Automatic visual inspection of mechanical assemblies via 3Dpoint cloud classification with deep neural networks

Assya Boughrara
Institut Clément Ader (ICA) ; Université de Toulouse ; CNRS, IMT Mines Albi, INSA, UPS, ISAE
Igor Jovančevic
Faculty of Natural Sciences and Mathematics, University of Montenegro
Velibor Dosljak (velibord@ucg.ac.me)
Faculty of Natural Sciences and Mathematics, University of Montenegro
Jean-José Orteu
Institut Clément Ader (ICA) ; Université de Toulouse ; CNRS, IMT Mines Albi, INSA, UPS, ISAE
Mathieu Belloc
Diota

Research Article

Keywords: aeronautics, robotized inspection, quality control, CAD model, deep learning, 3D point cloud, classification

Posted Date: January 31st, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3897234/v1

Additional Declarations: No competing interests reported.

Automatic visual inspection of mechanical assemblies via 3D point cloud classification with deep neural networks

Assya Boughrara[1], Igor Jovančević[2], Velibor Došljak[2*], Jean-José Orteu[1], Mathieu Belloc[3]

[1]Institut Clément Ader (ICA) ; Université de Toulouse ; CNRS, IMT Mines Albi, INSA, UPS, ISAE ; Campus Jarlard, 81013 Albi, France.
[2]Faculty of Natural Sciences and Mathematics, University of Montenegro, Cetinjska 2, 81000 Podgorica, Montenegro.
[3]Diota, 201 Pierre and Marie Curie Street, 31670 Labège, France.

*Corresponding author(s). E-mail(s): velibord@ucg.ac.me;
Contributing authors: assya.boughrara@mines-albi.fr; igorj@ucg.ac.me;
jean-jose.orteu@mines-albi.fr; mbe@diotasoft.com;

Abstract

In this work, we are focused on conformity control of complex aeronautical mechanical assemblies, typically an aircraft engine, at the end or in the middle of the assembly process. A 3D scanner carried by a robot arm provides acquisitions of 3D point clouds which are further processed by deep classification neural networks. The Computer-Aided Design (CAD) model of the inspected mechanical assembly is available, and the proposed approach relies on it. The models are trained on synthetic data, generated from the CAD models.

Keywords: aeronautics, robotized inspection, quality control, CAD model, deep learning, 3D point cloud, classification.

1 Introduction

Automatic control through artificial vision is a way to relieve human operators from repetitive visual inspection tasks and to ensure faster and more reliable quality control.

The Computer-Aided Design (CAD) model serves as a reference for the inspection process. This digital 3D model also acts as a guide for the inspection and provides prior knowledge about the mechanical element that is currently being inspected. The proposed algorithm is supposed to verify the presence and the correct mounting of various types of elements used to secure cables, pipes, or other components of mechanical assembly. We refer to these elements as *brackets*, i.e. metal parts of different dimensions, primarily used to hold other elements together (see Fig. 1).

The final objective is a robotized inspection of complex aeronautical mechanical assemblies, typically an aircraft turbine, for which we have the digital CAD model. The assembly can be fully assembled or still on the assembly line. The challenge is to verify that the elements constituting the assembly are present and in the right positions, as specified by the CAD model. Conformity is defined by the actual presence of the expected mechanical

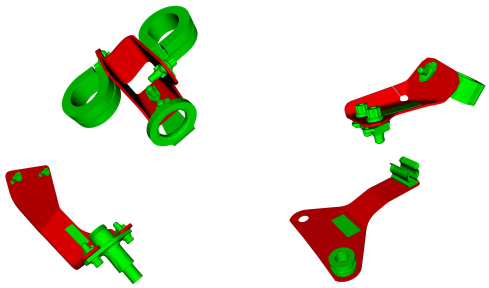
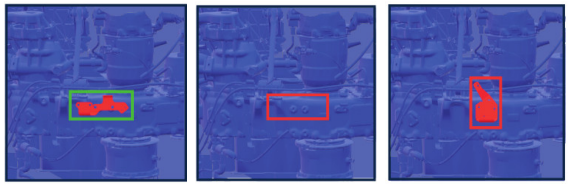


Fig. 1: Some examples of brackets to check (CAD models)

part in the assembly. A proposed inspection module must be able to differentiate a compliant case from any other possible deviation, such as:

1. no bracket is present
2. different bracket than the expected one is present

Examples of three possible scenarios for bracket parts are illustrated as CAD model 2D renders in Fig. 2.



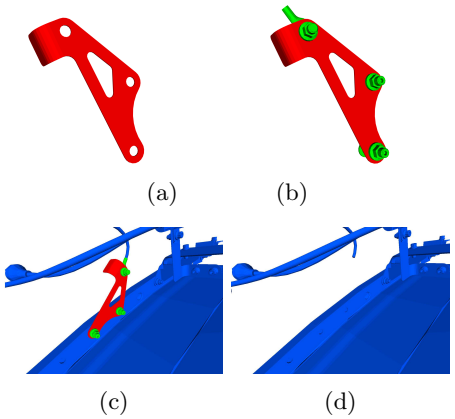
(a) (b) (c)

Fig. 2: Examples of different possible scenarios. (a) Correctly mounted mechanical part. (b) Missing mechanical part. (c) Incorrectly mounted mechanical part.

We propose the exploitation of the CAD model to generate a training dataset for deep classification neural networks. The main goal is to come up with a protocol allowing the training of a model exclusively on synthetic 3D point clouds generated from the CAD model while guaranteeing a good prediction performance on real 3D point cloud data acquired by a 3D scanner in a real scenario. Real scenarios could be challenging for several reasons. Namely, besides typical difficulties coming from varying lighting and domain gap, some non-rigid elements of the assembly, such as cables, might slightly change places and produce unexpected occlusions in real point clouds. Furthermore, disposable elements such as caps, are not present in the CAD model.

We got CAD data for 61 types of mechanical parts. When generating the synthetic data, we can use different sizes of the surrounding context. Through empirical analysis, we will select the optimal area around the bracket. We define several reference terms for different kinds of information contained in CAD model:

- **bracket** - mechanical part we are interested in (Fig. 3a)
- **bracket-fusion** - bracket together with elements that connect it to the rest of the mechanical assembly (Fig. 3b)
- **bracket-fusion with wider context**: includes the wider surrounding (context) in which the bracket-fusion is integrated, (Fig. 3c).
- **context alone** - context of the mechanical part of interest, without the part itself (Fig. 3d). This level of information will be useful for generating synthetic data for the cases when the mechanical part is not mounted.



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

Fig. 3: Examples of CAD models : (a) example of bracket alone (in red), (b) bracket and its closest surrounding, nicknamed *bracket-fusion*, (c) *bracket-fusion* within its context, (d) context alone. The bracket of interest is marked in red, the broader context is marked in blue, and elements that connect the bracket to the mechanical assembly, such as screws and clamps, are marked in green.

The inspection procedure is performed by a robot arm, equipped with several sensors. A control effector offers great flexibility and inspection capacity (see Fig. 4).

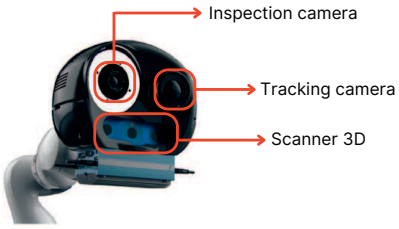


Fig. 4: Robotic end effector developed by *Diota*, with three mounted sensors: an inspection camera (narrow field of view), a tracking camera (wide field of view), and a 3D scanner *Ensenso N35* for 3D point cloud acquisition.

2 Previous works

The rise of artificial intelligence (AI) has benefited many fields, such as self-driving cars, or more recently, the aerospace sector. Among the major tasks that can use AI, various aeronautical manufacturers are focusing on visual control, and more precisely, on anomaly detection.

In aeronautical production or maintenance, the vast majority of visual inspection tasks aim to find defects or anomalies. Typically, those are long-lasting tasks requiring significant focus of the operator. With the constant ascent in air traffic and high demand on maintenance personnel to meet commercial schedules, the pressure on inspection operations is increasing, increasing the risk of errors [1, 2].

Time-consuming and expensive, human controls can be incomplete, prone to errors, and often lack traceability. In this context, we launched a project on the inspection of complex aeronautical assemblies in the production phase, based on the exploitation of 2D images or 3D point clouds [3–5]. The work presented in [6] dealt with the classification of 3D point clouds after training on real data, whereas the current work relies on synthetic data for training and real data for evaluating the models. This work is also complementary to our segmentation approach to the same problem, demonstrated in [7]. This work is a continuation of these efforts, focusing on leveraging deep learning classification techniques to analyze 3D point clouds.

Since we do not have labels for the target domain (real point clouds), the only available labeled samples come from another distribution, i.e. from the source domain (synthetic point clouds). This scenario is commonly named "Unsupervised Domain Adaptation" (UDA). In this so-called *sim2real* case, the source domain corresponds to the synthetic data, and the target domain is that of the real data. There are many more works on the topic of domain adaptation with *2D* images [8–11] compared to domain adaptation in *3D* data.

In *3D* UDA there are two major approaches: (1) the extension of the adversarial structures of Domain Adaptation from 2D to 3D, (2) *Self-Supervised Learning* (SSL) on 3D points to facilitate the learning of domain invariant features. Unlike previous work in a 2D environment, adversarial methods in a 3D environment have demonstrated flaws between a local geometric alignment and a global semantic alignment, by causing geometric distortions of clouds by trying to align domains [12]. This eliminates this approach. Self-supervised learning (SSL) has recently been shown to be very effective in learning relevant representations from unlabeled images [13–16] or videos [17–19]. This learning results in a domain-invariant feature extraction. Unfortunately, we found that these domain adaptation methods incorporated real, although not labeled, data in the training phase. We had to abandon using these methods since we do not want to involve real data during the learning phase at all. We therefore explored more broadly 3D deep learning methods adapted to point clouds without restricting ourselves to methods specifically dedicated to domain adaptation.

In 2017 *PointNet* [20] revolutionized the way of digitally understanding 3D point clouds by no longer considering a point cloud as an ordered set, but as a set of data invariant to rigid movements (i.e rotations) opening a new generation of 3D convolutional networks.

In the point cloud convolutional networks family, parameterized convolutions are the most popular. Usually, they associate points individually to parameterized kernels, but recently, many are

based on neighbourhood. *PointCNN* [21] learns a χ-transformation for each neighborhood of points. This transformation is applied to points to embed them in a domain with a regular structure where we can easily apply the convolution. Similarly, *KPConv* [22] proposed learning the exact kernel function in a set of fixed points in space (kernel points). Convolution is applied to the point cloud by summing the weighted features of the data points around each kernel point. The operator *ShellConv* of *Shellnet* [23], builds concentric spherical shells on each local neighbourhood. Information representing features of each shell is extracted by mapping points from each shell to high dimensional space and applying the aggregation function. Features extracted from different shells are then convolved using a 1D convolutional layer. Such an approach achieves good results on public datasets while being computationally more effective than *PointCNN* [21]. *SpiderCNN* [24] differs by applying weights on the points according to their order in the point cloud, which makes this structure non-invariant to the order of the inputs. On the contrary. The *Edgeconv* convolution operator used in *DGCNN* [25] neural network architecture, encodes relations between points by graphs, thus allowing local geometrical structures extraction, and also remaining invariant to the permutation of inputs. In all architectures mentioned in this paragraph neighbourhood points were selected using *k-Nearest Neighbors (KNN)*, except in *KPConv* which selects *radius neighbourhoods*, ensuring the robustness of the convolutions to variable densities, as argued in work of Thomas, et al. [26].

Sparsity is a key notion in point clouds. *PointCNN* [21] introduces sparsity to data but also to its convolution kernels. This allows a gain in computation time. Aiming at the same goal, *KPConv* proposes a subsampling strategy alleviating both the density and the computational cost. Even though *PCNN* [27] structure is the closest to *KPConv*, its assignment of weights to points and not to neighborhoods, makes the convolutional calculation quadratic to the number of points.

We have selected and evaluated the most relevant methods for the problem at hand: *ShellNet*, *PointCNN*, *PCNN*, *KPCONV*, *DGCNN*. All of them belong to the parametric convolutions family except *DGCNN* which belongs to the family of convolution graphs. Our architecture selection was based on the performance reported on the ModelNet 40 [28], a freely available dataset. Architectures of *PointCNN*, *ShellNet* and *KPConv* are descendants of the unmissable *PointNet* [20] network.

3 Evaluation Metrics

Let us consider one mechanical part in the assembly. The part has its predefined placement in the assembly. Each observation (point cloud) of this place can be assigned one of the two labels: 0 or 1. Label 1 means that the mechanical part is not mounted, and label 0 means that the mechanical part is mounted.

There are a few evaluation metrics for binary classification: *Accuracy*, *Precision*, *Recall*, *F1-score*. We will define each of them in terms of:

- True Positive (TP): an observation with label 1 is correctly classified as 1;
- False Positive (FP): an observation with label 0 is wrongly classified as 1;
- True Negative (TN): an observation with label 0 is correctly classified as 0;
- False Negative (FN): an observation with label 1 is wrongly classified as 0.

Accuracy measures how often the classifier correctly predicts. Formally it is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision measures the proportion of the correct positive predictions, among all the positive predictions. Formally it is defined as:

$$Precision = \frac{TP}{TP + FP}$$

Recall measure how often a machine learning model correctly identifies positive instances. Formally it is defined as:

$$Recall = \frac{TP}{TP + FN}$$

F1-score takes into account both precision and recall metrics by calculating their Harmonic mean.

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

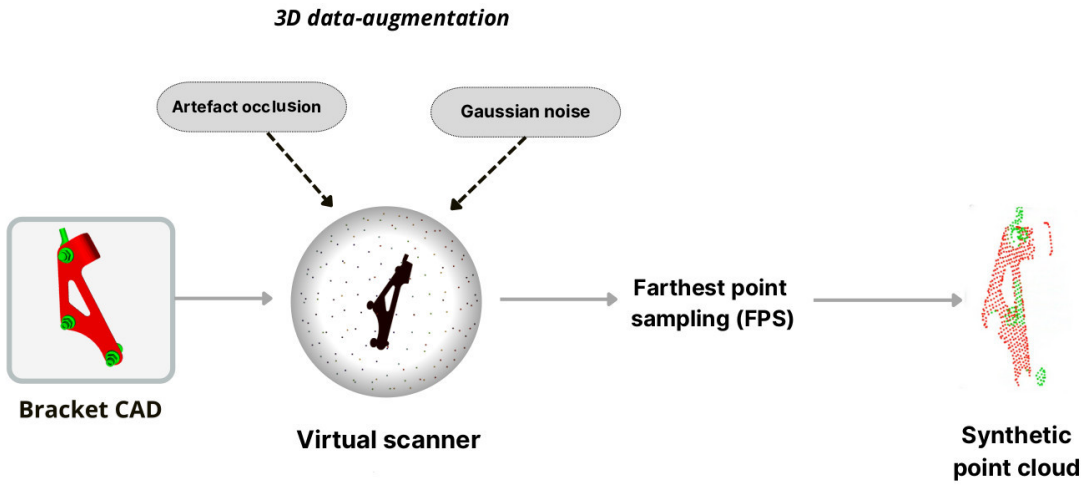


Fig. 5: Complete description of the synthetic data generation process

In our use case, we perform multi-class classification, so binary classification metrics are not directly applicable. However, we can compute them for each class independently, by considering each class prediction as a binary classification problem and taking the average as the multi-class classification metric. We use the method of weighted averaging.

$$Weighted_{F1\text{-}score} = \sum_{i}^{M} \frac{n_i}{N} F1\text{-}score_i$$

$$Weighted_{Recall} = \sum_{i}^{M} \frac{n_i}{N} Recall_i$$

$$Weighted_{Precision} = \sum_{i}^{M} \frac{n_i}{N} Precision_i,$$

where M is the number of classes, n_i the number of observations belonging to class i, N is the total number of observations we have, and $F1\text{-}score_i$, $Precision_i$ and $Recall_i$ are calculated *F1-score*, *Precision* and *Recall* of class i.

4 Synthetic data generation pipeline

We used a z-buffer-based method to generate synthetic 3D acquisitions from theoretical viewpoints. This method takes into account the self-occlusions of the element, just as the real scanner does. Also, this method ensures a realistic rendering very close to that obtained with the scanner (see some 2D renders in Fig. 6).

Because of security conditions, the scanner in real applications is positioned at a working distance of approximately 60 cm from the centroid of the mechanical part being inspected. To make the data as realistic as possible, we sampled N points from the Fibonacci sphere of radius 60 cm to use them as virtual camera centers for generating synthetic point clouds. Here N is a hyperparameter that should be selected optimally.

To keep a fixed number of points per cloud, we downsample all generated point clouds to 2048 points with the *Farthest Point Sampling* (FPS) method [29].

To increase the versatility of synthetically generated data, various forms of 3D data augmentation can be integrated into the synthetic data generation process. More versatile data should allow the model to become invariant to the variabilities to which it may be exposed in the prediction phase. Indeed, a synthetic scan of each CAD model can be seen as a "perfect" sample, without noise or occlusions by external *artifacts*. Nevertheless, some objects, such as plastic caps and protectors, are not present in the CAD model. Therefore, to generate more realistic point clouds, we implemented data augmentation methods specific to the problem we are solving:

- adding *gaussian noise* to the synthetically generated point cloud to simulate scanner acquisition noise.

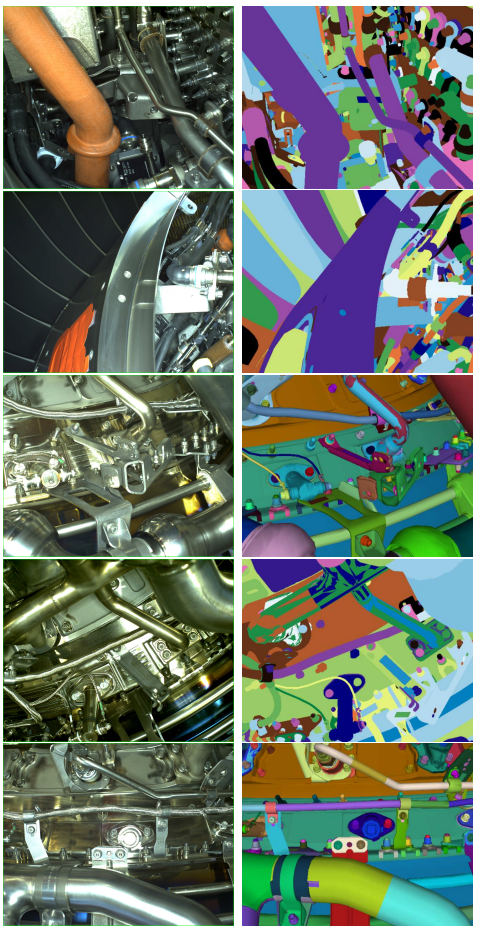


Fig. 6: The first column represents the 2D images taken by the camera with a wide field of view. The second column represents the synthetic 2D renderings corresponding respectively to the image acquisitions from the first row.

- adding *artifacts* to the CAD model before making a synthetic scan, simulating possible occlusions

A summary diagram of the synthetic data generation pipeline is presented in Fig. 5.

To check if deep learning models are capable of learning complex shapes in an industrial environment, we run a few experiments on synthetically generated data only. We trained models on synthetically generated data and tested them on a different subset of synthetically generated data. Here we also tested how synthetically generated data from one pipeline generalize by testing them on data generated through different pipelines. These experiments will notably enable us to determine which amount of synthetic information ranging from the bracket alone to the bracket-fusion with context provides the best information for further generalization to real data. Also, we will conclude how many virtual camera poses benefits to the data quality, since too many viewpoints may not add any information beneficial for the model learning.

For the experiments on synthetic data only, we used the *PointCNN* network. Indeed, due to its structure, its speed of learning, and its good performance, this algorithm is very popular in the domain and also inspired many other works, such as *ShellNet*. This architecture appears even more legitimate as it contains modules taken directly from *PointNet* which inspired the entire new generation of 3D *deep learning*. We therefore have good reasons to believe that training on *PointCNN* will give us a good grasp on the results that may come from other 3D *deep learning* networks.

We have not chosen *PointNet* for this experimental phase, simply because we preferred to base ourselves on a more recent architecture, which outperforms the results provided by *PointNet* on publicly available datasets.

4.1 CAD information used for data generation

The following experiments aim to determine whether it is better to train the network on the brackets alone as shown in Fig. 3a, or on bracket-fusion (i.e., brackets with their close context) as shown in Fig. 3b.

4.1.1 Training on the bracket-alone dataset

The first experiment aims to analyze the performance of training on the synthetic brackets alone and then to assess the generalization also on the synthetic brackets alone, as well as on the synthetic brackets-fusion (see Table 1).

We proceed with the training (learning/validation) of a model on the D1 dataset corresponding to synthetic data of the brackets alone. After training, we tested the model on the D1 (brackets-alone) and D3 (brackets-fusion) data, where we

Distribution	Dataset	Type	No. Examples
Training	D1-train	brackets-alone	1952
Validation	D1-val	brackets-alone	244
Test	D1-test	brackets-alone	244
Test	D3-test	brackets-fusion	244

Table 1: Distribution of datasets.

assessed the performance using mentioned metrics (see Table 2).

Test	*Accuracy*	*Recall*	*Precision*	*F1-score*
D1	100%	100%	100%	100%
D3	75%	75%	76%	72%

Table 2: Evaluation of the model trained on brackets-alone dataset (D1).

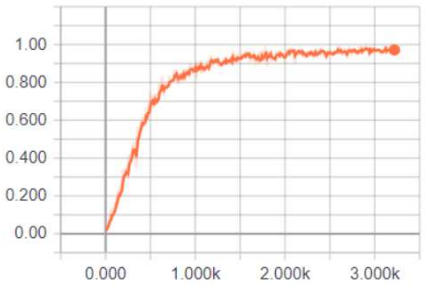
We observe that the model provides excellent performance on the dataset of the brackets alone (D1), but it gives us less satisfactory generalization on the brackets-fusion (D3). This indicates that the close context is a significant and even discriminating factor for the classification of the brackets.

To check out if the model just overfits on dataset D1, we repeated training with the usage of validation data from the D3 dataset. Here, we aim to determine at which point of the learning on D1, the model no longer generalizes on D3. Thus, we examine the learning curves to stop the learning before it starts generalizing poorly to the validation data (see Fig. 7).

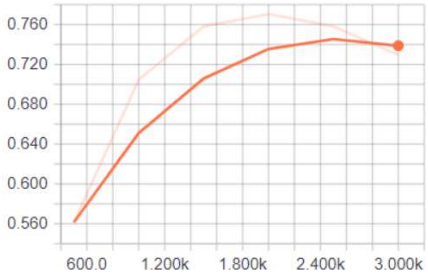
We decided to stop the learning process as soon as we saw signs of overfitting on D1 with the hope of getting the best possible generalization on D3. We note a very large performance gap between the Validation set and the Train set, over 20% in favour of the Train set. We examined the repercussions of this learning in the inference phase on D1-test and D3-test (see Table 3).

Test	Accuracy	Recall	Precision	F-score
D1	100%	100%	100%	100%
D3	69%	69%	70%	65%

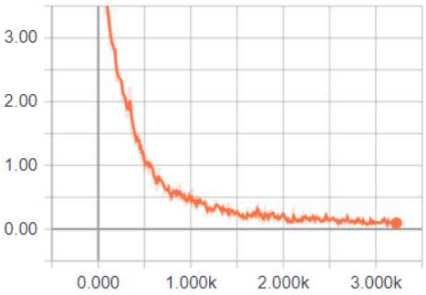
Table 3: Overview results achieved by model trined on brackets-alone dataset(D1) on datasets D1 and bracket-fusion dataset (D3).



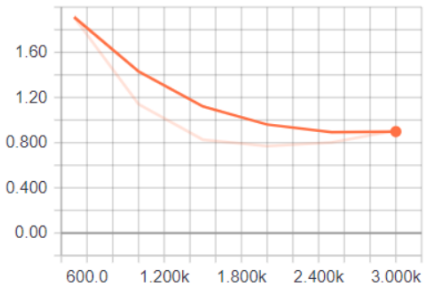
Training accuracy



Validation accuracy



Training loss



Validation loss

Fig. 7: Learning curves for Accuracy and Error. The train curves on D1 and the validation curves on D3.

4.1.2 Training on bracket-fusion dataset

After testing how the model trained on bracket-alone (D1) performs on bracket-fusion (D3), we conduct a reverse experiment: training on the bracket-fusion (D3) data and an evaluation on the bracket-alone (D1) data. The distributions of these datasets are detailed in Table 4. Note that both datasets, D1 and D3, are synthetic.

Distribution	Dataset	Type	No. samples
Training	D3-train	Bracket-fusion	1952
Validation	D3-val	Bracket-fusion	244
Test	D1-test	Brackets alone	244
Test	D3-test	Bracket-fusion	244

Table 4: Distribution of datasets.

The evaluations on the test datasets of the model after its learning are detailed in Table 5.

Test	*Accuracy*	*Recall*	*Precision*	*F1-score*
D1	97.5%	97%	96.9%	96.9%
D3	100%	100%	100%	100%

Table 5: Evaluation of the model trained on bracket-fusion dataset (D3).

It can be observed that the model trained on the D3 dataset (bracket-fusion) has better generalization when inferring on D1 (brackets-alone) than the previous model when inferring on D3. We deduce that the model can generalize very well to data without close context even though it has been trained on data with close context.

Two important aspects should be retained:

1) Real-world acquired 3D data will necessarily be of the support-fusion type because standalone brackets do not hold their position without their immediate context.
2) The fact that the model accurately recognizes standalone brackets without their close context may indicate that potentially a heavily damaged bracket could also be recognized, which would be a mistake.

The first argument would tend to encourage us to train a model on bracket-fusion rather than standalone brackets while maintaining vigilance concerning acquisitions of the damaged brackets.

4.2 Optimal number of virtual viewpoints

In this experiment, we looked for the optimal number of viewpoints N around the Fibonacci sphere and its impact on the quality of data. See Table 6 for the data distribution. To do this, we define D_2, a synthetic dataset of bracket-fusion with $N = 160$.

Distribution	Dataset	Type	No. samples
Train	D2-train	Bracket-fusion	7808
Validation	D2-val	Bracket-fusion	976
Test	D2-test	Bracket-fusion	976
Test	D1-test	Brackets alone	244

Table 6: Dataset distribution.

The results in the testing phase are presented in Table 7. This experiment leads to the same results as in 4.1.2 after learning on D3. Indeed, this model generalizes very well on brackets alone even though it has learned on bracket-fusion. Furthermore, it is worth noting that the performance is almost the same as the previous experiment. We can deduce that the model is neither subject to overfitting nor to poor generalization on brackets alone when the number of viewpoints is increased. This experiment shows that for $N = 160$ and $N = 40$, the results are similar. Thus, for the subsequent experiments, we will keep N fixed at 40.

Test	Accuracy	Recall	Precision	F-score
D1	96%	96%	95.9%	95%
D2	100%	100%	100%	100%

Table 7: Evaluation of the model trained on the synthetic data generated from 160 different viewpoints (dataset D2).

4.3 Data augmentation

This experiment aims to assess the contribution of the data augmentation during the training phase. Namely, we train the model on a new dataset, D_{DA}. This dataset contains samples coming from various forms of data augmentation. We start from the D3 dataset with its train-test-val distribution and for each sample point cloud, we add an artifact and introduce Gaussian noise to the position

of the points. Thus, each acquisition is represented in 3 different forms:

- an acquisition of bracket-fusion from a fixed point of view,
- an acquisition from the same point of view but with an artifact (which may or may not be occluding the bracket),
- an acquisition from the same point of view but with Gaussian noise of $\sigma = 0.2$.

A tabular overview of data distribution is given in Table 8.

Split	Dataset	Type	No. samples
Train	D_{DA}-train	Bracket-fusion	5856
Validation	D_{DA}-val	Bracket-fusion	732
Test	D_{DA}-test	Bracket-fusion	732
Test	D3-test	Bracket-fusion	244

Table 8: Data distribution.

After training on the augmented dataset, we obtain the performance that can be seen in Table 9. The results demonstrate that the data

Test	Accuracy	Recall	Precision	F-score
D_{DA}	100%	100%	100%	100%
D3	100%	100%	100%	100%

Table 9: Evaluation of the model trained on the augmented dataset -(D_{DA}).

augmentation helps the neural network to achieve the best results on bracket-fusion D3 or bracket-fusion with artifacts and/or Gaussian noise D_{DA}. Consequently, we decide that the dataset D_{DA} represents the best dataset to carry out the next experiments.

4.4 Including real data into validation set

In this experiment, the training is done on synthetic data, while the validation is performed on both synthetic and real data. We aim to progressively evaluate the model on the real data, during the training, to ensure that the model does not overfit on the synthetic training data. Thus, we evaluate the model during the training on both synthetic and real validation data, to validate the network as it learns, but also to assess its generalization capability on the real domain. We conduct the training on the dataset D_{DA}, to include data augmentation in the training.

We have 590 real point clouds, coming from 20 different types of brackets. These data make up the dataset called Presence set obtained using the 3D scanner. The scanner has a large field of view compared to the size of the bracket being inspected. As detailed in Sec. 1, thanks to our localization module, we can calculate a 3D region of interest and apply it on the full real cloud, to crop the part which is of interest to us. Further, we apply a *FPS* to reduce the difference in point distribution between synthetically generated data and data from the 3D scanner. These are the data we will use to verify the generalization of the model on real data. We present the number of samples dedicated to training and testing in Table 10.

Split	Dataset	No. Examples	Data Type
Train	D_{DA}-train	5856	Synthetic
Validation	D_{DA}-val	732	Synthetic
Validation	Presence set	590	Real
Test	D_{DA}-test	732	Synthetic
Test	Presence set	590	Real

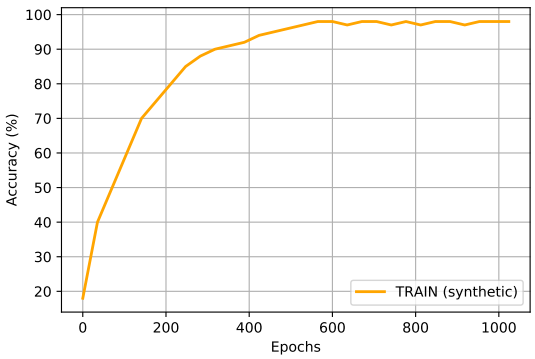
Table 10: Data distribution.

In Figure 8, we present the performance graphs during training of the *PointCNN*network on the D_{DA}-train data. We observe excellent generalization of the model on both synthetic and real validation data.

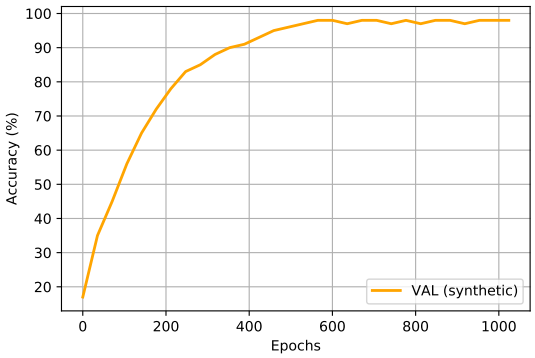
We even note that the model surpasses the 90% performance threshold on the real data. This result assures us of the good generalization ability of the model on real-world data.

Although we evaluated the model on synthetic data and real data during learning, we did so only to visually infer from the learning curves whether any overfitting occurred. To decide when to stop the training, we used the result from the synthetically generated data only. After training the network, we evaluate it to confirm the validity of the synthetic training after it is completed. The results on the real acquisitions are given in Table 11.

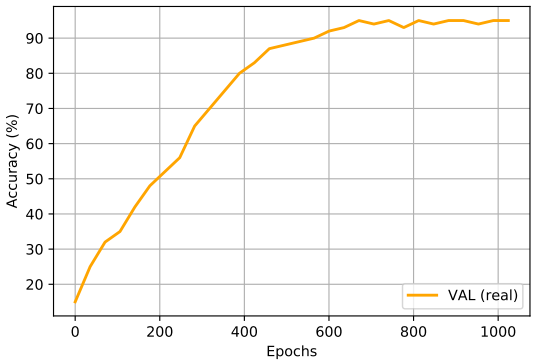
We observe that the model generalizes well on the real acquisitions, exceeding the 95% threshold on each of the metrics.



(a) Train Dataset



(b) Val Dataset - Syntehtic data



(c) Val Dataset - Real data

Fig. 8: Charts during learning on the D_{DA} dataset with validation on real and synthetic data to prevent overfitting. (a) Accuracy evaluation on D_{DA}-train; (b) Accuracy evaluation on D_{DA}-val; (c) Accuracy evaluation on the 590 real presence samples.

	PointCNN				
	Acc.	Prec.	Rec.	F1-sc.	Nb. acq.
Avg.	96%	98%	96%	97%	590

Table 11: Results for the average of the 20 classes of real acquisitions corresponding to the Presence set. The metrics used are Accuracy (Acc.), Weighted Precision (Prec.), Weighted Recall (Rec.) and Weighted F1-Score (F1-sc.).

4.5 Synthetic data with wider context

Some viewpoints used for generation of *bracket-fusion* dataset (D3) (as in Fig.3b), may be inaccessible to the camera in reality. Consequently, the D3 dataset contains some unrealistic data, which may introduce bias to the experiments. This occurs because we did not consider the context around the bracket. By involving a wider context (cables, pipes, etc.) during data generation (as shown in Fig.3c), we can eliminate these unrealistic data from the training set. Additionally, including the context can reduce confusion between classes and make models more robust to occlusion. Unfortunately, due to strong occlusion by the context, there are not many viewpoints from which brackets are visible. Therefore we generated data with a larger context (*D5*) and combined this data with D_{DA} dataset.

The results of testing on the Presence set are presented in Table 12. In the first row, we presented results of *PointCNN* model trained on D_{DA} dataset, and in the second-row results for a model trained on combined D_{DA} dataset, and synthetic point clouds with the surrounding context (*D5*). From this experiment, we can conclude that the surrounding context does not help in distinguishing different types of mechanical parts.

The precision is slightly increased, while the recall is decreased. Even though context does not help us in classification between different types of mechanical parts, it will be useful to detect cases when a mechanical part is missing (Section 6.1).

4.6 Conclusions on the data generation pipeline

By performing previous experiments, we made different choices on the synthetic data generation pipeline. The main conclusions are:

Training dataset	Training samples	Acc.	Prec.	Rec.	F1-sc.
D_{DA} (Fig. 3b)	5856	96%	98%	96%	97%
D_{DA} + $D5$(Fig. 3c)	7264	95%	98%	95%	96%

Table 12: Results of *PointCNN* model on **real dataset**. The model in the first row was trained on an augmented dataset without context, and the model in the second row was trained on a combination of augmented data without context, and data generated with surrounding context. The used metrics are Accuracy(*Acc.*) weighted-precision (*Prec.*), weighted-recall (*Rec.*) and weighted F1-score (*F1-sc.*).

- Data generated from bracket-fusion CAD models, generalize better than data created from bracket only.
- Setting the number of virtual viewpoints to $N = 40$ yields good results on both synthetic and real domains, exceeding 96% on all metrics.
- Training on synthetic D_{DA} dataset, including data augmentation, generalizes very well to the real domain.
- The inclusion of surrounding content does not help in distinguishing different types of brackets.

5 Choosing deep neural network architecture

So far we have worked on the classification of different types of mechanical parts. This can be used to detect if the wrong type of mechanical part has been mounted. We focused on testing different pipelines for generating synthetic data. To draw conclusions about the data generation, we used only the *PointCNN* model. We will now test how other deep learning architectures perform on this task. In this section, we also provided a more detailed analysis of results obtained by *PointCNN* model, by comparing per-class results with other models.

For training, we used a dataset D_{DA} of 3D point clouds generated from 3D CAD models of 61 different brackets. For the choice of architecture, we decided on *PCNN*, *PointCNN*, *Shellnet*, and *DGCNN*. For hyper-parameters selection, we used hyper-parameters recommended for ModelNet40 dataset 28 with *PCNN*, *PointCNN* and *Shellnet*. To ensure convergence of *DGCNN* we had to slightly tune parameters for our datasets. Namely, we reduced the batch size to 20 and assigned k=25 for the k-nearest-neighbor function.

After training the models on synthetic data, we evaluated them on (1) synthetic data and (2) real data, i.e. real 3D point clouds acquired by the 3D scanner in an industrial environment.

We have evaluated four selected networks. The results are expressed using the following average metrics: weighted precision, weighted recall, and weighted F1-score. The results per class are presented in Fig. 9 and overall metrics for each model can be seen in Table 13.

Model	Precision	Recall	F1-score.	Nb. acq
ShellNet	95%	91%	93%	590
PointCNN	98%	96%	97%	590
DGCNN	95%	77%	84%	590
PCNN	94%	72%	81%	590

Table 13: Results for each model based on the average of the metrics over all 20 real acquisition classes. The used metrics are weighted-precision (*Prec.*), weighted-recall (*Rec.*) and weighted F1-score (*F1-sc.*).

We can see in Fig. 9 that *PointCNN* achieves accuracy higher than 80% on all classes. For classes with many samples (e.g. classes 21, 49, 50), we managed to get good performances with most of the models. However, *DGCNN* demonstrates high sensitivity to occlusion by cables and some disposable elements not present in CAD.

An example is class 52 where *DGCNN* demonstrates an *Accuracy* of 22%. It can be seen in Figure 10 that the view to bracket in class 52 is obscured by two large cables, which distorts the point cloud relative to the shape of the CAD model, and leads *DGCNN* to error.

ShellNet shows difficulties in discriminating between mechanical parts of similar shapes. As can be seen in Figure 9, the accuracy for class 13 is 0%. With deeper analysis, we observed that class 13 was systematically confused with class 50 which has a very similar appearance to class 13 (see Figure 11). This behavior invalidates this network.

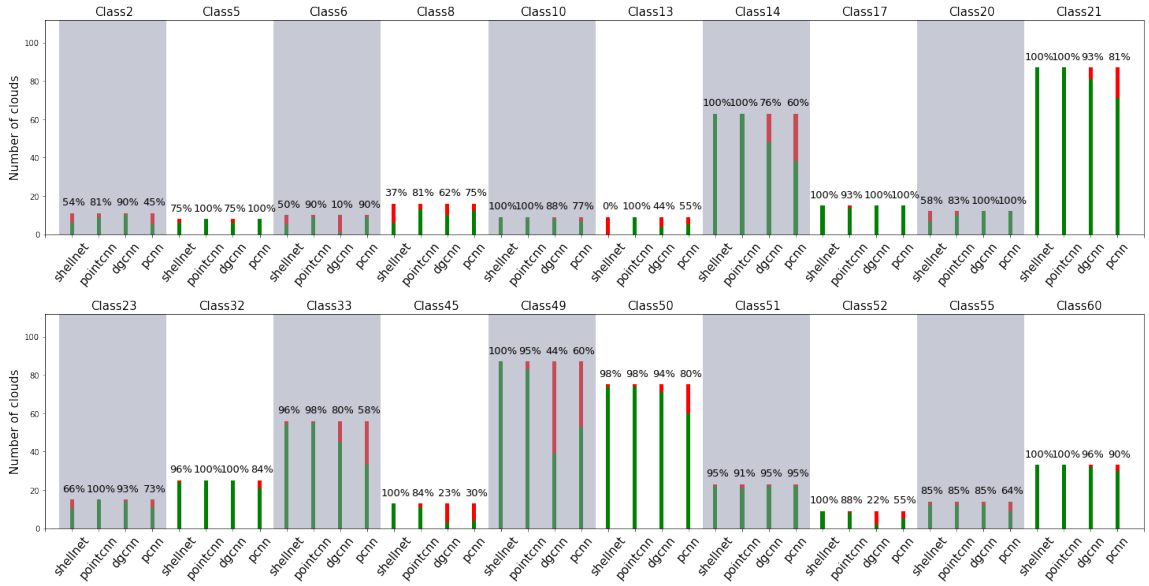


Fig. 9: Percentage of correct classifications of each model, per class, evaluated on real data. In green: the percentage of acquisitions correctly classified by a model. In red: the percentage of acquisitions wrongly classified by a model. On the Y-axis: the total number of tested acquisitions. On the X-axis: four classification models. White and grey vertical bands are added to separate the classes for better readability.

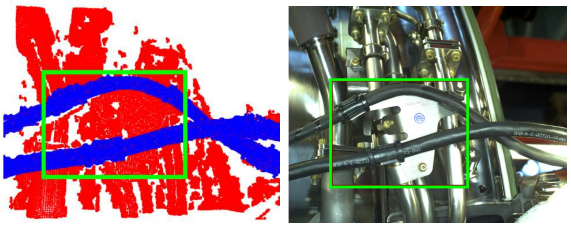
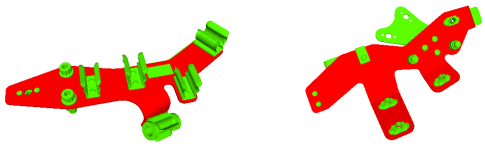


Fig. 10: Bracket 52 is occluded by cables. The green rectangle shows the bracket in the scene.



(a) Bracket 13 (b) Bracket 50

Fig. 11: Comparison between bracket 13 and bracket 50

PCNN shows the most instability. It is very sensitive to the localization error and to occlusion by even the finest cable. Moreover, as soon as the acquisition is performed on a slightly light-reflecting surface, the empty zones of points distort the cloud and the acquisition becomes unmanageable by *PCNN*. This reveals the difficulty of *PCNN* facing variations in point density.

(a) (b) (c)

Fig. 12: Visualization of bracket 21 for a given acquisition: (a) Synthetic 2D rendering; (b) Image acquisition; (c) 3D point cloud.

We find that even with well-visible brackets and without any occlusion, the *PCNN* network encounters difficulties. This is the case with the example in Figure 12 which shows bracket 21, for which the performance of *PCNN* on Figure 9 is the lowest.

Given the analysis above, we have decided to continue our study by using the *PointCNN* network, which shows the most stable behavior on real data.

6 Solving absence-presence problem

Up to this point, we have only considered the case where the mechanical part is mounted but may be of the wrong type. As we have seen the model trained on synthetically generated data can achieve high accuracy on the real data from the production line. Now we turn our attention towards a more challenging problem, detection if the mechanical part is not mounted and not replaced by another part. Since *PointCNN* appears to be the most robust for the gap between synthetically generated data and real data, we chose this architecture for the experiments in this section.

To simulate the absence of a mechanical part, we generated clouds of context only, for each type of mechanical part (as in Fig. 3d). When filtering those context clouds, we kept the same viewpoints that we used for the cases when the mechanical part is present, i.e. those viewpoints from which the mechanical part is visible enough, when present. Such an extended synthetic dataset has a distribution as shown in Table 14.

	Train	Val	Test	No. Cls	Pres./Abs.
Dataset D_{DA}	5856	732	732	61	Presence
Data with context	1404	184	183	61	Presence
Context alone data	1404	184	183	61	Absence

Table 14: Distribution of the synthetic dataset. No. Cls. - Number of classes, Pres./Abs. - Presence/Absence.

6.1 Training classifier with 61 presence and 61 absence classes

There are several possible interpretations of this problem. At first, for each mechanical part, we used two classes, one to represent the case when mechanical part is mounted, and one to represent the case when mechanical part is not mounted. In this representation, we got 122 different classes.

The model was trained on synthetic data and tested on different subset of synthetic data. Results can be seen in Table 15. We can notice that the model shows significantly worse results in cases where the mechanical part is not present. It can be concluded that the context alone is not discriminating enough for classification. So it is not expected to obtain great results on real data.

	Acc.	Prec.	Rec.	F1-sc.	Nb. samples
Dataset D_{DA}	99%	99%	99%	99%	732
Data with context	88%	97%	88%	91%	187
Context alone data	88%	98%	87%	91%	183

Table 15: Results on test subsets of the synthetic datasets.

We have 33 real acquisitions representing the cases where a mechanical part is missing, divided into 14 different classes. This set will further be nicknamed *Absence set*. We evaluate the model on *Presence set* and *Absence set* with real acquisitions. Results are presented in Table 16.

	Acc.	Prec.	Rec.	F1-sc.	Nb. acq
Presence set	49%	92%	49%	55%	590
Absence set	58%	58%	57%	57%	33

Table 16: Results on real clouds. Accuracy (*Acc.*) corresponds to total accuracy. The used metrics: weighted-precision (*Prec.*), weighted-recall (*Rec.*) and weighted F1-score (*F1-sc.*).

Both results obtained on synthetic data as well as those obtained on real data are not satisfactory. So we checked the false positives proportion on *Presence set* (see Fig. 13). False positive (false alarm) is an outcome in which the model raises an alarm that a bracket is missing, whereas the bracket is present.

Fig. 13 shows a very high proportion of false alarms on *Presence set*, precisely 230 acquisitions out of 590 (i.e 39% of *Presence set*). This huge proportion of false positives explains the low scores reported in Table 16. The huge precision for *Presence set* is explained by the fact that the model rarely confuses the presence cases between them. These results suggest that the context is a source of confusion for a model.

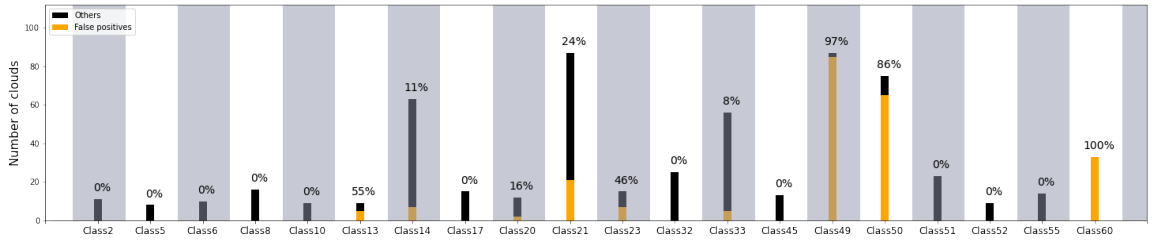


Fig. 13: Percentage of false positive classifications (false alarm) for each class, evaluated on real data with present bracket. In orange: the false positive percentage by class. In black: complement to 100%. On Y axis: the total number of tested clouds. On X axis: classes of tested real clouds. White and grey vertical bands are added to separate the classes for better readability.

The per-class accuracy on *Absence set* is shown in Fig. 14. These results confirm the high error rate presented in the second row of Table 16. By closely examining the 42% of misclassified data from the Absence set, we observed that there are only 10% of samples wrongly classified as presence case, while confusion between different absence classes represents the remaining 32% of error. These results once again suggest that the context is a source of confusion for a model.

6.1.1 Binarization of the outputs for Absence/Presence discrimination

As already mentioned, knowledge about the expected mechanical part is available in advance. A primary industrial need in this project is the ability to recognize the absence of a mechanical part. To adapt to this requirement, we decided to consider all presence classes as a single class, and all the absence classes as a single class. If no bracket is present, the solution should answer "bracket missing".

Note that the model is not retrained; we are simply interpreting the model's outputs differently. The results are presented in Figure 15 for the Presence set. We obtain an *Accuracy* of 54% on this dataset, which is barely higher than a random response. For the per-class results on Absence set, one can refer to Figure 16, and the average results for absence cases are given in Table 17.

These results are much better, and this interpretation raises the total accuracy to 91% on the *Absence set*, demonstrating a good ability of the model to detect the absence of a bracket.

We present the results of the binarization (presence vs. absence) of this model in the form of a confusion matrix in Table 18.

	PointCNN				
	Acc.	Prec.	Rec.	F1-sc.	Nb. acq
Avg.	91%	100%	91%	95%	33

Table 17: Average results on the 14 classes of real acquisitions associated with Absence set. The used metrics are *Accuracy* (Acc.), Weighted-*Precision* (Prec.), Weighted-*Recall* (Rec.) and Weighted-*F1-score* (F1-sc.).

label/pred	Presence	Absence	Nb. acq
Presence	319	271	590
Absence	3	30	33

Table 18: Results on real acquisitions after binarization of the model outputs.

False Negatives (FN) are indicated in red. We aim to be robust in recognizing absence cases for each type of bracket, in other words, to have a network that prioritizes the prevention of FNs over FPs (False Positives). However, we must remain vigilant about the fact that FPs must be in a reasonable proportion. We observe that the binarization of the outputs shows an appealingly low FN ratio. Unfortunately, it classifies a lot of presence samples as absence, which produces 271 FPs at a rate of 46%. This very poor performance in presence case invalidates this approach.

6.2 Training a binary model

Our next attempt to improve results was to train a new binary model. This learning was carried out on the dataset presented in the Table 19. The neural network was trained to properly recognize the situation of bracket absence from the one relating to the presence of bracket.

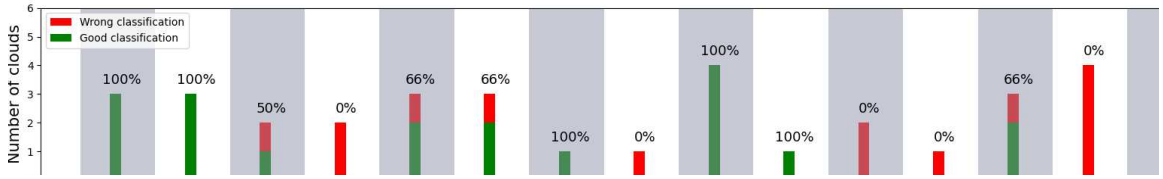


Fig. 14: Percentage of correct classifications by class, evaluated on Absence set. In green: the percentage of clouds correctly classified. In red: the percentage of clouds wrongly classified. On the Y-axis: the total number of tested clouds. On the X axis: classes of tested real clouds.

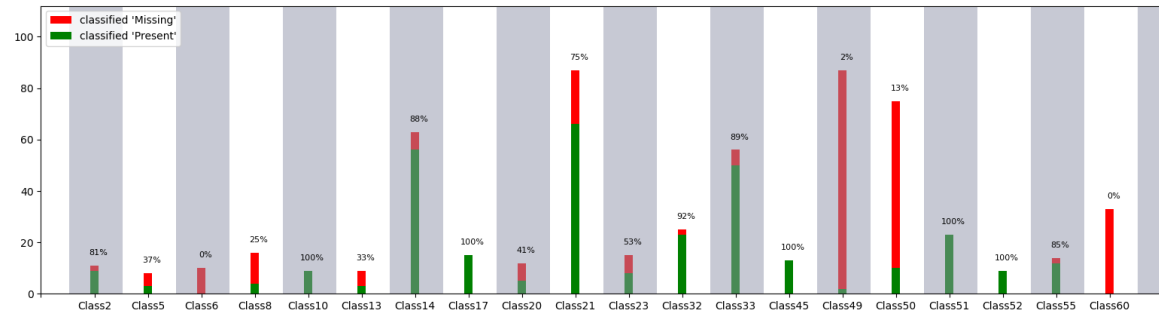


Fig. 15: Percentage of correctly classified presence cases, ordered by class, evaluated on Presence set. In green: the percentage of clouds correctly classified as presence case. In red: the percentage of wrongly classified clouds (classified as absence). On Y axis: the total number of tested clouds. On the X axis: classes of tested real clouds.

Fig. 16: Percentage of correctly classified absence cases, ordered by class, evaluated on Absence set. In green: the percentage of clouds correctly classified as absence. In red: the percentage of wrongly classified clouds (classified as presence). On Y axis: the total number of tested clouds. On the X axis: classes of tested real clouds.

Although training on the synthetic domain has shown to be quite good, exceeding 90% on all synthetic sets, evaluating the model on real data has demonstrated that the network classifies almost all samples as "absence". Indeed, a test on the real data of the Absence set leads to an *Accuracy* of 91% and on the Presence set, an *Accuracy* of 17% is achieved.

Thus, in terms of presence discrimination, the binary model clearly does not prove itself, as evidenced by its confusion matrix (see Table 20).

This indicates that the binary model predicts the absence of the bracket almost systematically. Therefore, we decided to invalidate this approach. We question whether the presence of context in

	Train	Val	Test	No. Cls	Pres./Abs.
Dataset D_{DA}	5856	732	732	1	Presence
Data with context	1404	184	183	1	Presence
Context alone data	1404	184	183	1	Absence

Table 19: Distribution of the synthetic dataset. No. Cls. - Number of classes, Pres./Abs. - Presence/Absence.

label/pred	Presence	Absence	Nb. acq
Presence	100	490	590
Absence	3	30	33

Table 20: Results on real acquisitions obtained with the binary model.

learning could lead the model to make errors during the evaluation on real data. We decided to explore an additional path by removing the context from the presence case clouds in the training.

6.3 Training binary model without context in presence case clouds

Previous experiments have repeatedly emphasized the fact that the presence of context in learning on synthetic data of presence and absence, could lead to confusion during inference on real data. From the experiments on multi-class classification, we concluded that it can be useful to train models on data without context. We have therefore removed context from the training data, as it was the cause of confusion between absence and presence. This gives us the distribution of synthetic point clouds as shown in Table 21.

	Train	Val	Test	No. Cls	Pres./Abs.
Dataset D_{DA}	5856	732	732	1	Presence
Context alone data	1404	184	183	1	Absence

Table 21: Distribution of the synthetic dataset. No. Cls. - Number of classes, Pres./Abs. - Presence/Absence.

Evaluation results on the real test set are demonstrated in Table 22.

The results on real data, summarized in the confusion matrix in Table 23, confirm that this

	Acc.	Prec.	Rec.	F1-sc.	Nb. samples
Presence set	0%	0%	0%	0%	590
Absence set	100%	100%	100%	100%	33

Table 22: Results of the binary model on real acquisitions.

label/pred	Presence	Absence	Nb. acq
Presence	0	590	590
Absence	0	33	33

Table 23: Results on real acquisitions after binarization of the model outputs.

model systematically classifies real-life cases as the absence of brackets. This is most likely because every real cloud involves the context, whereas in the training phase, the context is only presented to the model within absence samples.

6.4 Training model with 61 classes of presence and 1 class of absence

This model aims to consider the absence of all brackets as belonging to the same class, the one associated with the absence of a bracket. Here, we also seek to recognize the bracket in place, in case a mechanical part is mounted, by classifying the bracket presence into 61 classes, one class per type of bracket. The training is carried out on the synthetic dataset from Table 24 (first column).

	Train	Val	Test	No. Cls	Pres./Abs.
Dataset D_{DA}	5856	732	732	61	Presence
Data with context	1404	184	183	61	Presence
Context alone data	1404	184	183	1	Absence

Table 24: Distribution of the synthetic dataset. No. Cls. - Number of classes, Pres./Abs. - Presence/Absence.

This model is evaluated on the two real datasets. The results are presented in Table 25. It reaches *Accuracy* of 94% on the Absence set. As for the Presence set, it is less successful, reaching an *Accuracy* of 69%, demonstrating an error rate of 31% on this set. Moreover, since the precision on this set does not reach 100%, we deduce that the model was confused among the represented presence classes. The last row of the table shows an

Accuracy of 71%, indicating a total confusion on the entire dataset at a level of 29%. The total precision of 90% indicates confusion between absence cases and cases of bracket presence.

	Acc.	Prec.	Rec.	F1-sc.	Nb. samples
Presence set	69%	95%	69%	78%	590
Absence set	94%	100%	94%	97%	33
Total	71%	90%	71%	75%	623

Table 25: Results on real acquisitions. The model predicts one of the 62 classes. The used metrics are *Accuracy* (Acc.), Weighted-*Precision* (Prec.), Weighted-*Recall* (Rec.) and Weighted-*F1-score* (F1-sc.)

6.4.1 Binarization of model output

In the last experiment, we use the model from the previous experiment, demonstrated in Section 6.4.

Note that this model is trained to differentiate between 62 classes: 61 presence classes and 1 absence class.

Additionally, in this experiment, we perform the binarization of the model output. Namely, we consider all predictions of presence cases as belonging to a single class named presence. Therefore, we only differently interpret results from the previous experiment, without any additional training. The results are presented in Table 26. On the Presence set, we observe that the *Accuracy* only increases by 1%, which means that there were 6 acquisitions of presence but incorrectly attributed to a wrong bracket type.

	Acc.	Prec.	Rec.	F1-sc.	Nb. acq
Presence set	70%	100%	70%	82%	590
Absence set	94%	100%	94%	97%	33
Total	71%	95%	71%	80%	623

Table 26: Results on real acquisitions after binarization of model output.

The results presented in Table 26 are shown in the form of an Absence/Presence confusion matrix in Table 27.

With a total of 444 point clouds correctly classified, this model is the best we have gotten. Moreover, with 177 False Positives (FP) and only 2 False Negatives (FN), it also shows the lowest number of FNs, not counting the previous model

label/pred	Presence	Absence	Nb. acq
Presence	413	177	590
Absence	2	31	33

Table 27: Results on real acquisitions after binarization of the model output.

that systematically classified every sample as an absence. This model is the first to reduce the number of FPs to under 200. Thus, with an FP rate of 30% and only 2 FNs, this model is currently the best for handling absence problem.

7 Conclusion

In this paper, we focused on the automation of the visual conformity check procedure on a complex aeronautical assembly, relying on deep neural networks, using CAD models as a reference, and 3D visual sensors for data acquisition.

We have shown that training based purely on synthetic 3D point clouds (generated from a 3D CAD model) could produce good results in differentiating between real point clouds of various mechanical parts.

We first selected 4 architectures and demonstrated that *PointCNN* was the most robust to occlusions and similarities between target objects, making it promising, with an *Accuracy* of 95% on real data.

Next, we wanted to analyze the impact of the context around the parts (brackets) by including it in training data. We concluded that the context did not bring any additional information for the classification of real point clouds, in case of differentiating between parts of different types.

In a complementary approach, we investigated whether the surrounding context could contribute to the detection of the absence of a bracket. We found that the context was misleading the network, causing many false positives, thus degrading the performances of classification. This experiment also revealed that *PointCNN* had difficulty in distinguishing between classes, by confusing their surroundings.

However, the model manages to detect the absence of a bracket in a point cloud. This last point is promising since the industrial objective is to be able not to miss the absence of a bracket, whereas recognizing the context of the bracket is not required.

With the previous results in mind, we can separate the detection of the presence/absence of a bracket from the detection of the wrong bracket mounted. A first model decides on the absence/presence dilemma; then a second model would classify only the presence clouds to check whether the correct bracket has been mounted in the correct location or it has been replaced by another one.

In absence recognition, the best model reaches 94% on the Absence set. A major obstacle for this method was its False Positive (FP) ratio of 30% - a proportion far too high to afford in industry. It should be noted that the results are dependent on the quality of acquisitions and the number of points acquired. However, the proposed solution is efficient on real acquisitions as long as the carrying robot is well-positioned and the acquisition is not faulty. The industrial need is that the acceptable error rate on Presence set, therefore the FP rate, must not exceed 10%. This percentage takes into account possible poor 3D reconstruction of the sensor leading to poor cloud quality. On the other hand, on the FNs we must ensure that we obtain 0 FNs.

We continue the work on the inspection of mechanical assemblies in the future. This task can be easily transformed into a task of object detection and point cloud segmentation from which we can detect if a mechanical part is mounted to the assembly or not.

8 Declatarions

8.1 Ethics approval and consent to participate

Not applicable

8.2 Consent for publication

Not applicable

8.3 Availability of data and materials

The data utilized in this study are coming from DIOTA company, i.e. from 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 DIOTA company.

8.4 Competing interests

The authors have no competing interests, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

8.5 Funding

Also, this work has been carried out within a PhD thesis co-funded by the French “Région Occitanie.” ID of the grant is 19008718 / ALDOCT-000622.

8.6 Authors' contributions

Conceptualization, ideation and methodology were conducted by A.B., I.J. and JJ.O.. The software was developed by A.B.. M.B. took part in advising and defining the industrial requirements. A.B. wrote the initial draft of the manuscript. V.D and I.J. rewrote the final version of the manuscript. JJ.O. reviewed and edited the manuscript. I.J. and JJ.O. were supervising the whole research. JJ.O. was responsible for funding acquisition and project administration. All authors have read and agreed to the published version of the manuscript.

8.7 Acknowledgements

This research work is being carried out within the framework of the joint research laboratory ”Inspection 4.0” between IMT Mines Albi/ICA and the company *Diota* specialized in the development of numerical tools for Industry 4.0. Also, this work has been carried out within a PhD thesis co-funded by the French “Région Occitanie.” We would like to thank the “Région Occitanie” for its financial support.

References

[1] DA Max and RC Graeber. Human error in maintenance. *Aviation Psychology in Practice*, 1:87–104, 1994.

[2] CG Drury. Human reliability in civil aircraft inspection. Technical report, STATE UNIV OF NEW YORK AT BUFFALO DEPT OF INDUSTRIAL ENGINEERING, 2001.

[3] H. Ben Abdallah. *Inspection d'assemblages aéronautiques par vision 2D/3D en exploitant*

la maquette numérique et la pose estimée en temps-réel. PhD thesis, Université de Toulouse - IMT Mines Albi, Février 2020.

[4] Hamdi Ben Abdallah, Jean-José Orteu, Benoit Dolives, and Igor Jovančević. 3d point cloud analysis for automatic inspection of aeronautical mechanical assemblies. In *Fourteenth International Conference on Quality Control by Artificial Vision*, volume 11172, pages 209–217. SPIE, 2019.

[5] Assya Boughrara, Igor Jovančević, Hamdi Ben Abdallah, Benoît Dolives, Mathieu Belloc, and Jean-José Orteu. Inspection of mechanical assemblies based on 3D Deep Learning approaches. In *QCAV'2021 - 15th International Conference on Quality Control by Artificial Vision*, volume 11794 of *Proceedings of SPIE*, page 8 p., Tokushima (online), Japan, May 2021.

[6] I. Mikhailov, I. Jovančević, N. I. Mokhtari, and J.-J. Orteu. Classification using a 3D sensor in a structured industrial environment. *Journal of Electronic Imaging*, 29(4):041008, July 2020.

[7] Assya Boughrara, Igor Jovančević, Jean-José Orteu, and Mathieu Belloc. Inspection of mechanical assemblies based on 3D deep learning segmentation. In Igor Jovančević and Jean-José Orteu, editors, *Sixteenth International Conference on Quality Control by Artificial Vision*, volume 12749, page 127490Y. International Society for Optics and Photonics, SPIE, 2023.

[8] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

[9] Paolo Russo, Fabio Maria Carlucci, Tatiana Tommasi, and Barbara Caputo. From source to target and back: symmetric bi-directional adaptive gan, 2017.

[10] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. 2015.

[11] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation, 2017.

[12] He Wang, Zetian Jiang, Li Yi, Kaichun Mo, Hao Su, and Leonidas J. Guibas. Rethinking sampling in 3d point cloud generative adversarial networks. *CoRR*, abs/2006.07029, 2020.

[13] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. Jun 2015. arXiv:1406.6909 [cs].

[14] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. Aug 2017. arXiv:1603.09246 [cs].

[15] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. Mar 2018. arXiv:1803.07728 [cs].

[16] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. Jan 2016. arXiv:1505.05192 [cs].

[17] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. Apr 2017. arXiv:1611.06646 [cs].

[18] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. Jul 2016. arXiv:1603.08561 [cs].

[19] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. Oct 2015. arXiv:1505.00687 [cs].

[20] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning

on point sets for 3d classification and segmentation, 2016. cite arxiv:1612.00593Comment: CVPR 2017.

[21] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. PointCNN. *CoRR*, abs/1801.07791, 2018.

[22] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv : Flexible and deformable convolution for point clouds. *CoRR*, abs/1904.08889, 2019.

[23] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics, 2019.

[24] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. *CoRR*, abs/1803.11527, 2018.

[25] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[26] Hugues Thomas, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Yann Le Gall. Semantic classification of 3d point clouds with multiscale spherical neighborhoods. *CoRR*, abs/1808.00495, 2018.

[27] Chenxi Huang, Ganxun Tian, Yisha Lan, Yonghong Peng, E. Y. K. Ng, Yongtao Hao, Yongqiang Cheng, and Wenliang Che. A New Pulse Coupled Neural Network (PCNN) for Brain Medical Image Fusion Empowered by Shuffled Frog Leaping Algorithm. *Frontiers in Neuroscience*, 13:210, 2019.

[28] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920. IEEE Computer Society, 2015.

[29] Pegah Kamousi, Sylvain Lazard, Anil Maheshwari, and Stefanie Wuhrer. Analysis of Farthest Point Sampling for Approximating Geodesics in a Graph. *CoRR*, abs/1311.4665, 2013.