On learning deep domain-invariant features from 2D synthetic images for industrial visual inspection

Abdelrahman G. Abubakr, Igor Jovančević, Nour Islam Mokhtari, Hamdi Ben Abdallah, Jean-José Orteu

On learning deep domain-invariant features from 2D synthetic images for industrial visual inspection

Abdelrahman G. Abubakr[a], Igor Jovančević[a,*], Nour Islam Mokhtari[a], Hamdi Ben Abdallah[b], and Jean-José Orteu[b]

[a]DIOTASOFT, 201 Pierre and Marie Curie Street, 31670 Labège, France
[b]Institut Clément Ader (ICA); Université de Toulouse ; CNRS, IMT Mines Albi, INSA, UPS, ISAE ; Campus Jarlard, F-81013 Albi, France

ABSTRACT

Deep learning resulted in a huge advancement in computer vision. However, deep models require a large amount of manually annotated data, which is not easy to obtain, especially in a context of sensitive industries. Rendering of Computer Aided Design (CAD) models to generate synthetic training data could be an attractive workaround. This paper focuses on using Deep Convolutional Neural Networks (DCNN) for automatic industrial inspection of mechanical assemblies, where training images are limited and hard to collect. The ultimate goal of this work is to obtain a DCNN classification model trained on synthetic renders, and deploy it to verify the presence of target objects in never-seen-before real images collected by RGB cameras. Two approaches are adopted to close the domain gap between synthetic and real images. First, Domain Randomization technique is applied to generate synthetic data for training. Second, a novel approach is proposed to learn better features representations by means of self-supervision: we used an Augmented Auto-Encoder (AAE) and achieved results competitive to our baseline model trained on real images. In addition, this approach outperformed baseline results when the problem was simplified to binary classification for each object individually.

Keywords: deep learning, domain adaptation, domain randomization, augmented autoencoders, self-supervision, synthetic rendering, 2D images, industrial visual inspection.

1. INTRODUCTION

Inspection and quality control are major tasks in modern industries. With more complex systems being developed, automation of the inspection process becomes crucial. It helps to increase production speed and decrease human error rates. Our research is tackling various problems of automating the process of visual industrial inspection. We develop algorithms that receive 2D images and provide a diagnostic on the state of mechanical assemblies. Whenever available, we exploit the Computer Aided Design (CAD) models of the assemblies. More precisely, we are dealing hereafter with a large family of industrial inspection problems, called *presence/absence* problem. The challenge is to verify the presence of assembled parts at the locations predefined by the CAD model of the assembly. Hence, we are looking to detect and raise an alarm if the part is absent or replaced by another part. Naturally, we are defining our problem as a multi-class classification problem. The use case we are using to validate our approach is a set of 3 specific different metallic supports in addition to a very diverse background class (Fig. 1). It is evident, however, that our approach is a general one, seamlessly applicable on any task of inspecting parts whose CAD models are available.

Typical assemblies we are aiming to inspect are airplane engines. In industries such as aircraft industry, target assemblies are rarely available for acquisition, hence the number of training images is limited. Due to the high variability of parts, some classes may have very small number of samples in the collected data. In addition, the acquired images will not be reusable for training other models in slightly different context. Finally, manually annotating target objects in real images is a laborious and time-consuming task.

CAD is a standard way to describe geometrical properties of mechanical assemblies. For each part, there is a 3D CAD model with the real dimensions. Therefore, rendering CAD models to generate synthetic training

*corresponding author: Igor Jovančević, igorjovan@gmail.com

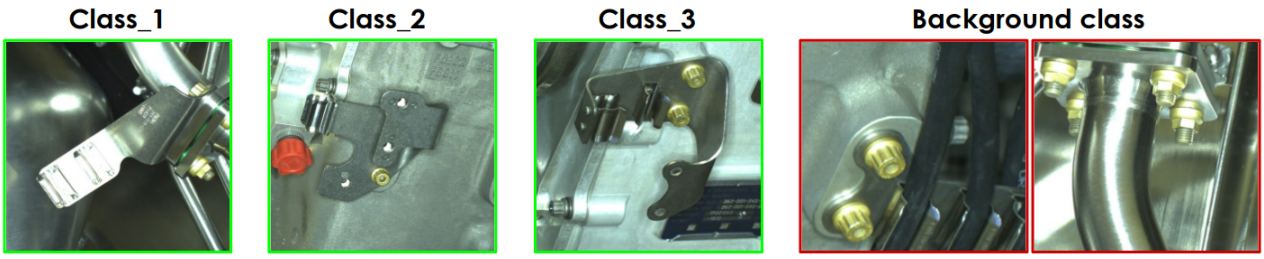

Figure 1. Samples of the 3 target objects used in this work, and the background class.

image data is an attractive approach which implicitly provides perfect annotations. However, industrial CAD models usually do not contain appearance properties, such as colour, texture, or the material properties of the objects. Additionally, there are often deformable or disposable parts like cables and plastic caps which are typically omitted in CAD models and present in real conditions.

Our goal is to explore the approaches to use 2D renders of such simplified 3D CAD models to train a Deep Convolutional Neural Networks (DCNN) for the purpose of visual inspection. The ultimate goal is to have a DCNN classification model trained on synthetic data only, and deployed to verify the presence of target objects in never-seen-before real images collected by RGB cameras from different poses and in different lighting conditions. The inspection process in this work is performed using a robotic arm equipped with a set of sensors mounted on an effector. More information on our hardware platforms and modes of operation is available in our paper.[1]

We train the classifier using unrealistic 2D renders of the simplistic 3D CAD models of our target objects. However, using synthetic data introduces a domain gap between features learned from synthetic data and those computed from real images. This is a non-trivial problem, and many approaches were proposed to overcome this issue by means of domain adaptation, domain randomization, domain generalization, and more[2–6] . This problem can be considered as a severe case of dataset bias; no matter how realistic the rendering might be, there will still be a difference between renders and real images, which will affect the learned features and prevent the model from generalizing knowledge on real images.

In this work, two approaches are adopted to solve the problem of domain gap between synthetic and real images. First, we try to narrow the domain gap by improving the synthetic data used for training. In Sec. 3.1, we introduce our rendering pipeline, and discuss the details of the domain randomization approach,[4,7] which is a key-stone in our solution. Second, we try to improve DCNN models and explore different ways of training to learn domain-invariant features that generalize well between synthetic and real domains. In Sec. 3.2 we try to understand the features learned by our classifier, and propose our approach to learn better features representations by means of self-supervision. Namely, we propose an Augmented Auto-Encoder (AAE), which enabled us to achieve results competitive to our baseline model trained on real images.

2. RELATED WORK

In our previous works, we were aiming to perform CAD based visual inspection by employing conventional image processing[1,8] and 3D point cloud processing techniques[9,10] as well as recent deep learning architectures on 3D point clouds.[11] In this work we are focusing on using DCNN on 2D images.

The use of synthetic data has a long history in computer vision. For example,[12] used renders of 3D CAD models as a source of labeled data. Due to the data-hungry nature of DCNN, synthetic rendering of 3D models is an attractive source of data in many applications, such as object recognition, detection, instance segmentation, and more.[5,6] An intuitive solution to the domain gap could be to generate realistic renders that are very similar to real images.[13] However, this approach is costly and requires full knowledge of the target domain.[4,5] In addition, even when having such expensive renders, trained model will still suffer from the domain gap.[4,6]

Many approaches were proposed to overcome the domain gap with non-realistic renders. For example,[5] used synthetic data to train object detection models for Pascal VOC classes, and investigated the importance of low level cues. The authors in[13] tried to have more photo-realistic rendering and simulated the context by using real images as background for their images. The authors in[6] freezed the backbone layers of Faster-RCNN pre-trained

on ImageNet, then trained the remaining layers of the detector with plain OpenGL[14] rendering. Using this trick, the features learned from target domain (real images) were used to train the model on source domain (synthetic images), which helped them to get competitive results to models trained on real images.

Another technique for domain adaptation is domain randomization.[4,7] It is based on training deep models on synthetic images that can transfer to real images directly. This can be done by randomizing rendering appearance parameters that describe the objects, such as colour, texture, lighting conditions, and more.[4,7] With enough variability in the rendered training data, the real world may appear to the model as just another variation of what it saw during training.[4] In addition to its simplicity, domain randomization achieved state of the art results for object localization and detection when training the models completely with synthetic data.[4]

3. METHODOLOGY

Fig. 2 shows the general pipeline of our approach. In testing phase, we use an RGB camera. The goal is to recognize each inspected object (Fig. 1) and either confirm its presence or report its absence (defect). We rely on an in-house-developed CAD-based localization method in order to estimate a relative pose of our camera with respect to the assembly. Knowing an approximate camera pose and camera intrinsic parameters, CAD model can be projected onto the image plane, which provides an expected region of interest (ROI) around the target object. Using this ROI and starting from a full image, we get a cropped image containing the area where the inspection item is expected to be found. This cropped image is then fed to a classifier to output a label for one of our target objects, or background which represents the fact that none of the known parts (classes) is recognised.

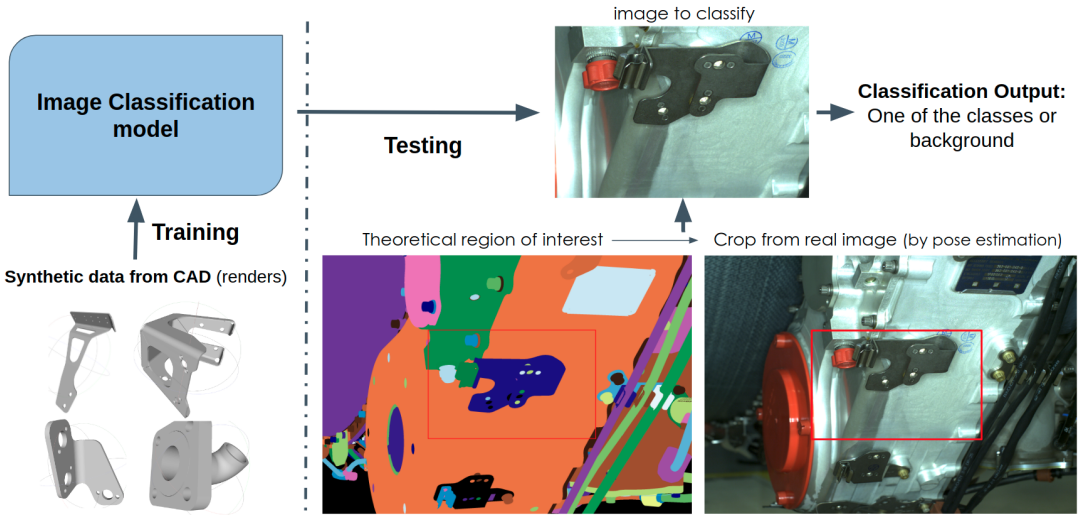

Figure 2. Multi-class classification based inspection pipeline.

3.1 Synthetic Data Generation

As our CAD models lack all appearance parameters, we adopted the method of domain randomization.[4,7] In order to apply domain randomization, a rendering pipeline is implemented using modern OpenGL that allows to control all variants of rendering parameters. Two shading models are used, Phong shading[15] and Cook-Torrance shading,[16] and we randomize the following parameters for each rendered object (Fig. 3).

- *Object colour:* Colour channels in OpenGL are scaled values ranging from 0 to 1. To better represent metallic material, we use "griesh colours". To generate griesh colours, the value of one channel is randomly sampled from the range $[0-1]$ let's call it R for the red channel, then the blue and green channels are randomly sampled so that they are within a distance of 0.1 from the R value, i.e. $B, G = R + rand(-0.1, 0.1)$.
- *Procedural texturing:* We generate random procedural textures. A procedural function takes a coordinate, and return a colour. For each object we randomly select one of the following textures: (1) mproved 3D Perlin noise (Simplex),[17,18] in which scale and orientations are also randomized, (2) random uniform colour disturbance (like salt and pepper noise), or (3) no texture.

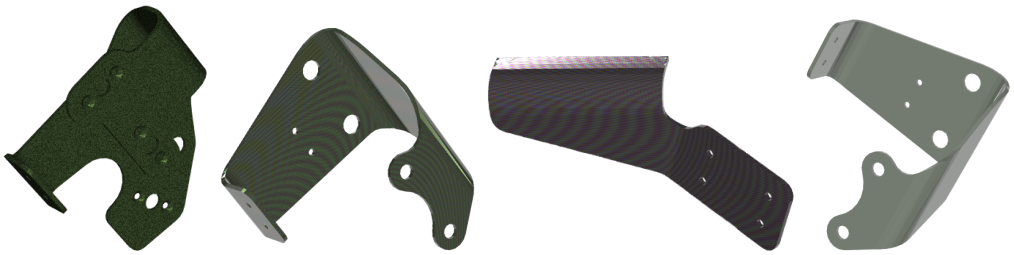

Figure 3. Samples of rendered objects after applying domain randomization.

- *Camera positions:* We sample the camera positions from a sphere around the target CAD to render. We fix the radius of the sphere r to 60 cm, and sample azimuthal angle θ and polar angle ϕ with certain step size.
- *Camera field of view (FOV):* This value represents the zoom level. A uniform random value is sampled from the range $[20 - 35]$, knowing that higher FOV value means the object is further (smaller).
- *Specular reflection coefficient:* This is the ratio of light reflecting from the surface. Uniform random value is sampled from the range $[0.1 - 0.5]$ for Cook-Torrance model, and range of $[0.5 - 1.0]$ for Phong model.
- *Shininess of the surface:* The higher the shininess value of an object, the more it properly reflects the light instead of scattering it all around. Uniform random value is sampled from the range $[2 - 16]$.
- *Fresnel complex coefficients:* Light reflecting off metallic surfaces is described by the Fresnel equations,[19] which are controlled by the complex index of refraction $\eta = n + ik$. In Cook-Torrance model, we randomly simulated metallic and non-metallic behaviours by controlling the values of n and k. n is randomly sampled from the range $[0.5 - 2.0]$, and k from the range $[0.5 - 7.0]$.

After generating the renders of target objects, we create a training dataset. Following,[6,7] we are randomizing the background by placing our renders on top of randomly selected real images. Those real images can be random captures from industrial environment, typically some photos from a factory with production line. They do not have to be directly related to our inspection object. In addition, using the idea of "flying distractors" introduced in,[7] we randomly position some of the rendered "context objects" around the rendered target objects. Fig. 4 shows samples of the generated synthetic dataset for training.

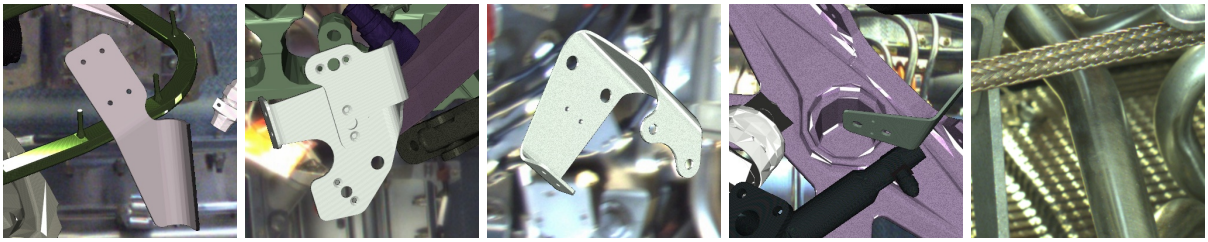

Figure 4. Samples of the synthetic dataset used for training. First 3 images contain rendered target objects, and the last 2 images are examples of the "background class", i.e. without our target objects.

3.2 Searching for Domain Invariant Features for Domain Generalization

The authors in[6] proved that freezing the backbone layers of Faster-RCNN pretrained on ImageNet worked well as a domain-invariant features and improved the final precision for object detection when training with plain OpenGL renders. In our use case, we applied the same approach but there was still a big domain gap as will be shown in Sec. 4. To find better features as initialization for our model, we need to understand the features learned when pretraining with ImageNet and why it is not good to represent our objects.

As DCNN are learning their weights directly from the huge amount of data seen during training, many researchers tried to understand what are these learned features through different layers. The work in[20] suggests that along different layers, networks seek to identify increasingly larger patterns in image, starting from simple edges and contours at early layers, and more complex shapes at deeper layers.[20] Recent works tried to investigate this assumption, specially the work in[21] which tried to answer an important question: how do neural networks

classify images; based on shape or texture? They came up with a simple experiment. Using style transfer,[22] they generated images with a texture different from the texture of the object in the image. For example, using the texture of an elephant projected on a cat image, all state-of-the-art classifiers recognize the image as "elephant", which clearly shows the bias of deep models to the texture, not the shape of objects.[21]

More recently, the work in[23] found that a network that models bag of local deep features had comparable results to state-of-the-art classification models, proving the findings in.[21] They even tested on set of scrambled images that are difficult for humans to recognize, and found that DCNN can still recognize scrambled images very well, which further proves the assumption that deep models do not recognize objects by their shapes.[21,23]

Auto-Encoders are family of neural networks for which the input is the same as the output. They work by compressing the input into a latent-space representation, then reconstructing the output from this representation. Denoising Autoencoders[24](DA) define a modified training process. Artificial random noise is applied to the input images while the target image stays clean. The main assumption is that DA produces latent representations which are invariant to noise because it facilitates the reconstruction of clean images.[25]

Using DA idea,[24] the authors in[25] introduced the concept of Augmented Auto-Encoders (AAE). Their goal was to learn 3D orientation of textureless objects for 6D object detection from RGB images. Using synthetic rendering and domain randomization, they generated a high variability of source images with extensive augmentation, while keeping clean images to be reconstructed. Their idea was to control what the latent representation encodes and which information to be ignored, forcing the model to learn geometrical transformation for the rendered objects, then apply it to real images to estimate the orientation of objects in real images.[25]

Inspired by the approach,[25] we adapted the idea of AAE for our goal to learn geometrical representation for mechanical assemblies. Fig. 5 shows the training process for our AAE. The input images are similar to what is used in classification, a rendered object on top of a random real image of a background. Output images contain only the object centered in the image with black background. Ideally, the features learned by the AAE can model the important geometrical features of the objects. Then using the features of the Encoder as initialization for training, the classifier can become discriminative enough. Therefore, it can recognize objects in real images depending on their shape and ignore all irrelevant texture and background information.

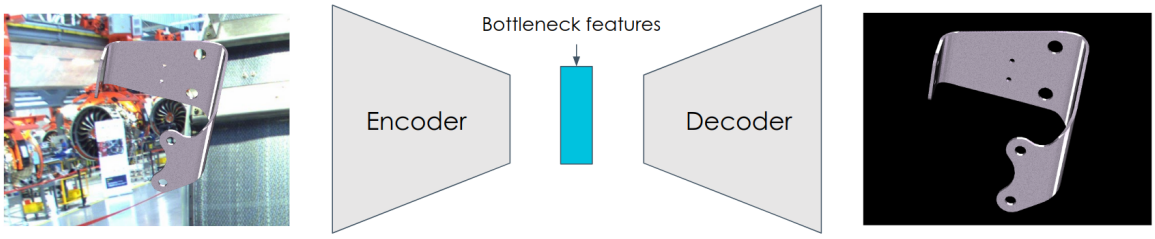

Figure 5. Our Augmented Auto-Encoder (AAE) to learn better geometrical features representation.

4. EXPERIMENTS AND DISCUSSION

We have only 179 real images collected for the 3 target objects with ROIs as shown in Fig. 2. To generate a test set for benchmark, the ROIs were used to crop the part of the image that contains the target object. To increase the number of samples, in addition to the original crop, 4 transformations are applied: horizontal and vertical flip, rotation 90 degrees clockwise and counter-clockwise. In addition, for each object, we crop 7 boxes with random positions and sizes to represent the "background" class. Hence, the total number of real images for testing is 2148. Finally, all crops are resized to 500x500. Fig. 1 shows samples of the generated test set.

In this work, InceptionV3[26] is used for classification. All experiments are done using Keras with Tensorflow backend. Training is done using Nvidia GeForce GTX 1060 with 6GB of RAM. Unless otherwise stated, SGD optimizer is used with learning rate of 0.0001 and momentum of 0.9 for classifier training, with batch size of 20 images, and training run for maximum 3 epochs. For AAE, InceptionV3 is used as the encoder, while for decoder we used a simple network consisting of successive convolutional and up-sampling layers that can reconstruct the image with the same input size. Training of AAE is done using Adam optimizer with learning rate 0.0003, batch size of 18 images and training run for 15 epochs. Input image size is set to 256x256 in all experiments.

4.1 Experiments

First, we get baseline results by training our 4-class classification model with real images with a 2-fold cross-validation. The real images data is split into 2 parts. The training is done with first split then testing with the other and vice versa. Then the results from the 2 splits are averaged to get final baseline results (1^{st} row in Table 1). This will be the reference to measure the gap between our baseline model trained on real images and models trained with synthetic images. In all our experiments, the evaluation metrics are Precision/Recall/F1-score, but F1-score will be used for comparison between different models. For the average results, we use the "macro" averaging between all classes, which calculates metrics for each label, and find their unweighted mean, so treating all classes equally regardless of their number of samples.

4.1.1 Freezing different layers of feature extractor

Following,[6] we tried freezing and training different stages of the pretrained InceptionV3. We tried freezing all layers, freezing all layers except last Inception block, freezing all layers except last 2 Inception blocks, and training all layers. In all cases, the last fully connected layers are trained. We use 55K synthetic training images rendered using Phong model. Table 1 shows the results of these experiments. We can see that the average F1-score is improving when training more Inception blocks, and the best result comes when all layers are trained allowing the model to adapt all its weights to the training data. These results are against the findings in,[6] and it shows that ImageNet pretrained weights are not good enough to describe our objects as discussed in Sec. 3.2.

Table 1. The results (Precision/Recall/F1-score) of training different stages of InceptionV3.

Training data	train/freeze	avg results	background	class_1	class_2	class_3
Real (2 splits)	train all	0.90 / 0.77 / **0.79**	0.87 / 0.99 / 0/92	0.87 / 0.46 / 0.53	0.99 / 0.74 / 0.82	0.86 / 0.90 / 0.88
Phong 55K	freeze all	0.42 / 0.39 / 0.28	0.99 / 0.19 / 0.31	0.11 / 0.27 / 0.15	0.43 / 0.33 / 0.38	0.16 / 0.75 / 0.26
Phong 55K	train last block	0.45 / 0.39 / 0.29	0.98 / 0.10 / 0.18	0.07 / 0.33 / 0.11	0.57 / 0.63 / 0.60	0.19 / 0.48 / 0.27
Phong 55K	train last 2 blocks	0.51 / 0.56 / 0.51	0.95 / 0.59 / 0.73	0.16 / 0.30 / 0.21	0.62 / 0.86 / 0.72	0.31 / 0.49 / 0.38
Phong 55K	train all	0.76 / 0.51 / 0.54	0.76 / 0.98 / 0.86	0.89 / 0.32 / 0.47	0.75 / 0.67 / 0.71	0.66 / 0.08 / 0.15

4.1.2 Training AAE and classifier with different synthetic data

To find the best configuration for AAE and the classifier, we experiment the effect of using different illumination models for rendering. In the first 3 rows of table 2, AAE is trained with 55K images of Phong renders, placed on top of real image crops with no rendered context in the background (flying distractors[7]). Using the Encoder network from AAE as the classifier, it is trained with synthetic data rendered with Phong, Cook-Torrance (55K images with flying distractors), and mix of both. We can see that using Cook-Torrance renders for training the classifier achieved the best F1-score (52%). However, using Phong renders combined with the context objects to train AAE (4^{th} row in Table 2), achieved much better F1-score (61%), showing the importance of flying distractors approach.[7] Last 3 rows in table 2 show the same experiments using ImageNet pretrained models, which confirm that Cook-Torrance resulted in a better overall F1-score, although the difference is small (1%).

Table 2. Effect of using different rendering models to train InceptionV3 pretrained with AAE and ImageNet.

Pretrain	Training data	train/freeze	avg results	background	class_1	class_2	class_3
AAE (Phong)	Phong	train all	0.65 / 0.38 / 0.40	0.70 / 0.98 / 0.81	0.43 / 0.16 / 0.23	0.51 / 0.31 / 0.38	0.95 / 0.09 / 0.16
AAE (Phong)	Cook-Torrance	train all	0.59 / 0.50 / 0.52	0.84 / 0.95 / 0.89	0.17 / 0.06 / 0.08	0.50 / 0.59 / 0.54	0.86 / 0.40 / 0.54
AAE (Phong)	Mixed	train all	0.64 / 0.45 / 0.48	0.75 / 0.97 / 0.85	0.38 / 0.13 / 0.19	0.51 / 0.41 / 0.46	0.92 / 0.29 / 0.44
AAE (Phong+cntxt)	Cook-Torrance	train all	0.79 / 0.56 / **0.61**	0.77 / 0.98 / 0.86	0.62 / 0.17 / 0.26	0.82 / 0.67 / 0.74	0.95 / 0.42 / 0.58
AAE (Cook-Torrance)	Cook-Torrance	train all	0.43 / 0.41 / 0.38	0.88 / 0.79 / 0.83	0.00 / 0.00 / 0.00	0.40 / 0.73 / 0.51	0.45 / 0.10 / 0.17
ImageNet	Phong	train all	0.76 / 0.51 / 0.54	0.76 / 0.98 / 0.86	0.89 / 0.32 / 0.47	0.75 / 0.67 / 0.71	0.66 / 0.08 / 0.15
ImageNet	Cook-Torrance	train all	0.78 / 0.51 / 0.55	0.78 / 0.97 / 0.86	0.93 / 0.14 / 0.24	0.69 / 0.66 / 0.67	0.72 / 0.29 / 0.41
ImageNet	Mixed	train all	0.79 / 0.49 / 0.50	0.79 / 0.97 / 0.87	0.94 / 0.16 / 0.27	0.65 / 0.74 / 0.69	0.80 / 0.09 / 0.16

4.1.3 Effect of number of training images

Results in Table 2 show that mixing the renders generated by both illumination models (110K images) decreased the performance. The intuitive explanation is that with more data, the model is overfitting and becoming more biased to the synthetic domain. To investigate this assumption, we trained the classification model pretarined with AAE for 1 epoch with 55K, 27K, and 15K images. The results in Table 3 show a big improvement when decreasing the number of training images. We can conclude that the weights learned by AAE are already good

enough and model our objects well, therefore training with more data (or for more epochs) will increase the domain gap while not improving the overall performance.

Table 3. Training the classifier on different number of images after pretraining with AAE

Pretrain	Training data	train/freeze	avg results	background	class_1	class_2	class_3
AAE (Phong+cntxt)	Cook-Torrance 55K	train all	0.79 / 0.56 / 0.61	0.77 / 0.98 / 0.86	0.62 / 0.17 / 0.26	0.82 / 0.67 / 0.74	0.95 / 0.42 / 0.58
AAE (Phong+cntxt)	Cook-Torrance 27K	train all	0.80 / 0.60 / 0.65	0.80 / 0.96 / 0.87	0.66 / 0.21 / 0.31	0.80 / 0.77 / 0.79	0.96 / 0.47 / 0.63
AAE (Phong+cntxt)	Cook-Torrance 15K	train all	0.79 / 0.66 / **0.70**	0.83 / 0.94 / 0.88	0.62 / 0.28 / 0.38	0.84 / 0.86 / 0.85	0.85 / 0.59 / 0.70

4.1.4 Overall results

Table 4 summarizes the best 4-class classification results achieved by different approaches. It is interesting to note that AAE itself is trained completely with synthetic data, and then using its weights, the classifier is also trained on synthetic data. This achieves our goal to train the model completely on synthetic data. Our best model achieved 70% F1-score which is 9% less than our baseline model trained on real images.

Table 4. Comparison between models trained on real images, and synthetic data with different pretraining strategies.

Pretrained weights	Training images	avg results	background class	class_1	class_2	class_3
ImageNet	Real (2 splits)	0.9 / 0.77 / **0.79**	0.87 / 0.99 / **0.92**	0.87 / 0.46 / **0.53**	0.99 / 0.74 / 0.82	0.86 / 0.90 / **0.88**
ImageNet	Synthetic (Cook-Torrance)	0.78 / 0.51 / 0.55	0.78 / 0.97 / 0.86	0.93 / 0.14 / 0.24	0.69 / 0.66 / 0.67	0.72 / 0.29 / 0.41
AAE (Phong+cntxt)	Synthetic (Cook-Torrance)	0.79 / 0.66 / 0.70	0.83 / 0.94 / 0.88	0.62 / 0.28 / 0.38	0.84 / 0.86 / **0.85**	0.85 / 0.59 / 0.70

Finally, we loosen our inspection task to a simple verification that an object is present. So we tested binary classification approach, i.e. training a separate model for each class. Each such binary model has a task to decide between class vs. background. The results in Table 5 show an advantage of the model trained with synthetic data with AAE pre-training over our baseline model in 2 out of 3 classes and also in overall score (3%).

Table 5. Results of binary classifiers (class vs. background) with synthetic images.

Pretrained weights	Training images	class_1	class_2	class_3	Overall
ImageNet	Real (2 splits)	0.94 / 0.55 / 0.56	0.95 / 0.87 / 0.89	0.99 / 0.97 / **0.98**	0.96 / 0.80 / 0.81
ImageNet	Synthetic (Cook-Torrance)	0.76 / 0.62 / 0.65	0.89 / 0.85 / 0.87	0.82 / 0.72 / 0.76	0.82 / 0.73 / 0.76
AAE (Phong+cntxt)	Synthetic (Cook-Torrance)	0.77 / 0.69 / **0.72**	0.90 / 0.92 / **0.91**	0.91 / 0.85 / 0.88	0.86 / 0.82 / **0.84**

5. CONCLUSION

This work was motivated by industrial needs for reference-based automatic visual inspection in a context where acquiring large amount of images is not practically feasible. To tackle the inspection challenge, we propose a solution that uses 2D renders of simplified 3D CAD models to train a DCNN multi-class classification model. The final objective is to deploy the trained model to recognize mechanical parts in never-seen-before real images.

To overcome domain gap problem, we adopted a two step approach. First, we implemented an OpenGL rendering pipeline which randomizes appearance features of rendered objects. Second, we proposed a novel method based on self-supervised learning that allows to learn better features representation for target objects. Based on the idea of AAE, the model can learn the most important features to represent target objects, while ignoring irrelevant information. By training on rendered data and testing on real data, we achieved F1-score comparable to the baseline model trained on real images. Moreover, when the multi-class problem was simplified to a binary classification, our models trained on synthetic data outperformed the baseline model by 3%.

REFERENCES

[1] Ben Abdallah, H., Jovančević, I., Orteu, J.-J., and Brèthes, L., "Automatic Inspection of Aeronautical Mechanical Assemblies by Matching the 3D CAD Model and Real 2D Images," *Journal of Imaging* **5**, 81–108 (10 2019).

[2] Csurka, G., "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint:1702.05374* (2017).

[3] Sun, B., Feng, J., and Saenko, K., "Return of frustratingly easy domain adaptation," in [*Thirtieth AAAI Conference on Artificial Intelligence*], (2016).

[4] 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, IEEE (2017).

[5] Peng, X., Sun, B., Ali, K., and Saenko, K., "Learning deep object detectors from 3D models," in [*Proc. IEEE Intl. Conf. on Computer Vision*], 1278–1286 (2015).

[6] Hinterstoisser, S., Lepetit, V., Wohlhart, P., and Konolige, K., "On pre-trained image features and synthetic images for deep learning," in [*Proc. European Conference on Computer Vision (ECCV)*], (2018).

[7] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S., "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in [*Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops*], 969–977 (2018).

[8] Jovančević, I., Larnier, S., Orteu, J.-J., and Sentenac, T., "Automated exterior inspection of an aircraft with a pan-tilt-zoom camera mounted on a mobile robot," *Journal of Electronic Imaging* **24**(6), 1–15 (2015).

[9] Jovančević, I., Pham, H.-H., Orteu, J.-J., Gilblas, R., Harvent, J., Maurice, X., and Brèthes, L., "3D point cloud analysis for detection and characterization of defects on airplane exterior surface," *Journal of Non Destructive Evaluation* **36**, 74 (12 2017).

[10] Ben Abdallah, H., Orteu, J.-J., Jovančević, I., Brèthes, L., and Dolives, B., "3D point cloud analysis for automatic inspection of complex aeronautical mechanical assemblies," *Journal of Electronic Imaging* **29**(04), 1–22 (2020).

[11] Mikhailov, I., Jovančević, I., Mokhtari, N., and Orteu, J.-J., "Classification using a three-dimensional sensor in a structured industrial environment," *Journal of Electronic Imaging* **29**(4), 1–14 (2020).

[12] Liebelt, J., Schmid, C., and Schertler, K., "Viewpoint-independent object class detection using 3d feature maps," in [*2008 IEEE Conference on Computer Vision and Pattern Recognition*], 1–8, IEEE (2008).

[13] Movshovitz-Attias, Y., Kanade, T., and Sheikh, Y., "How useful is photo-realistic rendering for visual learning?," in [*European Conference on Computer Vision*], 202–217, Springer (2016).

[14] "https://www.opengl.org/."

[15] Phong, B. T., "Illumination for computer generated pictures," *Communications of the ACM* **18**(6), 311–317 (1975).

[16] Cook, R. L. and Torrance, K. E., "A reflectance model for computer graphics," in [*ACM Siggraph Computer Graphics*], **15**(3), 307–316, ACM (1981).

[17] Perlin, K., "An image synthesizer," *SIGGRAPH Comput. Graph.* **19**, 287–296 (July 1985).

[18] Perlin, K., "Improving noise," in [*Proc. 29th Annual Conference on Computer Graphics and Interactive Techniques*], *SIGGRAPH '02*, 681–682, ACM, New York, NY, USA (2002).

[19] Gulbrandsen, O., "Artist friendly metallic fresnel," *Journal of Computer Graphics Techniques* (2014).

[20] Zeiler, M. D. and Fergus, R., "Visualizing and understanding convolutional networks," in [*European conference on computer vision*], 818–833, Springer (2014).

[21] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W., "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint:1811.12231* (2018).

[22] Gatys, L. A., Ecker, A. S., and Bethge, M., "Image style transfer using convolutional neural networks," in [*Proc. IEEE Conf. on computer vision and pattern recognition*], 2414–2423 (2016).

[23] Brendel, W. and Bethge, M., "Approximating cnns with bag-of-local-features models works surprisingly well on imagenet," *arXiv preprint:1904.00760* (2019).

[24] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A., "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research* **11**(Dec), 3371–3408 (2010).

[25] Sundermeyer, M., Marton, Z.-C., Durner, M., Brucker, M., and Triebel, R., "Implicit 3D orientation learning for 6D object detection from RGB images," in [*European Conference on Computer Vision (ECCV)*], (2018).

[26] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z., "Rethinking the inception architecture for computer vision," in [*Proc. of the IEEE conf. on computer vision and pattern recognition*], 2818–2826 (2016).