JEI Journal of Electronic Imaging

RESEARCH PAPER

Domain shared features for visual inspection of complex mechanical assemblies based on synthetically generated data

Velibor Došljak,[a,*] Igor Jovančević,[a] and Jean-José Orteu[b]

[a]University of Montenegro, Faculty of Natural Sciences and Mathematics, Podgorica, Montenegro

[b]Université de Toulouse, Institut Clément Ader (ICA), CNRS, IMT Mines Albi, INSA, UPS, ISAE, Albi, France

ABSTRACT. Even though neural network methodologies have been established for a long time, only recently have they achieved exceptional efficacy in practical deployments, predominantly due to improvements in hardware computational capacity and the large amounts of available data for learning. Nonetheless, substantial challenges remain in utilizing deep learning in many domains, mainly because of the lack of large amounts of labeled data that are versatile enough for deep learning models to learn useful information. For instance, in mechanical assembly inspection, annotating data for each type of mechanical part to train a deep learning model can be very labor-intensive. Additionally, it is required to annotate data after each modification of mechanical part specification. Also, the system for inspection is typically not available until the first few samples are built to collect data. This paper proposes a solution for these challenges in case of the visual mechanical assembly inspection by processing point cloud data acquired via a three-dimensional (3D) scanner. To reduce the necessity for manually labeling large amounts of data, we employed synthetically generated data for both training and validation purposes, reserving the real sensor data exclusively for the testing phase. Our approach reduces the need for large amounts of labeled data by using synthetically generated point clouds from computer-aided design models for neural network training. Domain gap is a significant challenge for the usage of synthetically generated data. To reduce the domain gap, we used different preprocessing techniques, as well as a neural network architecture that focuses more on shared features that will not significantly change between synthetically generated data and real data from the 3D sensor.

© 2024 SPIE and IS&T [DOI: 10.1117/1.JEI.33.3.031205]

Keywords: deep learning; conformity check; computer-aided design; synthetically generated data; domain gap; point cloud

Paper 231204SS received Nov. 1, 2023; revised Jan. 18, 2024; accepted Jan. 29, 2024; published Feb. 14, 2024.

1 Introduction

Quality control is one of the very important parts of modern industry, both in production and maintenance. Even before the rise of artificial intelligence, it was noticed that this process is prone to human error and required a lot of attention.[1] Much effort is being invested into automatizing this phase of the process. We developed a system for automatic visual conformity inspection of mechanical assemblies by processing data acquired by three-dimensional (3D) sensors and also using a computer-aided design (CAD) model of the assembly, as a reference. The goal is to check if an expected mechanical part is present in the assembly, and well mounted in a position

*Address all correspondence to Velibor Došljak, dosljakvelibor@gmail.com

1017-9909/2024/$28.00 © 2024 SPIE and IS&T

predefined by the CAD specification. This decision is made by processing a 3D point cloud obtained by scanning the region of interest on the assembly.

Training deep neural networks requires a large amount of labeled data, which is hard to obtain especially in industries such as aerospace industries. There are augmentation approaches to increase the training dataset's size and transfer learning techniques to gain satisfying results on the new case with a relatively small number of acquisitions in the training dataset. However, such techniques are not readily applicable in some cases. For example, annotating even a few point clouds for the segmentation task can be a tedious job. Also, performing many acquisition campaigns in large factories is not easily feasible, i.e., it is not practical to stop production to acquire a lot of data. For these reasons, our goal was to completely omit the necessity for collecting and annotating data and create a quality control system using just the CAD specification. Such synthetically generated point clouds can be automatically annotated without any human intervention.

Numerous deep learning architectures for processing 3D point clouds have been proposed.[2–4] They achieved very good results when data used for training and testing were independent and identically distributed (i.i.d). However, the performance of these models dropped significantly when trained on data obtained from one process (e.g., CAD models[5]), and tested on the data from the other process (e.g., Lidar scans[6]). Drop in performance is caused by differences in data distribution during training and testing, i.e., domain gap. Methods for overcoming the domain gap problem have been heavily studied in the field of two-dimensional (2D) images.[7,8] However, there have been very few works on domain generalization in the field of point clouds.

Formally speaking, if $\mathcal{X}$ is input space and $\mathcal{Y}$ is space of labels, domain is defined as tuple $(\mathcal{D}, g)$, where D is probability distribution on $\mathcal{X}$ and $g\colon\mathcal{X} \to \mathcal{Y}$ maps inputs to labels. In the task of overcoming the domain gap, we have two domains: source domain $(\mathcal{D}^{\mathcal{S}}, g^{S})$ and target domain $(\mathcal{D}^{\mathcal{T}}, g^{T})$. The goal is to train the model on source domain $\mathcal{D}^{\mathcal{S}}$, which is capable of predicting well on target domain $\mathcal{D}^{\mathcal{T}}$ when data from the target domain are not available in training time. It is assumed that label spaces $\mathcal{Y}^{S}$ and $\mathcal{Y}^{T}$ are shared between source and target domains. There are different setups. When samples from the target domain are known but not labeled, it is usually called domain adaptation. The case when there is no available information from the target domain is referred to as domain generalization. In our case, the input space is a set of point clouds with a fixed size. The source domain is represented by a set of synthetically generated point clouds and the target domain by a set of point clouds acquired via a 3D scanner.[9]

Considering the discrepancies between real-world and CAD representations, we proposed a model that focuses more on shared features between data from 3D sensor, and synthetic data generated from CAD models. Since we are inspecting solid mechanical parts, it is unlikely that they will differ significantly in CAD specifications and data acquired from the 3D sensor. However, significant differences can arise in the surrounding context. Examples of obvious causes of domain gap include the movement of cables and pipes in the real world; not all parts, ex. caps and protectors are represented in the CAD model. To make our algorithm focus on common features between synthetically generated data and real point clouds, we feed our classifier with features suitable for semantic segmentation.

2 Previous Work

With advancements in computer vision, there have been numerous applications in visual inspection systems. A lot of effort is put into computer vision-based systems for different kinds of defect detection. In the work of Huang et al.,[10] authors used R-CNN to extract information about the positions of mechanical parts in the image. Then, from the image segmentation results, the area, perimeter, circularity, and Hu invariant moment of the contour are extracted to form the feature vector. Finally, an SVM-based classifier was used to detect assembly defects. In the work of Ref. 11, authors used various traditional computer vision techniques for airplane exterior inspection and defects characterization. In the work of Mikhailov et al.,[12] PointNet[13] deep neural network was used to differentiate between different types of mechanical parts mounted on an assembly.

In the work of Boughrara et al.,[14] the authors carried out a comparison of multiple classification architectures such as PointCNN,[15] ShellNet,[16] DGCNN,[4] and PCNN[17] for the purpose of inspecting complex mechanical assemblies. The focus of their work was exploring the effectiveness of these models in detecting if the correct type of mechanical part is mounted.

It should be noted that the dataset used in their research is identical to the one used in our experiments. A distinctive aspect of their methodology was the training of the models on 61 types of mechanical parts, followed by the selection of the top-performing 20 classes for further examinations. This contrasts with our experiments where we used all 61 types throughout the training, validation, and testing phases of our experiments. Additionally, the authors employed tight crops around the mechanical parts in their experiments, whereas we used the entire point cloud as input to our neural network. While this makes our challenge more complex, it also reduces the need for a very accurate localization module. The authors demonstrated that among the compared models, PointCNN showed superior results compared to the other tested architectures, especially in terms of robustness to the differences in training and test data distributions caused by domain gap.

In work of Bolourian et al.,[18] a method is proposed for detecting defects on concrete surfaces using a model based on PointNet++,[19] focusing on point cloud data. Along with point coordinates, this study uses color and surface normal vectors, anticipating significant differences of these in areas with cracks, with the aim to enhance defect segmentation accuracy and reliability.

Learning from one domain and deploying in an unseen domain presents a slightly different challenge from the traditional machine learning setup. Typically, we assume that both the training and test datasets are i.i.d. However, when training a model on synthetically generated data, this assumption is often violated. Focusing on features shared between two domains can significantly improve the performance of a deep learning model. Several studies underscore the importance of focusing on specific features to increase model performance. Notably, most of these models have been applied to 2D images.

In the study from Abubakr et al.,[20] the authors generated synthetic 2D images of mechanical parts with random real background images from industrial settings. To produce these images, they experimented with various rendering algorithms. They trained an encoder-decoder architecture to extract and reconstruct the mechanical part without background from such images. The features extracted from the encoder were then used to distinguish between different types of mechanical parts through a classification network. The outcomes of this research were noteworthy and served as a significant inspiration for our work.

In the study by Božič et al.,[21] the authors put forth a method that combines segmentation mask learning with classification label learning for surface defect detection. They devised a novel loss function that merges the losses from both classification and segmentation. It is very important that their loss function can be used even when segmentation masks for all images are not available. Additionally, this loss function assigns different weights to the bordering pixels of a defect and the central pixels. This differentiation aims to reduce the impact of annotation errors around the defect's border.

In the study that demonstrated the work of Zhang et al.,[22] the authors introduced an architecture aimed at disentangling features common across multiple training domains from those specific to each domain. To improve performance, only the features shared across domains were used for task solving.

3 Data Preparation

Our dataset consists of 61 distinct mechanical parts. For each mechanical part, we have its CAD model and the CAD model of its surrounding context. Also, we have 3D point cloud scans from the assembly line. Each of these scans is annotated with the expected type of mechanical part and an additional label to indicate if the expected mechanical part is properly mounted or not. It is possible that the mechanical part is not mounted at all, that the wrong type of mechanical part is mounted, or that the correct type of mechanical part is mounted but there is an error in mounting, such as a flipped part, for example. Furthermore, we have an estimated transformation matrix that aligns the 3D sensor's coordinate frame with the CAD model's coordinate frame. While these matrices from the localization module do not guarantee perfect alignment, they are sufficient to know what we are looking at.

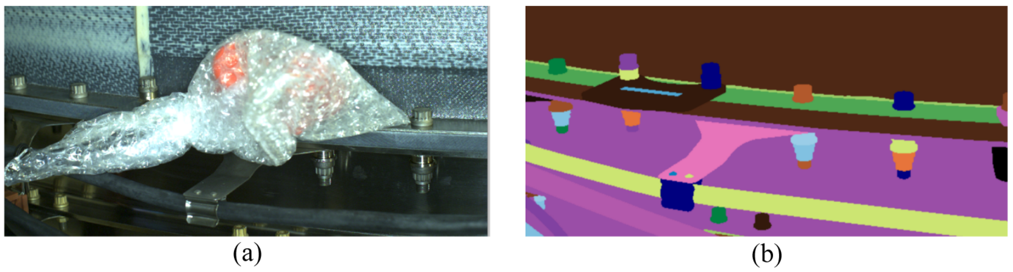
Fig. 1 (a) and (b) A comparison between the real-world scene and its corresponding CAD model representation.

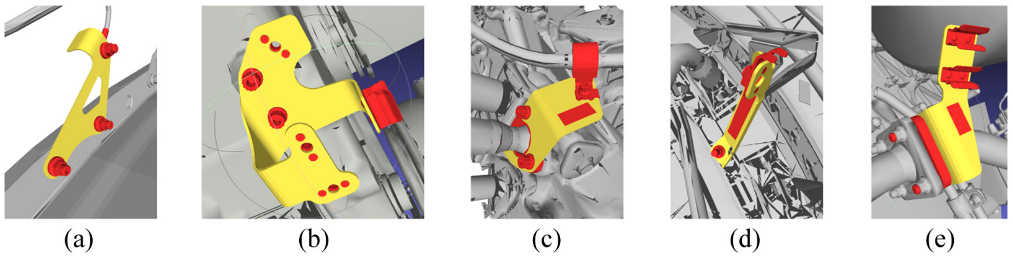
Fig. 2 (a)–(e) Examples of CAD models of the parts (supports) whose presence we aim to verify.

Figure 1 showcases a difference between real-world observations and CAD design. On the left, we notice an unexpected protective bag, absent in the CAD representation. This instance highlights one of the more challenging discrepancies. However, more commonly, differences might be the cable's shifted position or a missing protective cover.

Given our aim to use synthetically generated data for training and real sensor data for testing, it is very important to employ effective preprocessing measures to bridge the domain gap.

All our conclusions are based on experiments with data from an industrial use case – a visual inspection of a mechanical assembly from the aerospace industry, during production. Examples of the CAD models of the mechanical parts of interest are given in Fig. 2. All the tested parts belong to the type "metallic support."

3.1 Synthetic Data Generation

To generate synthetic point clouds, we employed the PyTorch3D[23] library. Since the library is implemented in PyTorch,[24] it can be used very efficiently on NVIDIA GPUs. For all the experiments and data generation, we used a machine with NVIDIA RTX 3060 with 12GB of VRAM.

First, we rendered the CAD model and then unprojected the rendered depth image to obtain the point cloud. In efforts to make the synthetically generated dataset more representative of different scenarios, we utilized varying crop radii for the point clouds: 250, 200, and 150 mm. This modification is very important, not solely due to the differences in crop sizes, but also as it helps in emulating varied point distributions. In a larger crop, points are much sparser compared to those in a smaller crop. This diversity, coupled with the strategy of random point sampling, is advantageous in enhancing the robustness of our model. Additionally, we incorporated random rotations and translations to the camera pose. All these simulations of different conditions add variability to the dataset, allowing the model to be more adaptable and versatile in processing inputs from the real 3D sensor.

Furthermore, we integrated random noise into the positions of the bounding box. This helps to simulate inaccuracies of localization algorithms in real-world scenarios, thus preparing the model to deal with potential inaccuracies in crop positioning.

3.2 Preprocessing Data from the 3D Sensor

It is very important to ensure that synthetically generated data are realistic enough for training neural networks. However, preprocessing the data sourced from the 3D sensor is equally important to achieve optimal results.

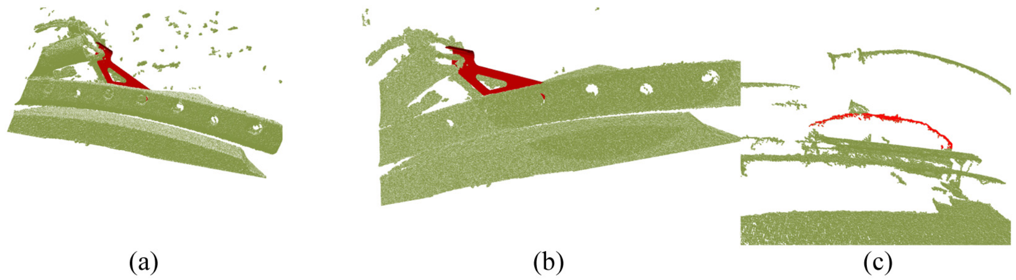
Fig. 3 Point cloud samples acquired via a 3D sensor. (a) Initial point cloud displaying prominent background noise. (b) Refined point cloud after outlier removal. (c) An instance where a non-optimal camera viewpoint renders the mechanical part as merely a line. Non-optimal camera viewpoint causes a difference in a small portion of information when the mechanical part is mounted and not mounted.

One of the primary challenges we encountered was the excessive noise in the point clouds from the sensor [Fig. 3(a)]. The initial step in preprocessing was to eliminate these outliers. The noise, which can be attributed to dust in the industrial setting, surface reflections, or sensor configurations, could reduce our model's accuracy, if not addressed.

Given the high density of the point cloud, containing more than 150,000 points, even the noise fragments in the background consist of numerous points, and their count varies substantially. To address this, we began by downsampling the point cloud to a more manageable resolution of 3000 points. After downsampling, it is expected that each noisy fragment in the background is represented by a single point. From this, we removed 952 points identified in low-density regions, retaining 2048. We name the remaining 2048 points, foundational points of the point cloud.

We expanded foundational points to our final point cloud, by integrating neighboring points. Occasionally, we noticed that certain elements, like wires, were not fully represented, prompting us to iteratively incorporate points until the gap between the retained and omitted points exceeded 1 cm. This method ensured the preservation of essential information while effectively filtering out noise.

Given certain scenarios, such as on Fig. 3(c), where the mechanical part of interest is represented with a small number of points, we decided on our point cloud expansion technique over the conventional radius-based or statistical outlier removal methods. By expanding the point cloud, we ensured the retention of not just large regions but also minor components directly linked to these large regions by some path.

In the final preprocessing step, before giving the point cloud to the input of the neural network, we employed the farthest point sampling (FPS) algorithm for downsampling. This approach aimed to align the distribution of input points between the synthetically generated data and the real data captured from the 3D sensor as closely as possible.

After experimenting with various point counts, we discerned that an optimal value is 2048 points. PointCNN classifier achieved the best results using 1024 points, and increasing the number of points to 2048 did not gain improvement in performance. With the architecture we proposed, a model trained on 2048 points obtained between 2% and 3% better accuracy than with 1024 points. This can be because we used a segmentation network as the backbone of our model, and most of these networks are built to work with 2048 points. But also in some cases, a small number of points can lead to the mechanical component of interest being underrepresented, which in turn causes small differences between scenarios where the part was mounted versus unmounted. Conversely, using too many points resulted in inefficiencies, lengthening the training process and necessitating larger kernel sizes or deeper network architectures to capture global information effectively. For example, using 4096 points instead of 2048 lengthens the training process three times while not gaining any substantial results improvement.

4 Experiments and Results

4.1 Baseline Results

Our final aim is to train on synthetic data and then test on real data. Yet, as an initial step, we trained and tested the model on real data to establish a foundational baseline. To do so, we

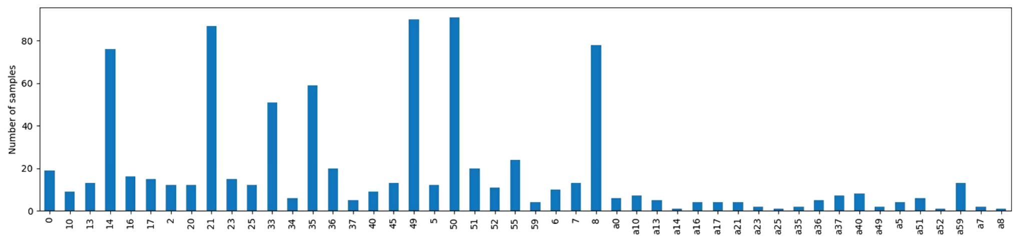

Fig. 4 Distribution of real point cloud data across various classes. Classes prefixed with “a” signify point clouds where the corresponding mechanical part is absent.

divided the real data into training and test sets, ensuring an 80-20 split for each type of support. It is very important to take a proportionally same amount of data from each class and not just randomly sample test data from the entire dataset because of varying sample sizes for different mechanical parts. Otherwise, it can happen that we do not have any data for one of the splits for some types of mechanical parts.

Following the findings from Boughrara et al.,[14] where the PointCNN[15] was highlighted as a standout architecture, we chose it as our primary backbone architecture. Structurally, our model mirrors the ModelNet40 classifier, with distinctions in the number of output classes and input data dimensions.

In terms of outputs, our neural network offers two output probabilities for each mechanical part: one output signals the presence of the part, whereas the other indicates its absence. With 61 different types of mechanical parts, we get a total of 122 classes. Given the very unbalanced dataset (Fig. 4), we used the macro average metrics, as defined in Ref. 25, to evaluate performance fairly across all classes.

It is important to note we do not have real acquisitions for all types of mechanical parts but just for classes represented in Fig. 4. Without such a metric weighting, a classifier that assigns all samples to the presence classes might achieve an impressive score, yet remain entirely ineffective. Notably, in our application, overlooking defective samples poses a greater problem than raising a false alarm.

After training, the model yielded a mean per class accuracy of 66%, precision of 0.61, a recall of 0.66, and an F1 score of 0.63. Here we used mean per class precision, recall, and F1 score just as in Scikit-learn.[25] The primary factor behind such results is the limited number of samples when the mechanical part is absent, making it challenging for the model to learn effectively. With just a few samples in some classes, it is very hard to train a model even with possibilities for augmentation. This limitation underscores the rationale for our inclination toward using synthetically generated data. Furthermore, this model performs perfectly on cases when a mechanical part is mounted, meaning there are not false alarms, but there is a significant percentage of missed samples. The model mostly mixes between presence and absence cases for the same mechanical part. This happens because of a small number of samples for training in the dataset for absence classes, and a small difference between the two point clouds. Context remains mainly the same, and the difference is in a small number of points [Fig. 3(c)].

4.2 Usage of Synthetic Training Data

Shifting our focus to synthetic data, we began with the identical model used in our experiment on real data. Notably, in this phase, we used the complete real dataset exclusively for testing and synthetically generated data for the training phase. This is in contrast to our earlier approach where we split the real data to be used for both training and testing purposes.

After training the model, we achieved a mean per class accuracy of 61%, precision of 0.56, a recall of 0.55, and an F1 score of 0.52. Just as with the model trained on real data, there was a clear tendency for the model to confuse the presence and absence classes for the same mechanical part. This observation suggests that the model gives more weight to the context surrounding the mechanical part rather than to the actual mechanical part we inspecting. To address this, several strategies could be employed. One such strategy is to zoom in more closely on the expected position of the mechanical part, similar to the approach taken by Boughrara et al.[14] However, this method presents its challenges as tighter cropping results in misclassification due to

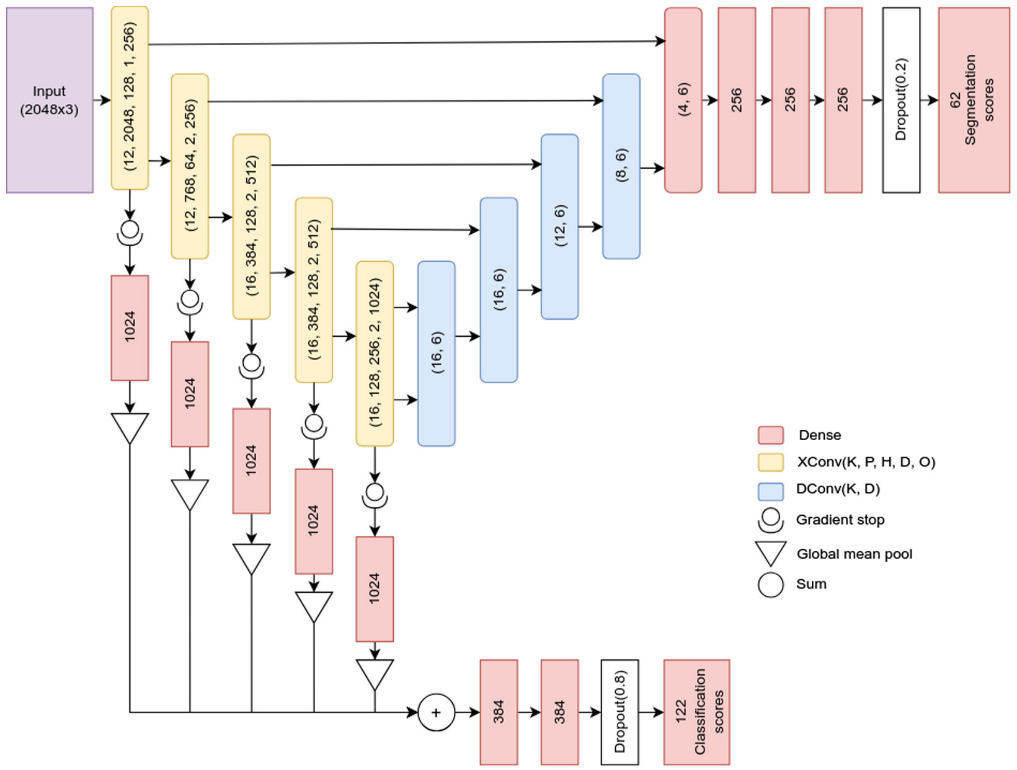

Fig. 5 Proposed model visualization. K—kernel size, P—number of points, H—hidden channels, D—dilatation, and O—output features.

inaccuracies in the localization module. In our work, we decided to push the model to learn more features specific to the mechanical part we are inspecting.

Since we inspect solid objects that cannot be deformed, it is much more frequent for surrounding contexts to differ in real-world and CAD representation. Focusing on the mechanical part of interest will also reduce the domain gap between real data and synthetically generated data.

An overview of our architecture is presented in Fig. 5. We aimed to align the features utilized for determining whether a mechanical assembly is correctly assembled with those used for segmentation. We believe that these features will encapsulate valuable information about the mechanical part under inspection, proving more beneficial than features learned by a classifier trained directly on raw point clouds.

Initially, we pretrained a segmentation model on synthetic data using the architecture proposed in the PointCNN paper.[15] Our primary objective was to extract features specific to the mechanical part, thereby minimizing the influence of the surrounding context. We hypothesized that the essential features for identifying the type of mechanical part would be captured within the outputs of the XConv layers in the encoder part of the segmentation network.

Initially, our approach centered solely on using features from the deepest XConv layer. However, this did not produce the expected improvements. This result is likely due to the U-Net-like architecture's data flow, where a significant amount of information can bypass the deepest XConv layer, proceeding directly to the deconvolution layers.

To address this challenge, we aimed to merge useful features from all XConv layers into a single feature vector. Given the varied output sizes from different layers, direct concatenation was not possible. Inspired by PointNet[13] architecture, we employed a global mean pool for each layer. To improve features obtained after pooling, we incorporated weighting adjustments through dense blocks. These blocks served to convert the output from each layer into a 1024-dimensional vector per point and reweigh features from XConv blocks. Following this, we executed a global mean pool and merged features from all layers in a global feature vector for the entire point cloud by summing the feature vectors derived from all XConv layers.

Finally, we utilized the derived global feature vector to determine classification scores. This was accomplished by deploying a fully connected classifier, composed of two dense layers, which then led to an output layer. Similar to the approach in Ref. 15, we incorporated a Dropout layer before the output layer. This addition was intended to improve the classification's resilience to potential discrepancies between features sourced from synthetically generated data and those from the 3D sensor.

For our loss function, we utilized cross entropy for both the segmentation and classification tasks. We first pretrained the segmentation network for 50 epochs to lay down a foundational feature set. After that, we trained both segmentation and classification segments of the network jointly for another 100 epochs. As the loss function, we used

$$L = L_{\mathrm{cls}} + \lambda L\mathrm{seg}, \tag{1}$$

$$L_{\mathrm{cls}} = L_{\mathrm{seg}} = -\sum_{i=0}^{C} t_i \log(p_i)). \tag{2}$$

Here, λ parameter is used to weigh between the impact of classification loss and segmentation loss, C represents the number of classes, $t_i = 1$ if a sample comes from class i else it is 0 and p_i represents probability assigned to class i. We evaluated models with different values for λ parameter between 0 and 1. Empirically, we found the optimal value of $\lambda = 0.2$. This can be explained by the fact that the segmentation network is already pretrained, but small fluctuations in features through learning can help to classification model to become more invariant. Also, it helps not to ruin too many features learned during the pretraining phase at the beginning of joint training due to the large learning rate.

The quality of the extracted features is intrinsically tied to the performance of the segmentation network on real data. Regrettably, we could not quantify this performance using metrics, as we lacked point-level annotations for the real data. However, the model achieved 98% overall accuracy and 97% mean per class accuracy on unseen synthetically generated data. Additionally, a visual assessment of its performance on real data suggests that it operates effectively in most cases.

After training the model on synthetically generated data, we obtained classification mean per class accuracy of 64%, precision 0.57, recall: 0.53, and F1 score 0.53 on the real data. As we can see, we obtained significant improvement in mean per class accuracy, and slight improvements in precision and F1 score. This can be explained by significantly better results in the case when the mechanical part is not mounted.

To ascertain whether our proposed enhancements would be compatible with other architectures, we also evaluated them using the ShellNet[16] backbone while retaining our classification part of the network.

When the ShellNet classifier was trained directly on raw point clouds, it yielded a mean per class accuracy of 0.58, precision of 0.53, recall of 0.5, and an F1 score of 0.5. Notably, these baseline results with ShellNet are marginally inferior to those observed with the PointCNN baseline, which aligns with the findings of Boughrara et al.[14]

However, upon applying our methodology-leveraging features learned by the segmentation network for classification. We observed a mean per class accuracy of 0.67, precision of 0.60, recall of 0.57, and an F1 score of 0.56. It seems that ShellNet is more suitable for our adaptation. This improvement may be attributed to its superior segmentation capabilities, which in turn produce enhanced features. Some examples of ShellNet segmentations were given in Fig. 6.

For our approach, it would be ideal if all the input point clouds would be like the one in Fig. 6(a), with a single mechanical part of interest present in the point cloud. Figure 6(a) shows a sample where our method is capable of perfectly segmenting the part. Such segmentation results provide good features for further analysis (classification). Segmentation results like Figs. 6(b) and 6(c) are still useful because they give us good information about where the classifier should pay attention. Our approach was created upon prior that on each point cloud only one type of mechanical part was visible, but as we can see from Fig. 6(d), there are some cases where two mechanical parts of interest are visible. Even though each point cloud was labeled with a single label, there are such cases where more than one mechanical part of interest is visible, which will cause additional confusion for our model when evaluating it on synthetically generated data.

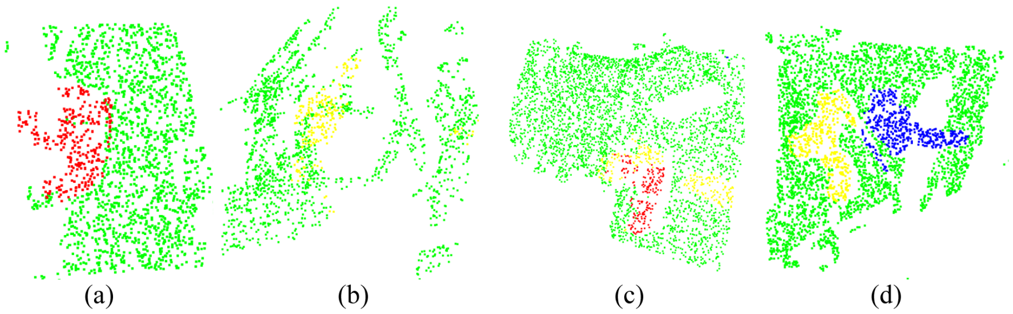

Fig. 6 Examples of ShellNet segmentations on real point clouds: (a) Clear and accurate segmentation. (b) Competent segmentation but with ambiguous borders. (c) The segmentation model mixes classes of two mechanical parts but the overall shape is correct. (d) Point cloud where two different mechanical parts are visible.

Even though in our dataset we did not find too many such samples, it should be noted that the classification approach is not the best one in such cases. For such cases, making tighter crops would be useful.

4.3 Results Overview

The summary of the results from our experiments is presented in Table 1. Our approach has successfully enhanced results across nearly all metrics when compared to classifiers trained solely on synthetic data, for both PointCNN and ShellNet models.

From the results, we notice that our model managed to improve the results compared to the classifier trained on synthetic data, which testifies to the success of our approach for reducing the domain gap. We even managed to get results close to the results obtained by the model trained on real data. This fact, combined with the significantly reduced need for human resources in our approach, is very motivating for further investment in the direction of usage of synthetically generated data.

There have been a few works on our particular use case. There is work of Boughrara et al.,[14] where authors used the same dataset of CAD models to generate synthetic data. Their approach differs from ours because it relies significantly on the localization module to make tight crops around the mechanical part, which is disadvantageous and risky. Also in the testing phase, they used only 20 classes and reported overall accuracy and weighted measurements of precision, recall, and F1 score, which in our opinion is not the best way of measuring performance in unbalanced datasets. To make a fair comparison, we also report weighted metrics on the same subset of classes. A comparison with our approach is presented in Table 2. We can see that even though our model does not rely on accurate localization, we managed to improve almost all metrics. Improvements were especially high in cases where the mechanical part is mounted as our model classifies most of such cases without errors.

Table 1 Summary of experimental results.

Training data	Model	MPA	Precision	Recall	F1	Throughput (FPS)	Support
Real	Cls PointCNN	0.66	0.61	0.66	0.63	112	177
Synthetic	Cls PointCNN	0.61	0.56	0.55	0.52	112	887
Synthetic	Seg+Cls PointCNN	**0.64**	**0.57**	0.53	**0.53**	64	887
Synthetic	Cls ShellNet	0.58	0.53	0.5	0.5	116	887
Synthetic	Seg+Cls Shellnet	**0.67**	**0.60**	**0.57**	**0.56**	77	887

Note: Bold values indicate instances where our method outperformed the baseline approach of training classifiers directly on raw point clouds. MPA, mean per class accuracy; precision - macro averaged precision between classes, Recall - macro averaged recall between classes, F1 - macro averaged F1-score between classes, throughput (FPS) - measured on Nvidia RTX 3060, Support - number of samples used for testing.

Table 2 Comparison with results from work of Boughrara et al.[14]

Dataset	Model	Precision	Recall	F1
Absence classes cropped	Model[14]	0.65	0.55	0.54
Absence classes	Seg+Cls Shellnet (ours)	**0.71**	0.5	**0.57**
Presence classes cropped	Model[14]	0.92	0.49	0.55
Absence classes	Seg+Cls Shellnet (ours)	**0.95**	**0.88**	**0.91**

Note: Precision - weighted averaged precision between classes, Recall - weighted averaged recall between classes, F1 - weighted averaged F1-score between classes.

5 Conclusions

We proposed an original solution for reducing the domain gap between synthetically generated point clouds from the CAD model and real point clouds from the 3D scanner in the use case of inspection of mechanical assemblies.

We generated realistic synthetic data and adapted the existing point cloud classification model so it can leverage segmentation masks available at no effort with synthetic data. By doing so we succeeded in fully automating the pipeline for the inspection of complex mechanical assemblies and removed the necessity for labeling data by humans while getting results comparable to results obtained by a model trained on real labeled data. This is a very important feature because it gives the possibility to have an inspection model even before the first defect occurs while reducing the work required by humans.

In addressing the domain generalization challenge, our findings suggest that improving performance can be achieved by directing the learned features toward those that are shared across domains. This alignment can be facilitated using a segmentation network to learn characteristic features, and later using those features for training a classifier.

However, it should be noted that even our best model does not yet achieve satisfying results for practical applications in the industrial environment, and it will require significant further improvements.

6 Future Work

We are considering including some real data in training the models but we are seeking a way to obtain their annotations as easily as possible. Therefore, we plan to first run a segmentation model trained on synthetically generated data, on real point clouds. Further, we can iteratively add real samples with good segmentations selected visually by humans, to improve the model when tested on real data point clouds. At the end of each iteration, we will get more and more usable segmentation masks of real clouds, which will require just small human corrections for completely accurate annotations.

In our pursuit of further improving domain generalization for our specific task, we intend to fuse the methodology proposed in this paper with domain randomization. A noticeable problem for our models, especially with PointCNN, is its occasional misclassification when real-world objects, not represented in the CAD specification, come into contact with the mechanical part under inspection.

To address this, we will use data augmentation at various intensities. One approach involves complete context augmentation; this would involve positioning mechanical parts in slots typically reserved for other components. We will get a mechanical part, a context from the other mechanical part, and fuse them. After that, we will use our method for generating synthetic data to get labeled data with highly randomized contexts. By doing so, we can get almost infinitely many contexts. Through this, the segmentation network is expected to focus more on focusing in on features that distinguish the various mechanical parts of interest, without focusing too much on changes in context around it. A slightly less intensive method would be the random addition of augmenting objects, such as pipes or planes, within the established CAD specifications. In this approach, we will keep the same context, just augment it by adding different artifacts to it. The basic idea is pretty much the same as in the case with a completely changing context, just in this case it will not be so aggressive, so it might be easier for the network to learn.

Code and Data Availability

The data utilized in this study are coming from DIOTA company, i.e., from the aerospace industry, and are sensitive and confidential. Consequently, the data are not available for sharing. Code is available from the authors upon request and with permission from the DIOTA company.

Acknowledgments

The authors express their gratitude to the DIOTA company, for providing the data and their industrial insight, both valuable for this research work. DIOTA has its Head Office at 201 Pierre and Marie Curie Street, 31670 Labège, France. The authors would like to thank the French “Région Occitanie” for its financial support to the joint research laboratory “Inspection 4.0” between IMT Mines Albi/ICA and the DIOTA company.

References

1. B. S. Dhillon, “Human error in aircraft maintenance,” *Human Reliability and Error in Transportation Systems*, pp. 131–144, Springer, London (2007).
2. H. Thomas et al., “KPConv: flexible and deformable convolution for point clouds,” in *IEEE/CVF Int. Conf. Comput. Vision (ICCV)*, pp. 6410–6419 (2019).
3. H. Zhao et al., “Point transformer,” in *IEEE/CVF Int. Conf. Comput. Vision (ICCV)*, pp. 16239–16248 (2021).
4. Y. Wang et al., “Dynamic graph CNN for learning on point clouds,” *ACM Trans. Graph.* **38**(5), 12 (2019).
5. Z. Wu et al., “3D shapenets: a deep representation for volumetric shapes,” in *IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 1912–1920 (2015).
6. A. Dai et al., “ScanNet: richly-annotated 3D reconstructions of indoor scenes,” in *IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 2432–2443 (2017).
7. K. Muandet, D. Balduzzi, and B. Schölkopf, “Domain generalization via invariant feature representation,” in *Proc. 30th Int. Conf. Mach. Learn.*, Vol., 28, pp. 10–18 (2013).
8. M. Ghifary et al., “Domain generalization for object recognition with multi-task autoencoders,” in *IEEE Int. Conf. Comput. Vision (ICCV)*, pp. 2551–2559 (2015).
9. X. Wei, X. Gu, and J. Sun, “Learning generalizable part-based feature representation for 3D point clouds,” in *Adv. in Neural Inf. Process. Syst.*, S. Koyejo et al., Eds., Curran Associates, Inc., Vol. 35, pp. 29305–29318 (2022).
10. H. Huang, Z. Wei, and L. Yao, “A novel approach to component assembly inspection based on mask r-CNN and support vector machines,” *Information* **10**(9), 282 (2019).
11. I. Jovancevic et al., “3D point cloud analysis for detection and characterization of defects on airplane exterior surface,” *J. Nondestruct. Eval.* **36**, 74 (2017).
12. I. Mikhailov et al., “Classification using a three-dimensional sensor in a structured industrial environment,” *J. Electron. Imaging* **29**(4), 041008 (2021).
13. C. R. Qi et al., “PointNet: deep learning on point sets for 3D classification and segmentation,” in *IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 77–85 (2017).
14. A. Boughrara et al., “Inspection of mechanical assemblies based on 3D deep learning approaches,” *Proc. SPIE* **11794**, 1179407 (2021).
15. Y. Li et al., “PointCNN: convolution on χ-transformed points,” in *Adv. Neural Inf. Process. Syst.*, S. Bengio et al., Eds., Curran Associates, Inc. (2018).
16. Z. Zhang, B.-S. Hua, and S.-K. Yeung, “ShellNet: efficient point cloud convolutional neural networks using concentric shells statistics,” in *IEEE/CVF Int. Conf. Comput. Vision (ICCV)*, pp. 1607–1616 (2019).
17. M. Atzmon, H. Maron, and Y. Lipman, “Point convolutional neural networks by extension operators,” *ACM Trans. Graph.* **37**(4), 1–12 (2018).
18. N. Bolourian et al., “Point cloud-based concrete surface defect semantic segmentation using modified pointnet++,” in *Proc. 29th EG-ICE Int. Workshop on Intell. Comput. in Eng.* (2022).
19. C. R. Qi et al., “PointNet++: deep hierarchical feature learning on point sets in a metric space,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Curran Associates Inc., Red Hook, New York, pp. 5105–5114 (2017).
20. A. G. Abubakr et al., “Learning deep domain-agnostic features from synthetic renders for industrial visual inspection,” *J. Electron. Imaging* **31**(5), 051604 (2022).
21. J. Božič, D. Tabernik, and D. Skočaj, “Mixed supervision for surface-defect detection: from weakly to fully supervised learning,” *Comput. Ind.* **129**, 103459 (2021).

22. H. Zhang et al., “Gated domain-invariant feature disentanglement for domain generalizable object detection,” arXiv:2203.11432 (2022).
23. N. Ravi et al., “Accelerating 3D deep learning with pytorch3D,” arXiv:2007.08501 (2020).
24. A. Paszke et al., “PyTorch: an imperative style, high-performance deep learning library,” in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, Curran Associates Inc., Red Hook, New York, 12 (2019).
25. F. Pedregosa et al., “Scikit-learn: machine learning in Python,” *J. Mach. Learn. Res.* **12**(85), 2825–2830 (2011).

Velibor Došljak is a teaching assistant at the University of Montenegro. He received his BS and MS degrees in mathematics and computer science from the University of Montenegro in 2019 and 2022, respectively. The work in this paper was done during his master’s thesis internship with Diota in 2021. He is currently working on his PhD on industrial applications of deep learning for of 3D point clouds.

Igor Jovančević graduated from the computer science program of the Faculty of Natural Science and Mathematics at the University of Montenegro with a mathematics degree in 2008. He graduated in 2011 from a joint Erasmus Mundus Master program in Computer Vision and Robotics (VIBOT) conducted by the University of Burgundy, University of Girona, and Heriot-Watt University. He received his PhD in computer vision in 2016 from IMT Mines Albi, a French “Grande Ecole” specializing in process engineering. He worked at Diotasoft, as a research engineer focusing on computer vision applications on the problems of industrial inspection and manufacturing process monitoring. He is currently working as an assistant professor at the University of Montenegro, focusing on applying computer vision to solve real-world use cases.

Jean-José Orteu graduated in 1987 from a French “Grande Ecole” (ENSEIRB, Bordeaux, France) with an engineering degree in electrical and software engineering and a master’s thesis in automatic control. He received his PhD in computer vision in 1991 from Université Paul Sabatier (Toulouse, France). Currently, he is a full professor at IMT Mines Albi (Albi, France), a French “Grande Ecole” specialized in process engineering. He carries out his research work in the Institut Clément Ader (ICA) laboratory (200 people). For more than 15 years, he has developed computer-vision-based solutions for 3D measurements in experimental mechanics (PhotoMechanics) and, since 2010, he is more specifically involved in the application of computer vision to NDE, inspection, and manufacturing process monitoring.