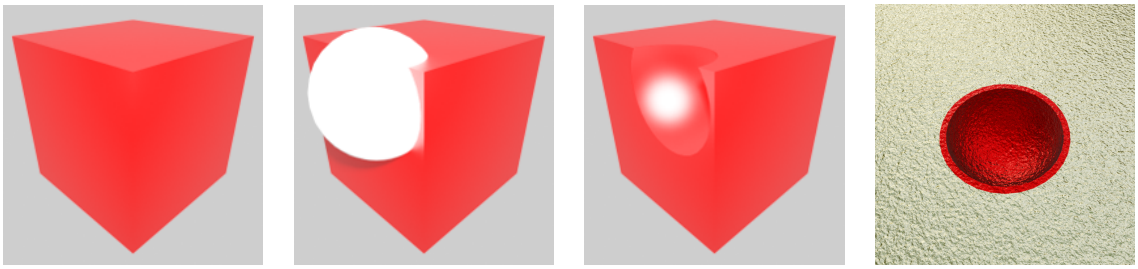
The presented work demonstrates the ability of state-of-the-art GAN-based domain adaptation techniques to bridge the domain gap between the synthetic (rendering) domain and the real-world domain. However, the work presented focuses on domains where the availability of training data to train the adaptation model is expected to be sufficiently high, such as everyday objects or street scenes. In industrial contexts the availability is often low. In addition images of objects in an inspection task show often few variety and low contrast between the objects that needs to be detected, such as a scratch or dent, and the rest of the part's surface.

3. APPROACH

We investigate the suitability of combining deep domain adaptation with image rendering for automated visual inspection by generating synthetic images for an industrial use case and a posteriori transferring the synthetic images to the real-world domain. The industrial use case is the visual inspection of a shallow cast iron part after demoulding in the automotive industry. We use industrial borescopes to inspect the part's cavities. Typical defects to be detected are blowholes or pinholes, which can be caused by entrained air or other gases in the melt. We manually modified two of the cast iron parts with punch marks to simulate the real defect pattern and obtained images with the hand-held industrial borescope, see Fig. 1a.

Fig 2 summarises our approach. First, using the endoscopic samples and a 3D model of the part to model, a virtual scene is modelled, including textures, defect models, and camera and illumination parameters. Synthetic images are rendered from the scene. The pipeline for generating training data has been described in more detail in our previous work[8]. In this pipeline, image features that are not relevant for defect detection are randomised, such as the base color texture, the illumination intensity, or the viewpoint of the virtual camera. A defect is introduced into the part with a negative defect object, see Fig. 3. The negative shape is introduced into the 3D model using a Boolean operation. A thin-walled object remains in the part which is used to annotate the rendered image pixel by pixel by assigning a class to each pixel, either 'part' or 'defect', see Fig 5d.

Next, we trained a deep domain adaptation model to learn the mapping from the rendered images to the domain of the real-world samples. We chose CycleGAN[7] as the domain adaptation model. CycleGAN can work with unpaired image data and has a large number of users who have demonstrated its suitability for various applications.



(a) (b) (c) (d)

Figure 3: Demonstrative display of the insertion of a defect into a 3D object. (a) 3D object; (b) defect tool negative is positioned (white); (c) Boolean operation is applied and the spherical defect is introduced into the part; (d) a thin-walled defect object is created, which is used to annotate the rendered images.

3.1 Comparison of Real-World and Rendered Datasets

We manually annotated the real-world images with bounding box annotations while the training data generator pipeline provides bounding box annotations to the rendered images. Table 1 summarises dataset statistics for both the real-world and rendered datasets. The two datasets differ significantly in the total number of images and the average defect size. The small average defect size in the rendered images is due to the brute force approach of the training data generation pipeline, where defect positions and camera viewpoints are generated independently, resulting in many rendered images having only small visible defects because they are located in the background relative to the camera.

Table 1: Dataset statistics for real-world and rendered datasets.

	real-world	rendered
dataset size	1000	49541
with defect	500	29181
defect-free	500	20360
average defect size	2791 px	366 px
image resolution	400 x 400 px	400 x 400 px

The textures in the two domains differ in terms of their roughness, reflectivity, and color. The surface of the real component appears to be smoother or rougher in places than the synthetic one, i. e. it has greater variations

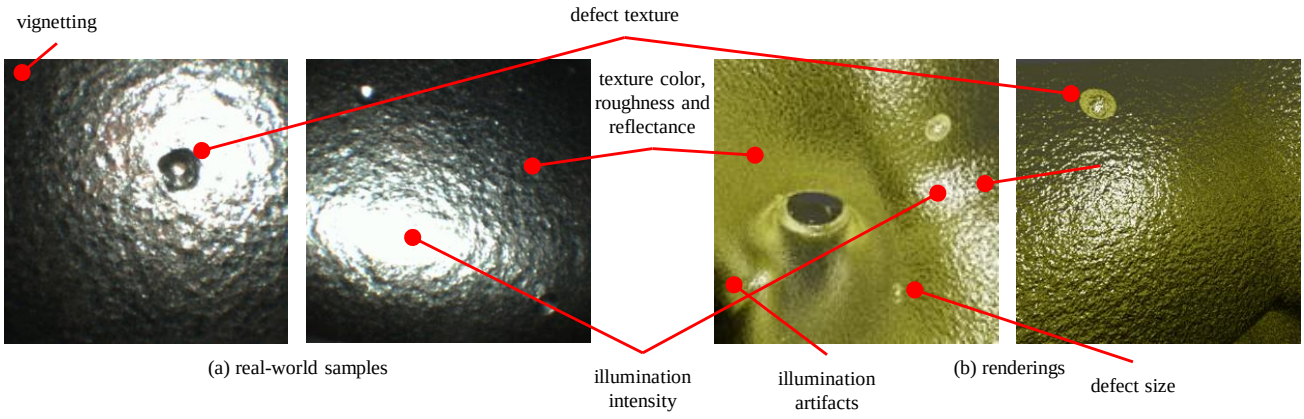


Figure 4: Qualitative differences between real-world samples and rendered images.

in roughness, see Fig. 4. In addition, the surface of the real component is more reflective, as can be seen from the almost white patches in the real white spots in the real images. The two aforementioned characteristics also lead to different shading of the texture in both domains. Furthermore, the surface has a similar but slightly different color.

Differences between the real-world and synthetic dataset can also be observed in the illumination. The intensity of the light source is different. In addition, the vignetting, which is clearly visible in the real images is particularly noticeable. The stronger vignetting can be seen in the darkening of the real images towards the edges of the image. In the synthetic images, the vignetting is less pronounced. The synthetic images also show lighting artefacts. These are white spots of a few pixels on more distant components. Such artifacts are often due to a physically incorrect simulation of light rays.

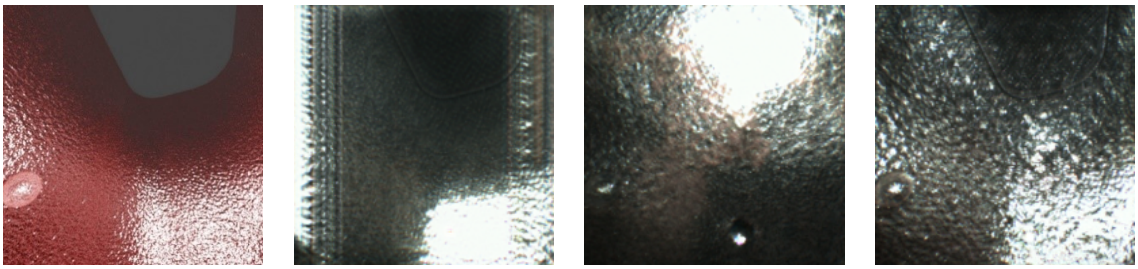
3.2 CycleGAN Implementation

CycleGAN was first introduced by Zhu et al.[7] in 2017. CycleGAN consists of two generator networks, each responsible for transforming images from one domain to another, and two discriminator networks that distinguish between real and generated images in each domain. The key innovation is the introduction of cycle consistency loss, which forces the reconstructed images to be similar to the original input. This loss term ensures that the mapping is consistent in both directions, allowing the generators to capture the underlying structure and style of the domains being learned. By simultaneously optimising the generators and discriminators, CycleGAN enables the transformation of images across domains without the need for paired training data.

We used the PyTorch implementation provided by the authors of CycleGAN. The challenges we encountered in the implementation were checkerboard artifacts, white spot artifacts, and hallucinations, see Fig. 5. Checkerboard artifacts result from the image synthesis process. Generative models typically first generate a refined low-resolution image and then these models sequentially resolve individual image regions using a wandering kernel. Overlaps of the image regions examined by the kernel are deconvolved multiple times, which can cause the checkerboard artifacts. The phenomenon that features of the target domain appear at random positions in the translated images or that image features disappear in the refined images is called hallucination. In our case, some defects that were present in the rendered image disappeared in the translated image and some additional defects appeared that were not present in the rendered image.

While both reduce the realism of a translated image, they are particularly critical when they change the position or size of defect objects in the image. When the defect is changed or new defects are created by the model, the bounding box annotation is left untouched and still represents the defect location in the original rendered image. This would lead to semantic inconsistency and make the translated images less suitable as training data.

Checkerboard and white spot artifacts are prevented in this work by two measures. The first measure is to reduce the complexity of the input image data by means of image pre-processing. For this purpose, the input images for CycleGAN training are scaled down to a lower resolution and then randomly cropped. In addition,



(a) (b) (c) (d)

Figure 5: Artifacts and hallucinations when translating a rendered image with CycleGAN. (b)-(d) are translations from (a). (a) Source image (rendered image); (b) checkerboard artifacts (right and left) and white spot artifacts (bottom right); (c) hallucinated defect (bottom center); (d) successful translation without visible artifacts and hallucinations

images of the defect-free object are mixed into the training dataset. With these measures we avoid major artifacts in the translated images.

For the hallucinations, we hypothesise that they represent a kind of overfitting of the CycleGAN algorithm. In overfitting, the domain adaptation model remembers the distribution of features of a distribution in the feature space, in this case the likely positions of defects in the images, and imitates them in the translated images. These hallucinations should be minimised to improve the quality of the translated images. To reduce the hallucinations, the following measures are taken. Firstly, the percentage of defects in the training data is reduced to 50%. This reduces the frequency with which a defect occurs in a particular image configuration at a particular location. In this way, less of certain defect features of the target distribution, such as their frequently occurring positions, are to be learned.

Second, we increase the minimum defect size of the CycleGAN model training data. The synthetic dataset has an average defect size of 366 px (see Table 1), which makes surface defects appear much smaller than in the real dataset, which has an approximate average defect size of 2791 px. To minimise the disappearance of actual defects, we make the average defect size in both domains reach a comparable level by only considering rendered images in the training dataset with an minimum defect size of 1500 px as the average.

This decision is based on the hypothesis that defects that appear more clearly in the input images are more likely to be detected by the CycleGAN algorithm as anomalies or geometric structures of the input image rather than surface texture. The CycleGAN algorithm is said to preserve such geometric structures through cycle consistency. In short, it is assumed that larger appearing defects are more likely to be preserved than smaller defects. These measures can minimise or even eliminate hallucinations.

4. EXPERIMENTS

In this section we investigate what performance can be achieved with translated rendered images compared to only rendered images as training data. In order to evaluate the performance we train a deep object recognition model with different datasets. First, we separate 250 real-world samples with defects as our final test data which will not be used for training CycleGAN or the object recognition model. The remaining 250 real-world samples with defects are used as training and validation data in the following experiments. At first, we evaluate the performance of training the model on the 250 real-world samples without rendered data. We split these data in an 85:15 ratio into training and validation data. The model from the epoch with the best performance on the validation data is tested on the test dataset.

Next, we benchmark the performance of the rendered data without translation. We sort the rendered dataset so that only the following minimum defect sizes are allowed: 0 px, 400 px, 1500 px. We use these three datasets as training data and the 250 real-world samples that have been used as training data in the previous benchmark as validation data. The best performing model is then tested on the test dataset. From each variation of the hyperparameter we use the parameter that leads to the best performing model in the next attempt.

In our third experiment, we investigate the performance of the translated rendered images. Following the pipeline in Fig. 2 we train CycleGAN with both real-world samples and rendered images. First, we divide the 250 real-world samples in an 85:15 ratio. The 85% are used to train CycleGAN while the 15% are used as validation data to train the model. With the real-world samples and the rendered images we perform a hyperparameter search for CycleGAN and create several datasets where we vary the following parameters:

- Minimum defect size in rendered dataset: 0 px, 400 px, 1500 px, 2790 px
- Ratio of defective to defect-free images: 100:0, 75:25, 50:50
- Training epochs with fixed learning rate: 0, 2, 5, 50, 75, 100
- Training epochs with decreasing learning rate: 0, 2, 5, 50, 75, 100
- Batch size: 1, 32, 64
- Learning rate: 0.002, 0.0002, 0.00002

We evaluate the translated datasets in a similar way to the evaluation of the rendered data alone: we train the object recognition model on each translated dataset. The best performing translated dataset, validated on the 15% real-world samples, is then tested on the test dataset.

Finally, we investigate possible training strategies. First, we mix translated rendered images with the 85% real-world training samples and train with this dataset mixture. Next, we pre-train the object detection model with the translated rendered images and fine-tune the model with the real-world samples. The translated, synthetic data originates from the best performaning domain adaptation attempt. In this training strategy, which is common for object detection, the model learns how to extract basic features from the pre-training domain in the pre-training phases. The acquired knowledge is then transferred to the application domain, where the model is fine-tuned to improve the overall training success. This strategy is typically used when the training process needs to be accelerated, or when there is only a small amount of image data in the application domain.

The object detection model used in all experiments is a Faster R-CNN with a ResNet-50 backbone[16] pre-trained on MS COCO[17]. In all experiments we trained with a batch size of 8, a learning rate of 0.001, and for 25 epochs. The model from the epoch with the best performance on the validation dataset was used for the final test.

As a metric to evaluate the performance of the models on both on the validation and test datasets, we use mean average precision (mAP), which is the standard metric used in the literature to evaluate the performance of object recognition models. The metric first computed the intersection over union (IoU) of the model's bounding box estimate and the ground truth. The IoU is compared to a threshold. If the threshold is exceeded, the bounding box is counted as a true positive. False positives, true negatives, and false negatives are calculated accordingly based on the model output. Together they are used to determine precision and recall. We calculate precision and recall for different confidence levels to determine the precision-recall-curve. The average precision is the integrated area under this curve. To finally calculate the mAP we would need to average the average precision values for each class. Since we only have one class in our use case, mAP and average precision are the same. We determine two mAPs, the first with an IoU threshold of 0.5 (which is the primary test metric for Pascal VOC*) and the second averaged for different IoU thresholds from 0.5 to 0.95 with an iteration step of 0.05 (which is the primary test metric for MS COCO).

4.1 Results

The results of the final tests that each of the best performing models performed on the test dataset are shown in table 2. First, the model trained with translated data performed significantly better (mAP: 0.92) than the model trained with rendered data alone (mAP: 0.87). The performance of using only real-world data cannot be achieved with either rendered or translated image data (mAP: 0.97). When training with a mixture of real-world and translated data or pre-training with translated data and fine-tuning with real-world data the performance is on a par with real-world data alone and no significant increase can be seen.

Table 2: Performance results: mAP for the best performing models tested on 250 real-world test images.

	mAP@IoU=0.50	mAP@IoU=0.50:0.05:0.95
real-world	0.97	0.74
rendered	0.87	0.65
translated	**0.92**	**0.66**
real-world + translated	0.98	0.74
pre-train translated, finetune real-world	0.97	0.75

5. DISCUSSION

While the results indicate that the domain gap has been reduced, there remains a gap in performance compared to real-world samples. Complete closure of the domain gap was not possible. With the CycleGAN algorithm, the domain-adapted images show features of both the original source domain and the target domain. Fig. 1b

*Pascal VOC and MS COCO are commonly used benchmark and pre-training datasets for object detection tasks

shows examples of pairs of rendered and translated images from the best performing translated dataset, see Tab. 2. Qualitative improvements visible in the samples are:

- Imitation of the color, roughness and shading of the real component'a texture
- Imitation of real blur effects
- Imitation of the real illumination intensity and the real reflection behavior of the component's surface
- Imitation of real vignetting effect

The most noticeable change is the imitation of the real component texture features in the translated images. Both the color and the roughness of the real component surfaces have been imitated, with the roughness in the adapted images being significantly greater than in the rendered images. The shading caused by the roughness of the components in the real images was also imitated. In addition, a similar blurring of the component texture appears in the refined images, whereas the component surface is mush sharper in the synthetic images.

Furthermore, the illumination intensity, or the reflections caused by the illumination, have become stronger and more similar to the real-world samples. Another change is the vignetting of the translated synthetic image, which is typical for real images. A much stronger edge fall-off of the illumination can be seen, which results in a greater darkening of the domain-adapted images towards the edge of the image compared to the synthetic images.

The image modifications mentioned above are characteristic features of the domain adaptation from the synthetic to the real domain that took place in this work. Compared to these image modifications, the still existing domain gap is particularly evident in the representation of the defects. They largely retain their original reflectance behaviour and texture from the rendering domain. As for the defect texture, there is an increase in blur as in the rest of the image, but the basic structure of the texture remains unchanged. In addition, the shape of the defect has not changed. Only the color of the component surface has been projected onto the defect.

In general, the translated images lack sufficient domain adaptation of the defects in the synthetic images, making them appear unrealistic. This weakness of the CycleGAN algorithm for this application task is due to the lack of ability of the network to semantically align the defects in the synthetic images with those in the real images. However, it is hypothesized that this is not a CycleGAN-specific problem, but a more general problem of image-to-image translation based on unpaired datasets. To avoid this problem, a paired dataset, where 1-to-1 translations of the images from one domain to the other are available, or the additional use of defect annotations during the domain adaptation training process may be essential. In this way, a semantic link between the defect appearance image in both domains is provided to the algorithm.

In practice, the acquisition of a paired dataset usually requires a lot of effort or may not be possible at all. There are GAN-based approaches that allow hybrid training with paired and unpaired data. As an example, Tripathy et al.[18] present a hybrid approach that uses paired data to minimize the semantic swapping of features in the refined data, as is the case with hallucinations.

Furthermore, the domain-adapted images are prone to artifacts and overfitting induced hallucinations. The latter in particular is a major drawback for the defect detection task, as the detections lose precision and sensitivity due to hallucinations such as interference defects, as shown by the experimental design. The hallucinations are a consequence of the domain adaptation strategy adopted, in which the algorithm attempts to mimic features of the target domain and thus approximate the refined domain to the target domain. In the absence of external monitoring, this provides the model with insufficient constraints on the spatial arrangement of the imitated features in the refined images. As a result, hallucinations, such as interference defects, may occur in image regions that appear random. This problem could be solved however by manually re-annotating the translated data. However, this would undermine one of the main motivations for using rendering data in the first place, namely not having to manually and annotate a large number of training images.

While the translated images show an increase in performance compared to the rendered images, we do not see an increase in performance compared to the use of real-world data alone, or the combination of translated

images with real-world data as a blend or using fine-tuning with real-world data, see Tab. 2. However, we cannot conclude that using translated rendering data as pre-training data or in dataset blends is not useful at all for defect detection applications. In our use case at hand, the real-world samples already provides very good training data (mAP: 0.97), so an increase is hardly possible.

6. CONCLUSION AND OUTLOOK

As rendered images are increasingly used in place of or in addition to real-world data to train AI-based visual inspection models, the transferability of rendered-trained models to real-world applications is often limited by a domain gap. Recent advances in deep domain adaptation techniques promise to reduce the domain gap by using GANs to translate rendered images into the real-world domain. This promises to reduce the modelling effort required to set up the rendering pipeline for an industrial application, and to increase the model performance. While successful applications of deep domain adaptation have been demonstrated for translating rendering data for everyday objects or street scenes, its suitability for automated visual inspection in industrial quality control has remained uncertain due to challenges such as low data availability or low contrast images that are likely to be encountered in this domain.

The contribution of this work is to demonstratively apply a deep domain adaptation method, namely CycleGAN, to translate rendered images for a use case of visual inspection of cast iron parts with industrial borescopes and to discuss the results qualitatively and in addition quantitatively by using the translated images to train an object recognition model and validate the models performance on real-world test data.

Our qualitative results show that CycleGAN successfully imitates the real-world samples color, roughness, reflectance behaviour. It also imitates real illumination and vignetting effects. The translated images are difficult to distinguish from real-world samples, at least for a human observer. However, some of the defects remain unrealistic, as the real-world texture and the reflective behaviour are not fully translated. We attribute this to the lack of semantic coupling in unpaired domain adaptation.

Our empirical research shows that translated data is not only subjectively closer to the real world. Our models trained on translated data outperform models trained on rendered data only. However, we were not able to use the translated images to outperform models trained with real-world data only. This is mainly due to a remaining domain gap that CycleGAN could not fill.

We studied only one defect class with low intraclass variation. We would direct future research towards coupling rendering methods with deep domain adaptation for more challenging applications with more diverse defect classes or with fewer real-world training samples. Due to the excellent qualitative results for our use case, we would expect a corresponding increase in performance with translated data in more challenging use cases.

ACKNOWLEDGMENTS

This research was funded by the German Federal Ministry for Economic Affairs and Climate Action under grant number 20Q2109D.

REFERENCES

[1] Sun, B., Feng, J., and Saenko, K., "Return of Frustratingly Easy Domain Adaptation," *Proceedings of the AAAI Conference on Artificial Intelligence* **30** (Mar. 2016).

[2] Fulir, J., Bosnar, L., Hagen, H., and Gospodnetic, P., "Synthetic Data for Defect Segmentation on Complex Metal Surfaces,"

[3] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P., "Domain randomization for transferring deep neural networks from simulation to the real world," in [*2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*], 23–30 (Sept. 2017). ISSN: 2153-0866.

[4] Denninger, M., Sundermeyer, M., Winkelbauer, D., Olefir, D., Hodan, T., Zidan, Y., Elbadrawy, M., Knauer, M., Katam, H., and Lodhi, A., "BlenderProc: Reducing the Reality Gap with Photorealistic Rendering," in [*International Conference on Robotics: Sciene and Systems, RSS 2020*], (July 2020). ISSN: 2330765X.

[5] Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R., "Learning from Simulated and Unsupervised Images through Adversarial Training," in [*2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 2242–2251, IEEE, Honolulu, HI (July 2017).

[6] Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D., "Unsupervised Pixel-Level Domain Adaptation With Generative Adversarial Networks," 3722–3731 (2017).

[7] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A., "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in [*2017 IEEE International Conference on Computer Vision (ICCV)*], 2242–2251, IEEE, Venice (Oct. 2017).

[8] Schmedemann, O., Baaß, M., Schoepflin, D., and Schüppstuhl, T., "Procedural synthetic training data generation for AI-based defect detection in industrial surface inspection," *Procedia CIRP* **107**, 1101–1106 (2022).

[9] Csurka, G., "A Comprehensive Survey on Domain Adaptation for Visual Applications," in [*Domain Adaptation in Computer Vision Applications*], Csurka, G., ed., *Advances in Computer Vision and Pattern Recognition*, 1–35, Springer International Publishing, Cham (2017).

[10] Wang, M. and Deng, W., "Deep Visual Domain Adaptation: A Survey," *Neurocomputing* **312**, 135–153 (Oct. 2018). arXiv:1802.03601 [cs].

[11] Liu, M.-Y., Breuel, T., and Kautz, J., "Unsupervised Image-to-Image Translation Networks," in [*Advances in Neural Information Processing Systems*], **30**, Curran Associates, Inc. (2017).

[12] Lee, H.-Y., Tseng, H.-Y., Huang, J.-B., Singh, M., and Yang, M.-H., "Diverse Image-to-Image Translation via Disentangled Representations," 35–51 (2018).

[13] Toldo, M., Maracani, A., Michieli, U., and Zanuttigh, P., "Unsupervised Domain Adaptation in Semantic Segmentation: A Review," *Technologies* **8**, 35 (June 2020). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[14] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A., "Image-To-Image Translation With Conditional Adversarial Networks," 1125–1134 (2017).

[15] Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T., "CyCADA: Cycle-Consistent Adversarial Domain Adaptation," in [*Proceedings of the 35th International Conference on Machine Learning*], 1989–1998, PMLR (July 2018). ISSN: 2640-3498.

[16] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition," 770–778 (2016).

[17] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P., "Microsoft COCO: Common Objects in Context," (Feb. 2015). arXiv:1405.0312 [cs].

[18] Tripathy, S., Kannala, J., and Rahtu, E., "Learning Image-to-Image Translation Using Paired and Unpaired Training Samples," in [*Computer Vision – ACCV 2018*], Jawahar, C. V., Li, H., Mori, G., and Schindler, K., eds., *Lecture Notes in Computer Science*, 51–66, Springer International Publishing, Cham (2019).