Inspection of mechanical assemblies based on 3D Deep Learning approaches

Assya Boughrara, Igor Jovančević, Hamdi Ben Abdallah, Benoît Dolives, Mathieu Belloc, Jean-José Orteu

Inspection of mechanical assemblies based on 3D Deep Learning approaches

Assya Boughrara[a,b], Igor Jovančević[b,*], Hamdi Ben Abdallah[a], Benoît Dolives[b], Mathieu Belloc[b], and Jean-José Orteu[a]

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

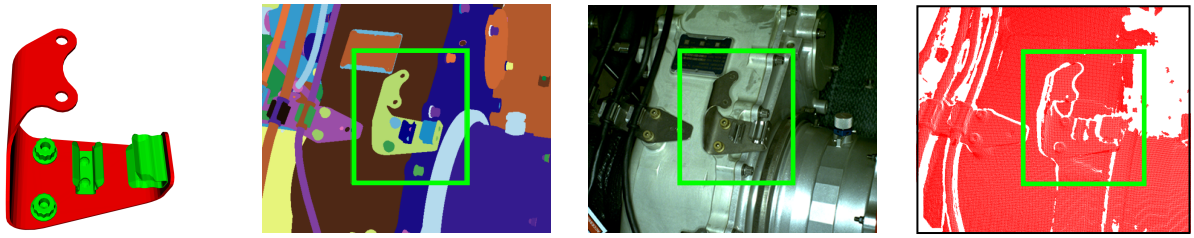
ABSTRACT

Our 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. 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 networks. Computer Aided Design (CAD) model of the mechanical assembly to be inspected is available, which is an important asset of our approach. Our deep learning models are trained on synthetic and simulated data, generated from the CAD models. Several networks are trained and evaluated and results on real clouds are presented.

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

1. INTRODUCTION

Our aim is a robotized inspection of complex aeronautical mechanical assemblies, typically an assembled aircraft turbine or a turbine being assembled, for which we have the digital Computer Aided Design (CAD) model. The challenge is to verify that the elements constituting the assembly are present and in the right positions, as specified by CAD model. A 3D scanner carried by a robot arm provides 3D point clouds. The CAD model of the mechanical assembly is considered the reference, i.e. ground truth. During the inspection, we focus our attention on mechanical elements of type *bracket* (see Fig. 1a). The robotized inspection platform designed by DIOTA, consists of a mobile robot equipped with three sensors (two cameras and a 3D scanner) mounted on a robot effector arm. A wide-field camera is used for estimating a relative pose between our sensors and an assembly. A narrow-field camera has a role to capture details and observe the elements very finely (see Fig. 1c). Finally a structured light stereo 3D scanner is digitizing the observed area to the form of point clouds (see Fig. 1d).



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

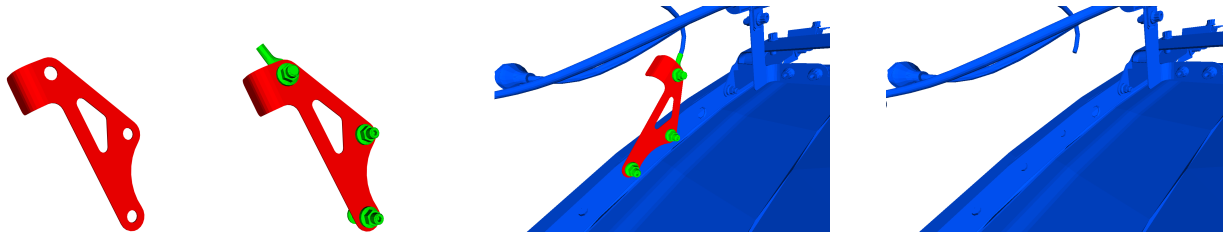
Figure 1: Summary of the available information: (a) bracket CAD model; (b) synthetic image generated from the CAD model; (c) image acquisition; (d) 3D point cloud. The green bounding box shows the bracket in the scene.

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

An in-house developed localization module provides an approximate bracket location in the point cloud scene given by the 3D scanner.

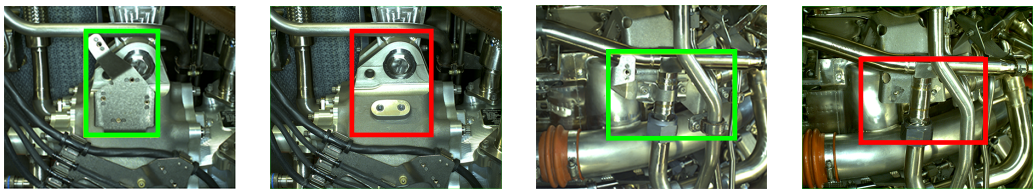
Our inspection task is thus reduced to a pure classification, with no detection needed, assuming the localization module is sufficiently accurate. To tackle the task, we rely on deep learning classification of 3D point clouds.

The inspection process consists of the following steps: (1) verify that a bracket is present at the expected location, or report absence of any bracket (2) determine if a wrong bracket has been mounted, to report the inconsistency with the CAD model. In fact, in real conditions, we always see brackets with their close context *, nicknamed *bracket fusion* (Fig. 2b). Some examples of *bracket fusion* are presented and denoted with green bounding boxes in Fig. 3.



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

Figure 2: Examples of CAD models : (a) example of bracket alone (in red), (b) bracket and its close context (in green), nicknamed *bracket fusion*, (c) *bracket fusion* within its extended context (in blue), (d) extended context alone



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

Figure 3: Examples of brackets in their industrial context. A green bounding box means the bracket is present, a red bounding box means the bracket is missing. In (c) we can see a pipe which causes some occultation of the bracket.

Supervised deep learning model requires a large set of annotated data. Annotation is usually performed manually, which is tiring and time consuming. Moreover, in the aeronautical context, where each type of assembly has its specific brackets, manual annotation seems even less feasible. Therefore, we propose the exploitation of CAD models to provide a training dataset for our classification models.

The main goal is to find an experimental protocol allowing to train a deep classification model exclusively on synthetic data generated from the CAD model, while guaranteeing a good inference performance on real 3D data acquired by the 3D scanner in real situations. Real situations could be challenging due to 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 occultations in real point clouds. Furthermore, there are disposable elements such as caps which are not present in the CAD model.

The current work is a continuation of our preliminary work[1] which dealt with the classification of 3D point clouds after training on real data. whereas our current work is using synthetic data for training and real data for evaluating the models.

*bracket alone cannot be mounted on the engine without its surrounding (screws, clamps, etc.), so the scanner never sees the bracket alone, without its close context.

2. PREVIOUS WORKS

Our developed application is based on a vision-based sensor system mounted on an automated articulated arm. Similar works have been reported in[2] dealing with the visual inspection of welds for automotive manufacturing and in[3] which describes a robot arm equipped with a 3D optical scanner for the inspection of mechanical components. For our study, we focus on classification models based purely on point cloud. We ignore normals as complementary features because they are only available for artificial data.

In 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 neighborhood. *PointCNN*[4] learns a χ-transformation for each neighborhood of points, thus evidencing local patterns. Similary, *KPConv*[5] proposed a weighted decomposition by neighborhood of points, which does not limit the number of kernels for decomposition to improve flexibility. The operator *ShellConv* of *Shellnet*,[6] builds concentric spherical shells on each local neighborhood based on a statistical heuristic, which enables a lower computational cost than *PointCNN*.[4] *SpiderCNN*[7] differs by applying weights on the points according to their order in point cloud, which makes this structure non-invariant to the order of the inputs. On the contrary, *KPConv* is applying weights to points according to their *spatial* positions which allows invariance to the order of the inputs. In another structure form, the component *Edgeconv* of *DGCNN*[8] encodes convolutions by graphs, thus allowing local geometrical structures extraction, and also remaining invariant to the permutation of inputs. Note, *KPConv* is the only one not to model the neighborhoods with *k-Nearest Neighbors (KNN)*, choosing the *radius neighborhoods*, ensuring the robustness of the convolutions on variable densities, as argued in.[9]

Sparsity is a key notion in point clouds. *PointCNN*[4] introduces sparsity to data but also to its convolution kernels. This allows to 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*[10] 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 our problem: *ShellNet*, *PointCNN*, *PCNN*, *KPCONV*, *DGCNN*. All of them belong to parametric convolutions family except *DGCNN* which belongs to the family of convolution graphs.We based our choice on these networks' performance reported on the ModelNet40,[11] a freely available dataset. Architectures of *PointCNN*, *ShellNet* and *KPConv* are descendants of the unmissable *PointNet*[12] network.

3. APPROACH

For our inspection problem, we are developing a 3D deep-learning-based multi-class classification approach whose learning is based purely on synthetic data, i.e. synthetic 3D point clouds generated from the CAD model.

3.1 Generation of synthetic clouds

To train our models, we have generated 3D point clouds databases using the 3D CAD models (Fig. 2a and 2b) of 61 different brackets, each corresponding to one class. We developed a z-buffer-based method combined with Fibonacci sphere for space sampling, to generate synthetic acquisitions from theoretical points of view. This method takes into account self-occultation of the element, just as the real scanner does. In order to respect the security conditions, our virtual scanner is positioned at a working distance of 60 cm from the centroid of the bracket, just as the real scanner. Fig. 4 shows a visualisation of Fibonnacci Sphere centered on bracket and some examples of synthetic point clouds provided by our method.

For each *bracket fusion* (Fig. 2b), we generate 40 different synthetic clouds, out of which we use 32 for training, 4 for validation and 4 for testing. This ensures that we have different clouds in each dataset. To be closer to real acquisitions, we simulate occultation and noise. Data-augmentation is applied on each cloud by adding an artefact such as an artificial cable for instance. The artefacts could occlude the *bracket fusion* or not. We also add Gaussian noise before the z-buffer captures the point cloud. The total distribution of our clouds is summarized in Table 1. In order to keep a fixed number of points per synthetic cloud, we downsample all our clouds to 2048 points with the *Farthest Point Sampling* (FPS) method.[13]

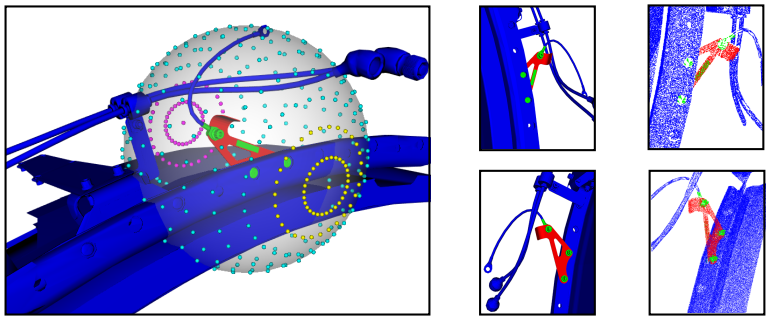


Figure 4: Left: a visualisation of Fibonnacci Sphere centered on *bracket fusion*; Middle: examples of synthetic images generated from the CAD model; Right: examples of synthetic point clouds provided by our method

	Train	Val	Test
Synthetic clouds	5856	732	732

Table 1: Distribution of synthetic learning dataset on *bracket fusion* with data-augmentation.

3.2 Training models with synthetic data

Preliminary experiments allowed us to better adapt our synthetic data generation to the needs of our deep learning models. First, learning on the synthetic clouds of brackets only (Fig. 2a) did not allow a good generalization on the synthetic *bracket fusion* clouds (Fig. 2b). The inverse process shows excellent results (nearly 97% on the test data of type bracket). Moreover, *bracket fusion* models correspond to the reality as explained in Sec. 1. Hence, we have decided to perform all our training procedures on the models of *bracket fusion*. Second, our experiments showed that the performance was not better if number of theoretical views on Fibonacci sphere is higher than 40. We went up to 160 different views per bracket. Therefore, the number of views was fixed to 40.

By training *PCNN*, *PointCNN* and *Shellnet* using hyper-parameters recommended for ModelNet40 dataset,[11] we obtained accuracy of 97% on our test synthetic dataset. However, for *DGCNN* we had to adjust hyper-parameters to our dataset to obtain good performance and particularly to ensure the convergence. Namely, we reduced the batch size to 20 and assigned k=25 for k-nearest-neighbor function. Note that for the training phase we have used the OLYMPE super-computer (UMS CNRS 3667) [†] with power of 1.365 Pflops/s.

4. RESULTS

After training our 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. 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 are able to calculate a 3D region of interest and apply it on the full real cloud, in order to crop the part which is of interest to us. Further, we apply a *FPS* and our input cloud is ready for inference.

We have evaluated four out of five selected networks: *Shellnet*, *PointCNN*, *DGCNN*, *PCNN*. The results are expressed using the following average metrics: weighted-Precision, weighted-Recall and weighted F1-score. The Accuracy corresponds to total Accuracy.

4.1 Four networks trained on 61 synthetic classes with present bracket

First we trained and tested 4 selected networks on synthetic data coming from 61 classes, i.e. 61 different brackets. The distribution of the dataset is shown in Table 1. In all the synthetic clouds, one of the 61 brackets was present. Evaluation on the 732 synthetic clouds has shown an overall accuracy of near 100% for all four models. This is expected since our synthetic test data are similar to our synthetic training data.

[†]https://www.calmip.univ-toulouse.fr/spip.php?article582

We then evaluated the same models on 590 real 3D point cloud acquisitions, divided into 20 different classes. It should be noted that all 590 real clouds contained a good bracket, i.e. there was no samples with missing bracket and there was no replaced brackets. This set made of real data will further be nicknamed Presence set. The results per class are presented in Fig. 5 and overall metrics for each model can be seen in Table 2.

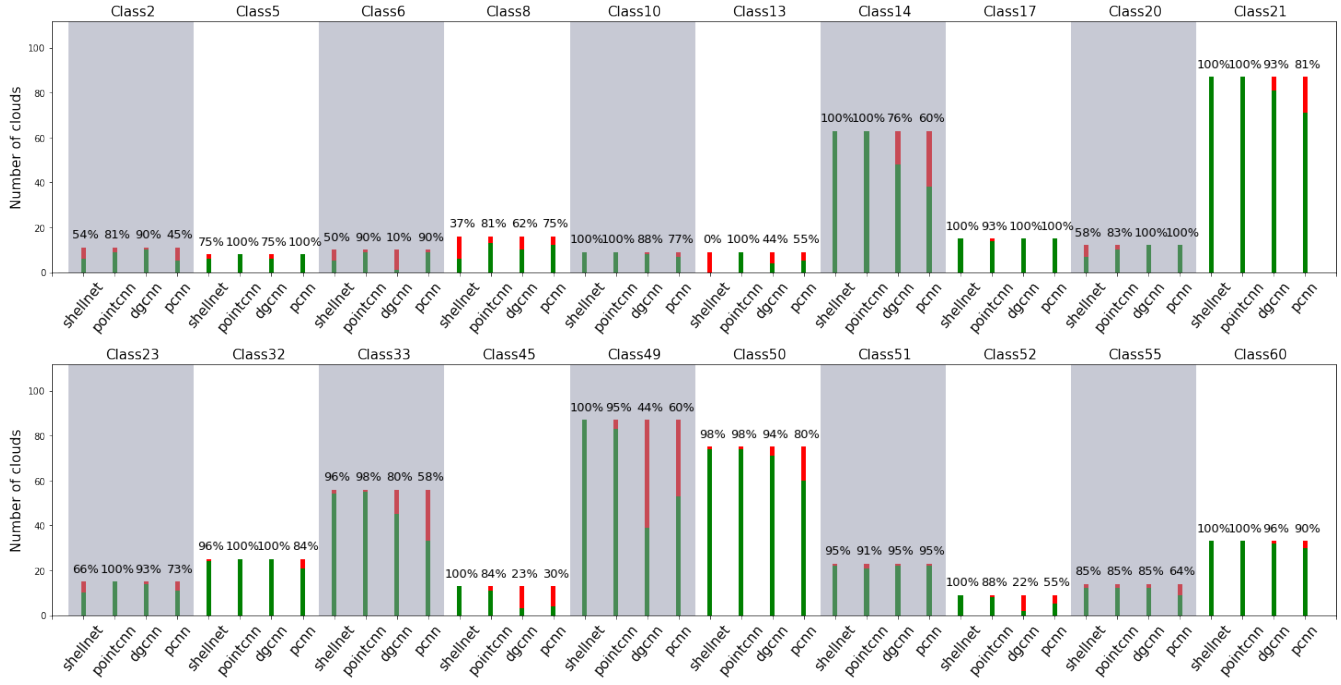


Figure 5: Percentage of correct classifications of each model, per class, evaluated on real data. In green: the percentage of acquisitions correctly classified by a model. In red: the percentage of acquisitions wrongly classified by a model. On Y axis: the total number of tested acquisitions. On X axis: four classification models. White and grey vertical bands are added to separate the classes for better readability.

	ShellNet			*PointCNN*			*DGCNN*			*PCNN*			
	Prec.	Rec.	F1-sc.	Prec.	Rec.	F1-sc.	Prec.	Rec.	F1-sc.	Prec.	Rec.	F1-sc.	Nb. acq
Avg.	95%	91%	93%	98%	96%	97%	95%	77%	84%	94%	72%	81%	590

Table 2: Results for each model based on the average of the metrics over all 20 real acquisition classes. The average metrics used are weighted-precision (*Prec.*), weighted-recall (*Rec.*) and weighted F1-score (*F1-sc.*).

We observe in Fig. 5 that *PointCNN* always shows an accuracy higher than 80%. For classes with many samples (e.g. classes 21, 49, 50), we manage to get good performances on most models. Preliminary results have shown that our approach can be robust and generalises well on real acquisitions. However, *DGCNN* demonstrates high sensitivity to occultation by cables and some disposable elements not present in CAD

ShellNet shows difficulties to discriminate between similar brackets. As it can be seen, the accuracy for class 13 is 0%. With deeper analysis, we observed that class 13 was systematically confused with class 50 which has an appearance very similar to class 13. This behaviour invalidates this network for our study.

PCNN shows the most instability. It is very sensitive to the localization error, even the smallest, and to occultation 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*.

Having in mind previous observations, we decided to focus on *PointCNN* network which exhibited the most stable behaviour.

An improvement axis aroused. During *bracket fusion* data generation (as in Fig. 2b), some views generated in the synthetic environment may be inaccessible to the camera in reality. Hence, we unwillingly generate unrealistic data which may cause a bias in our experiments. This happens due to the fact that we did not take into account the context around the bracket. By involving wider context (cables, pipes etc.) during data generation (as shown in Fig. 2c), we are able to remove these unrealistic data from training set. Additionally, including context helped us to decrease confusion between classes and be more robust to occultation. Unfortunately, because of strong occultation by the extended context, there is not many points of view from which brackets are visible. Nevertheless, we use these data to increase our dataset from Table 1. The distribution of total dataset used for second experiment is presented in Table 3a. Results of testing on Presence set are presented in Table 3b for our two models based on *PointCNN*. This experiment shows that the context does not improve performance.

	Train	Val	Test
Synth. dataset (Fig. 2b)	5856	732	732
Synth. dataset (Fig. 2c)	1408	187	187
Total synth. dataset	7264	919	919

(a) Distribution of synthetic learning dataset

	Acc.	Prec.	Rec.	F1-sc.
PointCNN (1)	96%	98%	96%	97%
PointCNN (2)	95%	98%	95%	96%

(b) Results on Presence set (real data)

Table 3: Results of models based on *PointCNN*: (1) is the model trained on the *bracket fusion* dataset (as in Fig. 2b); (2) is the model trained on the *bracket fusion* dataset augmented by data generated with bracket context (as in Fig. 2c)

4.2 *PointCNN* trained on 61 synthetic classes with present bracket and 61 synthetic classes with missing bracket

Up to this point, we have trained our models with synthetic in which there was a bracket present. We have tested our models with synthetic and real data in which there was a bracket present. We have not generated any clouds with missing bracket. In order to simulate absence of a bracket, we generated clouds of context only, for each bracket (as in Fig. 2d). For the generation process, we have 122 classes: 61 classes with present bracket and 61 classes with no bracket (context only). When filtering those context clouds, we keep the same points of view which we keep for the bracket clouds, i.e. those points of view from which the bracket is visible enough, when present. Such an extended synthetic dataset has a distribution as shown in Table 4. Same as in previous experiment, we train *PointCNN* network with this extended dataset. We test the trained model on synthetic data; results can be seen in Table 4. It can be noticed that the model achieve 88% accuracy level on synthetic data with context, which shows that the context alone is not discriminant enough for classification.

	Learning set		**Results on Test set**				
Synthetics clouds set	Train	Val	Test	Acc.	Prec.	Rec.	F1-sc.
Synth. dataset *bracket fusion* (Fig. 2b)	5856	732	732	99%	99%	99%	99%
Synth. dataset *bracket fusion* + context (Fig. 2c)	1408	187	187	88%	97%	88%	91%
Synth. dataset context only (Fig. 2d)	1404	184	183	88%	98%	87%	91%

Table 4: Distribution of synthetic learning datasets and the results on synthetic Test set

We have 34 real acquisitions representing the situation where the bracket is missing, divided into 14 different classes. This set will further be nicknamed Missing set. We evaluate our model on our total real acquisitions set; results are presented in Table 5. Both results obtained on synthetic data as well as those obtained on real data are not good enough. So we evaluated the false positives proportion on Presence set (see Fig. 6). False positive (false alarm) is an outcome in which our model raises an alarm that a bracket is missing, whereas the bracket is present. Fig. 6 shows a very high proportion of false alarms on Presence set, precisely 230 acquisitions in total out of 590 (i.e 39% of Presence set). This huge proportion of false positives explains the weak scores that are reported in Table 5, but also the huge precision in this set. The huge precision for Presence set is explained by the fact that the model does not confuse often the presence situations between them. These results suggest that the context is a source of confusion for our model.

	Acc.	Prec.	Rec.	F1-sc.	Nb. acq
Presence set	49%	92%	49%	55%	590
Missing set	55%	56%	55%	54%	34

Table 5: Results on real clouds acquisitions. Accuracy (*Acc.*) corresponds to total accuracy. The metrics used: weighted-precision (*Prec.*), weighted-recall (*Rec.*) and weighted F1-score (*F1-sc.*).

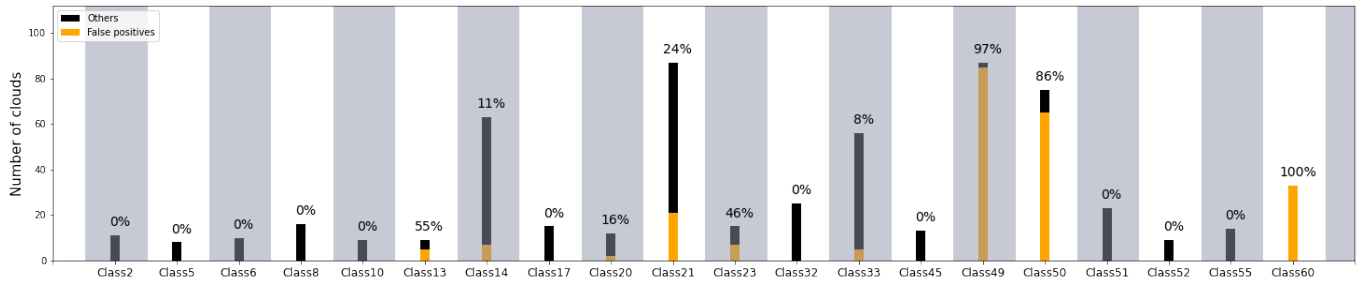


Figure 6: Percentage of false positive classifications (false alarm) of each class, evaluated on real data with present bracket. In orange: the false positive percentage by class. In black: complement to 100%. On Y axis: the total number of tested clouds. On X axis: classes of tested real clouds. White and grey vertical bands are added to separate the classes for better readability.

The accuracy by class on Missing set is shown in Fig. 7. These results confirm the high error rate (45%) already presented in second row of Table 5. By closely examining the data, we observed that out of 45%, there is only 10% of false negatives (i.e. 4 samples from Missing set), while confusion between different context classes represents the remaining 35% of Missing set. Nevertheless, a primary industrial need that motivated our work, was an ability to precisely recognize the absence anomaly, without necessarily deciding among context classes. In order to adapt to this requirement, we decided to consider all the absence classes as a single class. In this new mode, we just need to know if the model recognizes the specified bracket. If not, our solution will answer "bracket missing". Fig. 8 presents the results.

These results are much better, and this interpretation raises the total accuracy to 88% on the Missing set, showing a good ability of the model to detect the absence of a bracket.

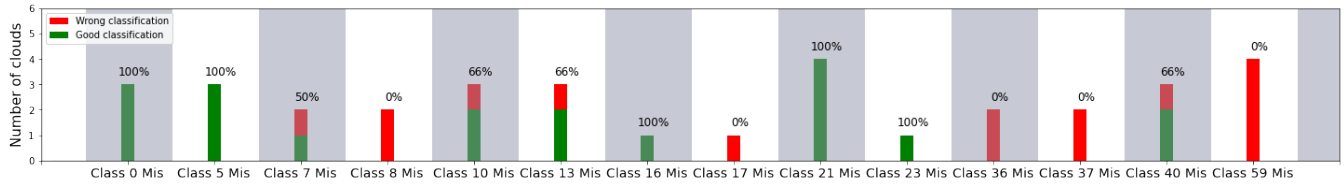


Figure 7: Percentage of correct classifications by class, evaluated on Missing set. In green: the percentage of clouds correctly classified. In red: the percentage of clouds wrongly classified. On Y axis: the total number of tested clouds. On X axis: classes of tested real clouds.

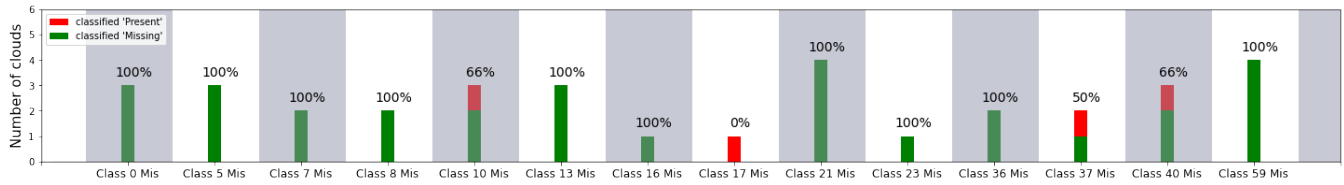


Figure 8: Percentage of correct "missing situation" detected by class, evaluated on Missing set. In green: the percentage of clouds correctly detected as "missing situation". In red: the percentage of wrongly classified clouds. On Y axis: the total number of tested clouds. On X axis: classes of tested real clouds.

5. CONCLUSION AND FUTURE WORK

In this paper we have shown that a learning based purely on synthetic 3D point clouds (generated from a 3D CAD model) could produce good results in classification on real point clouds acquired by a 3D scanner. We first selected 4 networks and demonstrated that *PointCNN* was the most robust to occultations and similarities between target objects, making it very promising for our inspection process on complex aeronautical parts. In a second step, we wanted to analyze the influence of the context around our parts (brackets) by including it into our training data. This led us to observe that the context did not bring any information for the classification of our real data. In a complementary approach, we investigated whether the context could contribute to the detection of absence of 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 classes by confusing their contexts.

However the model manages to detect the absence of a bracket in a point cloud. This last point is very promising. Indeed, the industrial objective is to be able to detect the absence of a bracket, recognizing the context of the bracket is not required. An in-depth analysis of the cases of context/presence confusion will be carried out to improve the absence detection.

With the previous results in mind, our plan is to separate the detection of the presence/absence of a bracket from the detection of the situation corresponding to a wrong bracket mounted. A first model would decide on absence/presence dilemma; then a second model would classify only the presence clouds to check whether the right bracket has been mounted in the right location or it has been replaced by another one.

ACKNOWLEDGMENTS

This research 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] Mikhailov, I., Jovancevic, I., Mokhtari, N. I., and Orteu, J.-J., "Classification using a 3D sensor in a structured industrial environment," *Journal of Electronic Imaging* **29**, 041008 (July 2020).

[2] Wörn, H., Längle, T., and Gauß, M., "Arikt:adaptive robot based visual inspection.," *KI* **17**, 33– (01 2003).

[3] Raffaeli, R., Mengoni, M., and Germani, M., "Context dependent automatic view planning: the inspection of mechanical components," *Computer-aided Design and Applications* **10**, 111–127 (2013).

[4] Li, Y., Bu, R., Sun, M., and Chen, B., "PointCNN," *CoRR* **abs/1801.07791** (2018).

[5] Thomas, H., Qi, C. R., Deschaud, J., Marcotegui, B., Goulette, F., and Guibas, L. J., "Kpconv : Flexible and deformable convolution for point clouds," *CoRR* **abs/1904.08889** (2019).

[6] Zhang, Z., Hua, B.-S., and Yeung, S.-K., "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," (2019).

[7] Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y., "Spidercnn: Deep learning on point sets with parameterized convolutional filters," *CoRR* **abs/1803.11527** (2018).

[8] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M., "Dynamic graph CNN for learning on point clouds," *CoRR* **abs/1801.07829** (2018).

[9] Thomas, H., Deschaud, J., Marcotegui, B., Goulette, F., and Gall, Y. L., "Semantic classification of 3d point clouds with multiscale spherical neighborhoods," *CoRR* **abs/1808.00495** (2018).

[10] Huang, C., Tian, G., Lan, Y., Peng, Y., Ng, E. Y. K., Hao, Y., Cheng, Y., and Che, W., "A New Pulse Coupled Neural Network (PCNN) for Brain Medical Image Fusion Empowered by Shuffled Frog Leaping Algorithm," *Frontiers in Neuroscience* **13**, 210 (2019).

[11] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J., "3D ShapeNets: A deep representation for volumetric shapes.," in [*CVPR*], 1912–1920, IEEE Computer Society (2015).

[12] Qi, C. R., Su, H., Mo, K., and Guibas, L. J., "Pointnet: Deep learning on point sets for 3d classification and segmentation," (2016). cite arxiv:1612.00593Comment: CVPR 2017.

[13] Kamousi, P., Lazard, S., Maheshwari, A., and Wuhrer, S., "Analysis of Farthest Point Sampling for Approximating Geodesics in a Graph," *CoRR* **abs/1311.4665** (2013).