Inspection d'assemblages mécaniques par une approche Deep Learning 3D : résultats préliminaires

Assya Boughrara, Igor Jovancevic, Hamdi Ben Abdallah, Benoit Dolives, Mathieu Belloc, Jean-José Orteu

Inspection d'assemblages mécaniques par une approche Deep Learning 3D : résultats préliminaires

A. Boughrara[1,2] I. Jovančević[2] H. Ben Abdallah[1] B. Dolives[2] M. Belloc[2] J.-J. Orteu[1]

[1] Institut Clément Ader (ICA) ; Université de Toulouse ; CNRS, IMT Mines Albi, INSA, UPS, ISAE ; Campus Jarlard, 81013 Albi, France
[2] DIOTASOFT, 201 Pierre and Marie Curie Street, 31670 Labège, France
<assya.boughrara@mines-albi.fr>

Résumé

Nos travaux de recherche sont menés dans le cadre du laboratoire de recherche commun "Inspection 4.0" entre IMT Mines Albi/ICA et la société DIOTA spécialisée dans le développement d'outils numériques pour l'industrie 4.0. Dans cet article, nous nous intéressons au contrôle de conformité d'ensembles mécaniques aéronautiques complexes (typiquement un moteur d'avion en fin ou en milieu de chaine d'assemblage). Un scanner 3D porté par un bras de robot permet l'acquisition de nuages de points 3D pour la phase d'inspection. Nous avons à notre disposition un modèle CAO de l'assemblage mécanique à inspecter, et c'est ce modèle qui guidera notre démarche. Nous mettons en œuvre des techniques de classification 3D par Deep Learning. Ces modèles d'apprentissage profond sont formés sur des données synthétiques et simulées, générées à partir des modèles CAO. Plusieurs approches sont proposées et des résultats sur des acquisitions réelles sont présentés.

Mots-clés

Inspection robotisée, Contrôle qualité, Modèle CAO, Deep Learning 3D, Scanner, Nuages de points 3D, Aéronautique.

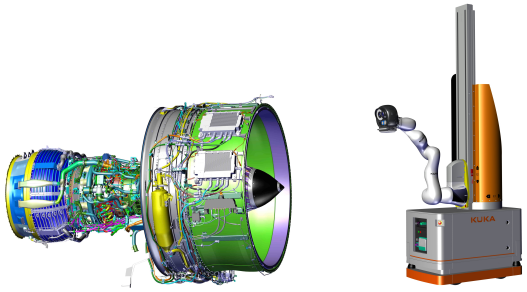
Abstract

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 article, we are interested in the 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 3D point clouds for the inspection phase. We have at our disposal the CAD model of the mechanical assembly to be inspected, and this CAD model will guide our approach. We employ state of the art 3D deep learning classification techniques. These deep learning models are trained on synthetic and simulated data, generated from the CAD models. Several approaches are proposed and some results on real acquisitions are exposed.

Keywords

Robotized Inspection, Quality Control, CAD Model, 3D Deep Learning, Scanner, 3D Point Clouds, Aeronautics.

1 Introduction



(a) (b)

FIGURE 1 – (a) Modèle CAO d'un assemblage mécanique complexe à inspecter ; (b) plateforme d'inspection robotisée équipée d'un capteur intégrant deux caméras et un scanner 3D.

Le milieu industriel, en particulier celui de l'aéronautique, recherche des solutions fiables d'inspection automatisée en fin ou en milieu de chaîne de production. L'application que nous avons développée utilise un robot industriel équipé de capteurs visuels, comme celui présenté par [1] pour l'inspection visuelle des soudures pour la construction automobile, ou le bras robotique équipé d'un scanner optique 3D pour l'inspection des composants mécaniques présenté dans [2]. Nos travaux portent sur le contrôle robotisé d'assemblages mécaniques aéronautiques complexes (typiquement une turbine d'avion assemblée ou en cours d'assemblage ; cf. Fig. 1a). Nous disposons de la maquette numérique (modèle CAO 3D) de la pièce à contrôler, qui fournit son état de référence (état souhaité). La pièce est constituée de différents éléments (typiquement des supports [1] ;

1. On appelle "support" des petites pièces métalliques permettant essentiellement de fixer des câbles ou de faire tenir ensemble d'autres éléments.

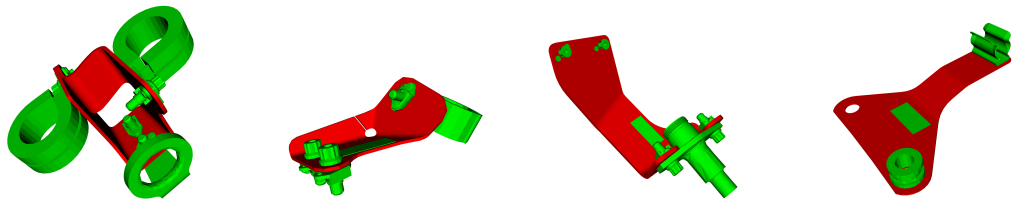


FIGURE 2 – Quelques exemples de modèles CAO de supports

cf. Fig 2) qui doivent être contrôlés. La maquette CAO, qui guide le processus d'inspection, nous permet de connaître a priori quel support est en cours d'inspection. Notre objectif est de déterminer :

- si le support attendu est monté
- si un autre support que le support attendu est monté (erreur de montage)
- si aucun support n'est monté (erreur de montage)

Notre plateforme d'inspection robotisée (cf. Fig. 1b) est équipée d'un capteur composé de deux caméras et d'un scanner 3D. Une caméra de suivi à grand champ permet de positionner précisément le bras articulé par rapport à la pièce contrôlée. Une caméra à champ de vision réduit, dite caméra d'inspection, permet de capturer les détails (cf. Fig. 3c). Enfin, un capteur stéréo à lumière structurée permet d'acquérir des nuages de points 3D (cf. Fig. 3d).

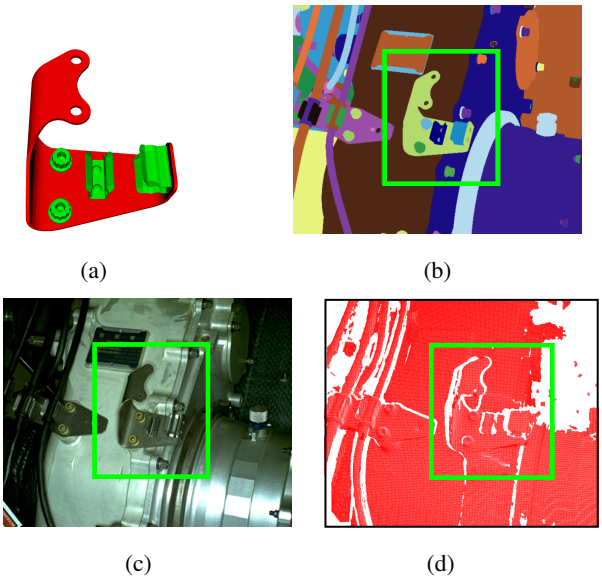


FIGURE 3 – Résumé des informations dont nous disposons : (a) modèle CAO du support ; (b) image synthétique générée à partir du modèle CAO ; (c) image acquise par la caméra d'inspection ; (d) nuage de points 3D. Le rectangle vert montre le support dans la scène.

Un module de localisation développé par DIOTA [3] fournit la pose de la caméra par rapport à l'élément à contrôler, définissant une bounding box 3D pour extraire l'élément à inspecter du nuage de points 3D fourni par le scanner. Nous avons décidé de traiter notre tâche d'inspection comme une tâche de classification (plusieurs types de support à contrôler ; cf. Fig. 2), que nous nous proposons de réaliser via des approches Deep Learning sur des nuages de points 3D. À noter que chaque nuage se voit attribuer une classe correspondant au support considéré. À aucun moment nous n'abordons d'aspect de classification par segmentation de points.

Un modèle d'apprentissage supervisé requiert un large jeu de données annotées, généralement à la main, ce qui est chronophage. En aéronautique, où chaque moteur dispose de pièces spécifiques, l'annotation manuelle paraît peu réalisable. Nous proposons donc d'exploiter les modèles CAO 3D des éléments à contrôler pour fournir des jeux de données d'entraînement synthétiques pour nos modèles de classification.

L'objectif principal est de définir un protocole expérimental permettant d'entraîner un modèle de classification profond exclusivement sur des données synthétiques générées à partir du modèle CAO, tout en garantissant une bonne performance d'inférence sur des données 3D réelles acquises par le scanner 3D en situation réelle. Les situations réelles peuvent être difficiles pour plusieurs raisons. Outre les difficultés typiques dues à la variation de l'éclairage et de l'écart entre les domaines, certains éléments non rigides de l'assemblage, tels que les câbles, peuvent changer légèrement de place et produire des occultations dans les nuages de points réels. En outre, certains éléments jetables, tels que les bouchons, ne sont pas présents dans les modèles CAO.

À noter que ce travail est une poursuite de notre travail préliminaire [4] qui traitait de la classification de nuages de points 3D après entraînement sur des données réelles. Dans nos travaux actuels nous utilisons des données synthétiques pour l'entraînement et des données synthétiques ou réelles pour l'évaluation des modèles.

2 Éléments bibliographiques sur le Deep Learning 3D

Notre scanner nous fournissant des nuages de points 3D, nous souhaitons exploiter des méthodes de *Deep Learning* adaptées à la classification de nuages de points 3D. Nous ignorons les normales comme caractéristiques complémentaires des points car elles ne sont disponibles que pour les données synthétiques et nous ne souhaitons pas les calculer

pour les données réelles car c'est une opération coûteuse et pourrait être une source d'imprécision.

Parmis les réseaux traitant des nuages de points 3D, une famille incontournable est celle des convolutions paramétrées. Usuellement, ces réseaux associent aux kernels paramétrés les points de façon individuelle, mais de nouveaux réseaux viennent raisonner par voisinage, comme *PointCNN* [5], *KPConv* [6], *Shellnet* [7] ou *PCNN* [8]. Chacun de ces réseaux représente les voisinages de façon différentes. *PointCNN* apprend une χ-transformation pour chaque voisinage de points, mettant ainsi bien en évidence des patterns locaux, là où *KPConv* propose une décomposition pondérée par voisinage de points. *PCNN*, est la structure se rapprochant le plus de *KPConv*. Toutefois, sa définition attribue des poids à des points et non à des voisinages, rendant le calcul convolutif quadratique sur le nombre de points. De plus, *PCNN* utilise une fonction de corrélation gaussienne, là où *KPConv* utilise une corrélation linéaire facilitant la back-propagation.

SpiderCNN se différencie en appliquant des pondérations sur les points suivant l'ordre des entrées, rendant ainsi cette structure non invariante à l'ordre des entrées. *KPConv* permet une invariance à l'ordre des entrées en présentant des poids localisés dans l'espace. De même, le composant *Edgeconv* de *DGCNN* [9], encode des convolutions par graphes, lui permettant ainsi d'extraire les structures géométriques locales en restant invariant à la permutation des données.

KPConv est le seul de ces réseaux à ne pas modéliser les voisinages avec les *k-Nearest Neighbors (KNN)*, en préférant les *radius neighborhoods*, assurant la robustesse des convolutions sur des densités variables [10]. De même, *PointCNN* s'ajuste au mieux en intégrant à la fois la sporadicité à ses données mais également à ses noyaux de convolution, allégeant ainsi les calculs. *KPConv* a également recours à une stratégie de sous-échantillonnage allégeant à la fois la densité et le coût en calcul, sans contraindre ses kernels à recevoir un nombre fixe de points.

L'opérateur, $ShellConv$ de *Shellnet*, envisage les voisinages différemment, en construisant des coquilles sphériques concentriques sur chaque voisinage local en se basant sur une heuristique statistique. De plus, ce réseau présente un temps de calcul plus faible que *PointCNN*, en raison d'une structure de réseau plus simple, nécessitant ainsi moins de calculs car purement basé sur un calcul statistique, contrairement au composant χ-transformation devant apprendre une matrice de transformation, opération nécessitant des calculs laborieux.

Après cette étude bibliographique, nous avons sélectionné les méthodes de classification de points 3D *Shellnet*, *PointCNN*, *PCNN*, *KPConv* et *DGCNN*. Tous correspondent à des convolutions paramétriques, sauf *DGCNN* qui appartient à la famille des graphes de convolutions. À priori tous ces réseaux sont de bons candidats pour adresser notre problématique. L'étude expérimentale que nous avons menée a pour objectif de les comparer et de déterminer celui qui donne les meilleurs résultats pour notre sujet.

3 Approche proposée

Pour notre problème d'inspection, nous développons une approche de classification multi-classes basée sur l'apprentissage profond 3D dont l'apprentissage est basé uniquement sur des données synthétiques, c'est-à-dire des nuages de points 3D synthétiques générés à partir d'un modèle CAO (cf. Fig. 2).

3.1 Nos données industrielles

Nous disposons du modèle CAO de l'assemblage à contrôler et nous pouvons en extraire les modèles CAO des différents éléments constituant cette pièce (supports, éléments connectés au support tels que des vis ou des colliers).
La figure 2 montre plusieurs exemples de supports : le support seul est en rouge et les éléments de contexte [2] proches (vis, colliers) sont en vert. Par la suite, l'ensemble "support + contexte rapproché" sera appelé *support fusion*.
Dans le cadre de l'application industrielle traitée, nous disposons au total de 61 supports différents, tous associés à la même maquette numérique globale.

3.2 Génération de nuages de points 3D synthétiques

Pour l'apprentissage de nos réseaux Deep Learning, à partir de données CAO, nous avons commencé par constituer différentes bases de données synthétiques. Nous avons développé une méthode basée sur le z-buffer, combinée à la sphère de Fibonacci pour l'échantillonnage de l'espace, afin de générer des acquisitions synthétiques à partir de points de vues théoriques. Cette méthode peut être assimilée à un scanner virtuel et prend en compte l'auto-occultation de l'élément, tout comme le fait un scanner réel. Afin de respecter les conditions de sécurité du scanner réel, notre scanner virtuel est positionné à une distance de travail de 60 cm du centroïde du support.

Sphère de visibilité. Dans le but de simuler des points de vue théoriques autour de chaque support, nous définissons une sphère de Fibonacci centrée sur le centroïde du support. La discrétisation de la sphère est uniforme sur sa surface. Elle permet de simuler N points de vue théoriques autour du support, où N est une valeur arbitraire. La sphère a un rayon de 60 cm, correspondant à notre distance de travail imposée par les contraintes industrielles.

Méthode z-buffer. La méthode z-buffer, dite aussi tampon de profondeur, est une méthode incontournable dans le cadre de la visualisation 3D. Celle-ci permet de traiter le problème de visibilité tel que le fait l'oeil humain en déterminant les éléments visibles et ceux étant occultés fournissant un rendu très réaliste de la scène. Cette méthode nous permet ainsi par rendu synthétique d'obtenir une scène de

2. Dans nos travaux, nous appellons "contexte" les éléments qui se trouvent autour d'un support donné. Nous parlerons de contexte rapproché ou de contexte élargi suivant la taille de la zone que nous considérons autour du support.

points 3D ne conservant que les points visibles depuis un point de vue donné.

Data-Augmentation en environnement 3D. Nous avons mis en oeuvre des méthodes de data-augmentation spécifiques à notre problématique, en ajoutant, durant la génération des nuages synthétiques : (1) du bruit gaussien, simulant le bruit d'acquisition du scanner ; (2) des artefacts afin de simuler une occultation partielle, comme celle provoquée par un câble par exemple (cf. Fig. 4). Nous ne conservons que les points 3D inclus dans la bounding box 3D du *support fusion*.

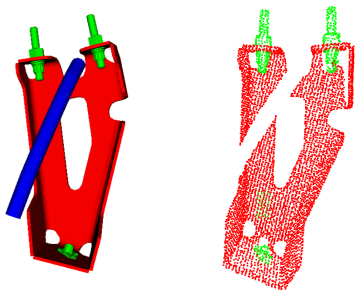


FIGURE 4 – À gauche : un support en rouge avec un câble en bleu ; Á droite : l'occultation provoquée par le câble a conduit à des points manquants dans le nuage de points 3D synthétique généré.

3.3 La phase d'apprentissage

Des expériences préliminaires ont permis d'ajuster au mieux la génération des données synthétiques. Nous avons choisi $N = 40$, après avoir vérifié que même en allant jusqu'a 160 vues différentes par *support fusion* les performances n'étaient pas améliorées.

Constitution de bases de données. Pour chaque *support fusion* (cf. Fig. 2), nous générons 40 nuages synthétiques différents, dont nous utilisons 80% pour l'entraînement (i.e. 32 nuages), 10% pour la validation (i.e. 4 nuages) et 10% pour le test (i.e. 4 nuages). Cela permet d'avoir des nuages différents dans chaque ensemble de données. Pour être plus proche des acquisitions réelles, nous simulons l'occultation ou la présence de bruit sur chaque nuage avec nos méthodes de Data-augmentation (DA). La distribution totale de nos nuages est résumée dans la table 1. Dans le but de conserver un nombre fixe de points par nuage, nous sous-échantillonnons tous nos nuages à 2048 points en utilisant la méthode Farthest Point Sampling (FPS). Nous nous sommes basés sur l'étude [4], qui a comparé les méthodes de sous-échantillonnages FPS et Random pour de la classification de nuages 3D et qui a montré que de meilleurs résultats étaient obtenus avec la méthode FPS.
Le nombre de nuages de Train indiqué dans la table 1 a été obtenu de la façon suivante : 5856 = (1) + (2) + (3)
(1) 61 supports x 40 nuages (sans DA) x 80% = 1952
(2) 61 supports x 40 nuages (DA : artefact) x 80% = 1952
(3) 61 supports x 40 nuages (DA : bruit) x 80% = 1952

	Train	Val	Test
Nuages synthétiques	5856	732	732

TABLE 1 – Distribution du dataset d'entraînement pour les *supports fusion* avec data-augmentation. Ce dataset d'entraînement sera nommé par la suite dataset D1.

Apprentissage de nos réseaux. En entraînant *PCNN*, *PointCNN* et *Shellnet* à l'aide des hyper-paramètres recommandés pour le jeu de données ModelNet40 [11], nous avons obtenu une précision de 98% sur notre jeu de données synthétique de test. Cependant, pour *DGCNN*, nous avons dû ajuster les hyper-paramètres à notre jeu de données pour obtenir de bonnes performances et surtout pour assurer la convergence. Nous avons réduit la taille du lot à 20 et attribué $k = 25$ pour la fonction k-plus-voisins. Notons que pour la phase d'apprentissage nous avons utilisé le supercalculateur OLYMPE (UMS CNRS 3667) [3] avec une puissance de 1.365 Pflops/s.

4 Résultats

Après avoir entraîné nos réseaux sur des données synthétiques, nous les avons évalués sur : (1) des données synthétiques et (2) des données réelles acquises par le scanner 3D dans un environnement industriel. Comme détaillé dans la section 1, grâce à notre module de localisation, nous sommes en mesure de calculer une région d'intérêt (bounding box) 3D et de l'appliquer sur l'acquisition réelle complète, afin de recadrer la partie qui nous intéresse, avant d'appliquer une passe *FPS* pour avoir un nombre fixe de points par nuage (2048 points).
Par la suite, les résultats sont exprimés à l'aide des métriques moyennes suivantes : précision pondérée (weighted-precision), rappel pondéré (weighted-recall) et F1-score pondéré (weighted-F1-score). L'accuracy correspond à l'accuracy totale (aussi appelée Overall Accuracy).

4.1 Quatre réseaux entraînés sur 61 classes synthétiques avec support présent dans le nuage

Nous avons entraîné et testé quatre des cinq réseaux sélectionnés : *Shellnet*, *PointCNN*, *DGCNN*, *PCNN*. Ils ont été entraînés sur les données synthétiques du dataset D1 (cf. table 1). Dans tous ces nuages synthétiques, l'un des 61 supports est présent. L'évaluation sur les 732 nuages synthétiques de test a montré une précision globale de 98% pour les quatre modèles. Ce résultat est cohérent puisque nos données de test synthétiques sont assez similaires à nos données d'entraînement synthétiques.
Nous avons ensuite évalué ces mêmes modèles sur 590 acquisitions de nuages de points 3D réels, divisées en 20 classes différentes. Il convient de noter que les 590 nuages réels contenaient tous le bon support, c'est-à-dire qu'il n'y

3. https ://www.calmip.univ-toulouse.fr/spip.php ?article582

avait aucun nuage avec support manquant ou support erroné. Par la suite, cet ensemble d'acquisitions réelles sera nommé Présence set. Les résultats par classe sont présentés sur la figure 5 et les métriques globales obtenues pour chaque modèle sont exposées dans la table 2.

Nous observons sur la figure 5 que *PointCNN* montre toujours une accuracy supérieure à 80%. Pour les classes comportant de nombreux échantillons (par exemple, les classes 21, 49 et 50), nous parvenons à obtenir d'assez bonnes performances pour la plupart des modèles.

En revanche, *DGCNN* démontre une grande sensibilité à l'occultation par les câbles et certains éléments non présents dans la CAO du *support fusion*. Un exemple est la classe 52 où *DGCNN* démontre une accuracy de 22%. On voit clairement sur la figure 6 que le support est occulté par deux gros câbles, ce qui dénature le nuage de points par rapport à la forme du support, et ce qui induit *DGCNN* en erreur.

ShellNet montre des difficultés à discriminer entre des supports similaires. On peut constater sur la figure 5 que la précision pour la classe 13 est de 0%. Une analyse plus approfondie montre que la classe 13 est systématiquement confondue avec la classe 50 dont l'apparence est très similaire. Ce comportement invalide ce réseau pour notre étude.

Le réseau *PCNN* présente la plus grande instabilité en étant très sensible à l'erreur de localisation, même la plus faible, et également aux occultations. De plus, dès que l'acquisition est réalisée sur une surface légèrement réfléchissante, les zones vides de points déforment le nuage et le support n'est alors pas reconnu par *PCNN*.

Compte tenu des observations précédentes, nous avons décidé de concentrer nos expériences suivantes sur le réseau *PointCNN* qui présente le comportement le plus stable.

Un axe d'amélioration pour la constitution des dataset d'entraînement ou de test a alors été envisagé : durant la génération des données synthétiques, ne pas se restreindre au contexte proche du support (*support fusion*) mais faire intervenir un contexte plus large (câbles, tuyaux, etc.) autour du support considéré (cf. Fig. 7).

La prise en compte d'un contexte élargi devrait nous permettre de réduire la confusion entre les classes et d'être plus robuste aux occultations. Malheureusement, en raison de la forte occultation par le contexte élargi, certaines classes disposent de moins de points de vue pour lesquels le support est visible.

Pour étudier l'intérêt de la prise en compte du contexte élargi par rapport au contexte proche, nous avons mené une deuxième expérience en utilisant ces données (support avec contexte élargi) pour augmenter notre jeu de données de la table 1. La distribution du jeu de données total utilisé pour la deuxième expérience est présentée dans la table 3a. Nous avons comparé les résultats obtenus avec le modèle *PointCNN* entraîné sur le dataset synthétique D1 et le modèle *PointCNN* entraîné sur le dataset synthétique D2. Les résultats des tests sur le Présence set (nuages réels) sont présentés dans la table 3b. Cette expérience montre que la prise en compte du contexte élargi n'améliore pas les performances.

4.2 *PointCNN* entraîné sur 61 classes synthétiques avec support présent et 61 classes synthétiques avec support absent

Jusqu'à présent, nous avons entraîné nos modèles avec des nuages synthétiques dans lesquels il y avait un support et nous les avons testés sur des nuages synthétiques et réels dans lesquels il y avait un support présent. Afin de simuler l'absence de support, nous avons généré des nuages de contexte uniquement (i.e. sans support), pour chaque support (cf. Fig. 8b). Pour le processus de génération, nous avons 122 classes : 61 classes avec support présent et 61 classes sans support (contexte seul). Nous conservons pour les situations de support absent les mêmes points de vue que pour les nuages avec support présent, c'est-à-dire les points de vue à partir desquels le support est visible, quand il est présent. Un tel ensemble de données synthétiques a une distribution comme indiqué dans la table 4.

Comme dans l'expérience précédente, nous entraînons le réseau *PointCNN* avec ce jeu de données élargi. Le modèle entraîné (sur des données synthétiques) est testé sur le jeu de test des données synthétiques ; les résultats sont présentés dans la table 4. On peut remarquer que le modèle atteint un niveau de précision de 88% sur les données synthétiques avec contexte. Ceci nous montre que le contexte seul n'est pas assez discriminant pour la classification. Ce résultat n'est pas surprenant étant donné que cette tâche est plus complexe que la précédente.

Nous disposons de 34 acquisitions réelles représentant la situation où le support est manquant, réparties en 14 classes différentes. Par la suite, cet ensemble sera nommé Absence set. Nous évaluons notre modèle sur notre ensemble total d'acquisitions réelles ; les résultats sont présentés dans la table 5.

Les résultats obtenus sur les données réelles ne semblent pas très bons et nous avons essayé de comprendre pourquoi.

Sur la figure 9, la barre orange montre une très forte proportion de fausses alarmes [4] (faux positifs) sur le Présence set, précisément sur 230 nuages (soit 39% du Présence set). Cette forte proportion de fausses alarmes explique les faibles taux de accuracy et rappel rapportés dans la table 5, mais aussi la forte valeur de précision de cet ensemble. Cette précision très élevée reflète que les situations de présence de support sont rarement confondues entre elles. Ces résultats suggèrent que le contexte est une source de confusion pour notre modèle.

L'accuracy par classe sur l'ensemble Absence set est présentée sur la figure 10. Ces résultats confirment le taux d'erreur élevé (45%) déjà présenté dans la deuxième ligne

4. Les fausses alarmes contenues dans la barre orange correspondent aux situations où le support a été détecté absent alors qu'il est présent. Dans la barre noire, figurent aussi des fausses alarmes, mais celles-ci correspondent aux situations où le réseau a confondu deux classes entre elles.

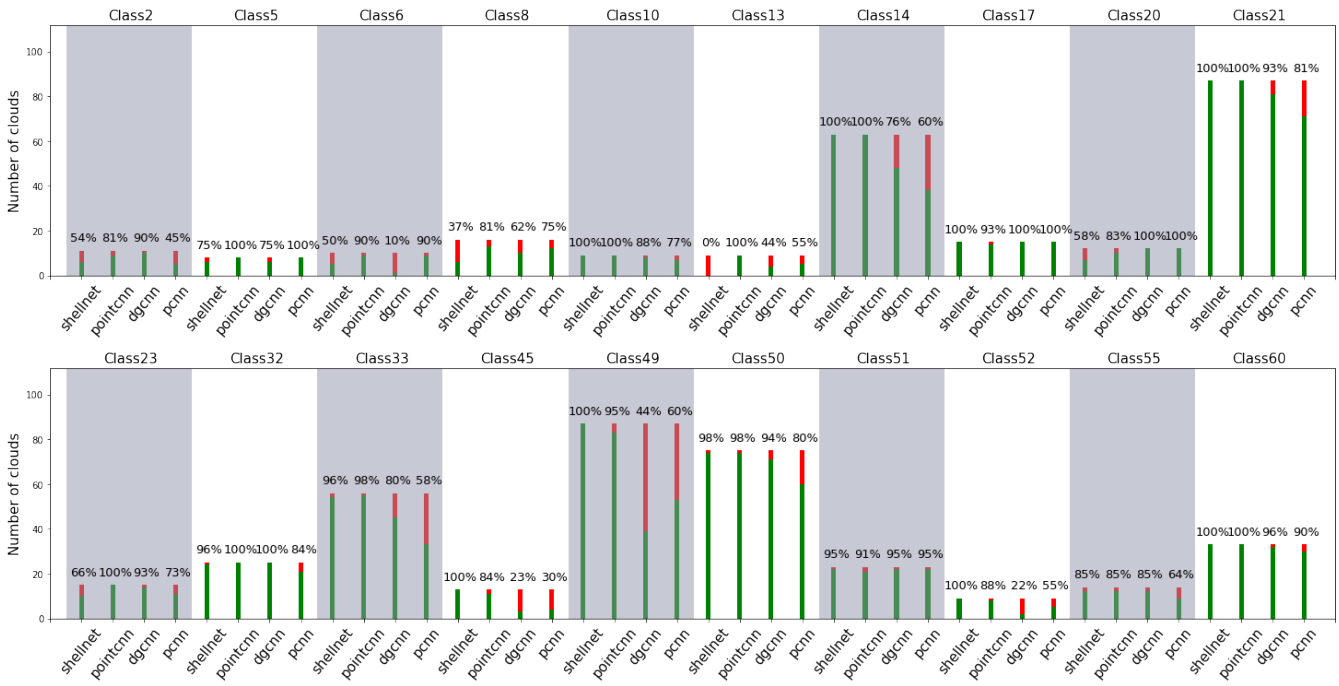


FIGURE 5 – Evaluation sur les 20 classes du dataset Présence set (nuages réels). En vert : le pourcentage d'acquisitions correctement classées par un modèle. En rouge : le pourcentage d'acquisitions mal classées par un modèle. Sur l'axe Y : le nombre total d'acquisitions testées. Sur l'axe X : les quatre modèles de classification testés. Des bandes verticales blanches et grises sont ajoutées afin de séparer les classes pour des raisons de meilleure lisibilité.

	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 – Résultats pour chaque modèle sur la moyenne des 20 classes d'acquisitions réelles. Les métriques utilisées sont weighted-precision (Prec.), weighted-recall (Rappel) and weighted F1-score (F1-scr).

	Train	Val	Test
Dataset synth. D1	5856	732	732
Dataset synth. avec contexte élargi	1408	187	187
Total Dataset synth.	7264	919	919

(a) Distribution du dataset synthétique D2

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

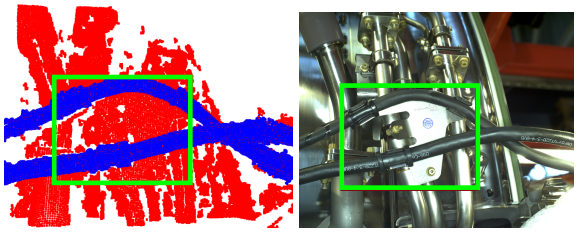
(b) Résultats sur le Présence set (nuages réels)

TABLE 3 – (a) Distribution du dataset synthetique D2, définissant ainsi un nouveau jeu de données d'apprentissage (b) Résultats des modèles "*PointCNN* (1)", entraîné sur le dataset D1, et "*PointCNN* (2)", entraîné sur le dataset D2

	Set d'apprentissage		**Résultats sur le set "Test"**				
Jeu de nuages synthétiques	Train	Val	Test	Acc.	Prec.	Rap.	F1-sc.
Dataset synth. D1	5856	732	732	99%	99%	99%	99%
Dataset synth. avec contexte élargi	1408	187	187	88%	97%	88%	91%
Dataset synth. contexte seul	1404	184	183	88%	98%	87%	91%

TABLE 4 – Distribution du dataset synthétique d'entraînement et résultats sur le set "Test"

de la table 5. En examinant de près les données, nous avons observé que sur les 45%, il n'y a que 10% de faux négatifs (c'est-à-dire 4 échantillons du Absence set qui sont détectés comme situation de présence alors que le support est absent), tandis que la confusion entre les différentes classes de contexte représente les 35% restants du Absence set.



(a) (b)

FIGURE 6 – Le support de la classe 52 est occulté par des câbles. Le rectangle vert montre le support dans la scène.

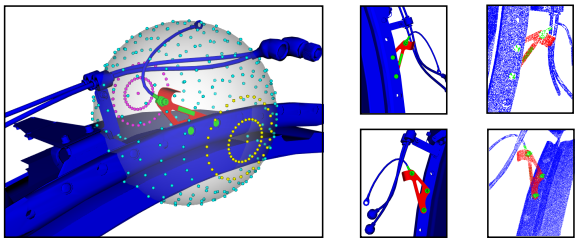
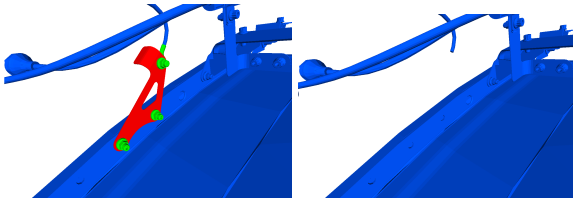


FIGURE 7 – À gauche : une visualisation de la sphère de Fibonnacci centrée sur le *support fusion*; au milieu : exemples d'images synthétiques générées à partir du modèle CAO; à droite : exemples de nuages de points synthétiques fournis par notre scanner virtuel. Les parties bleues correspondent au contexte élargi.



(a) (b)

FIGURE 8 – Visualisation de la même scène synthétique à partir du même point de vue. L'image de gauche décrit un cas où le support est présent et celle de droite un cas où le support est absent.

	Acc.	Prec.	Rap.	F1-sc.	Nb. acq
Présence set	49%	92%	49%	55%	590
Absence set	55%	56%	55%	54%	34

TABLE 5 – Résultats sur les acquisitions réelles. Nous avons 20 classes pour le dataset Présence set et 14 classes pour le dataset Absence set.

Á ce stade, il convient de rappeler que le besoin industriel est d'être capable de détecter l'absence d'un support. On ne demande pas de reconnaitre à quelle classe appartient le support manquant.

Afin de répondre à cet objectif, nous avons décidé de considérer toutes les classes d'absence comme une seule classe. Dans ce nouveau mode, nous avons juste besoin de savoir si le modèle reconnaît la présence d'un support ou non. Si ce n'est pas le cas, notre solution répondra "support absent". À noter que le modèle n'est pas réentraîné; nous considérons simplement les sorties du modèle différemment. La figure 11 présente les résultats.

Ces résultats sont bien meilleurs, et cette façon d'exploiter les résultats porte l'accuracy totale à 88% sur le Absence set, montrant une bonne capacité du modèle à détecter l'absence d'un support.

5 Conclusion

Dans cet article, nous avons montré qu'un apprentissage basé uniquement sur des nuages de points 3D synthétiques (générés à partir d'un modèle CAO 3D) pouvait donner de bons résultats en classification sur des nuages de points réels acquis par un scanner 3D. Nous avons d'abord sélectionné 4 réseaux de Deep Learning 3D et nous avons démontré que le réseau *PointCNN* était le plus robuste aux occultations et aux similarités, le rendant très prometteur pour notre processus d'inspection de pièces aéronautiques complexes. Dans un deuxième temps, nous avons voulu analyser l'influence des éléments situés autour de nos supports (le contexte élargi) en les incluant dans nos données d'entraînement. Nous avons alors constaté que ce contexte élargi n'apportait aucune information à la classification de nos données réelles. Nous avons ensuite cherché à savoir si le contexte élargi pouvait contribuer à la détection d'absence de support. Nous en avons conclu qu'il induisait le réseau en erreur, provoquant de nombreuses fausses alarmes et dégradant ainsi les performances. Cette expérience a également révélé que *PointCNN* confondait les situations d'absence entre elles, mettant en évidence que le contexte élargi n'était pas discriminant en l'absence de support. Toutefois, le modèle parvient à détecter l'absence de support dans un nuage de points, ce qui est prometteur car l'objectif industriel est de pouvoir détecter l'absence d'un support, et la reconnaissance de la classe associée à un éventuel support manquant n'est pas requise.

Sur la base des résultats précédents, nous envisageons de séparer la détection de la présence/absence d'un support de la détection de la situation correspondant à un support mal monté. Un premier modèle déciderait du dilemme absence/présence, puis un second modèle classerait uniquement les nuages de présence pour vérifier si le bon support a été monté ou s'il a été remplacé par un autre.

6 Remerciements

Ce travail de recherche a été réalisé dans le cadre d'une thèse de doctorat cofinancée par la Région Occitanie. Nous tenons à remercier la Région Occitanie pour son soutien financier.

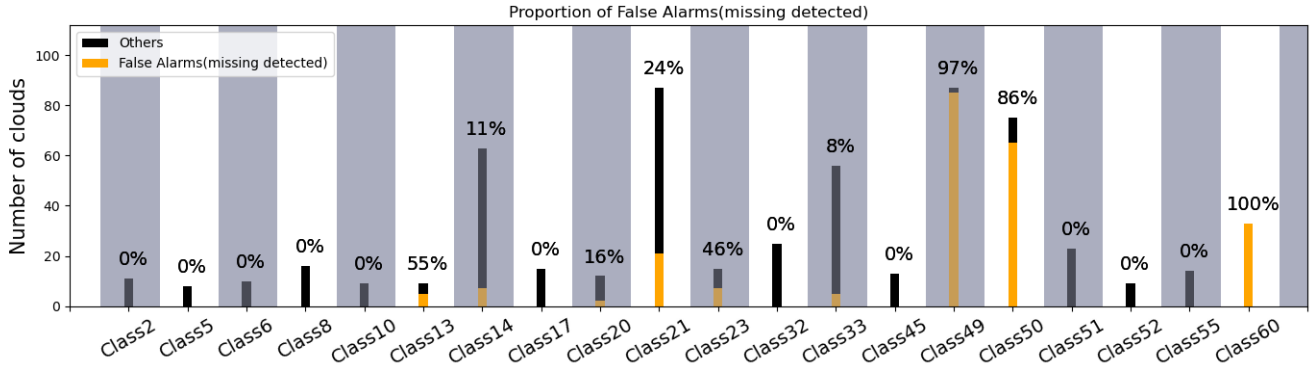


FIGURE 9 – Evaluation sur les données réelles des 20 classes du Présence set, représentant les fausses alarmes pour chaque classe. En orange : proportion de fausses alarmes par classe. En noir : le complément à 100%. L'axe Y : le nombre total de nuages testés. L'axe X : classes réelles testées. Les bandes grises et blanches sont ajoutées pour séparer les classes et assurer une meilleure lisibilité.

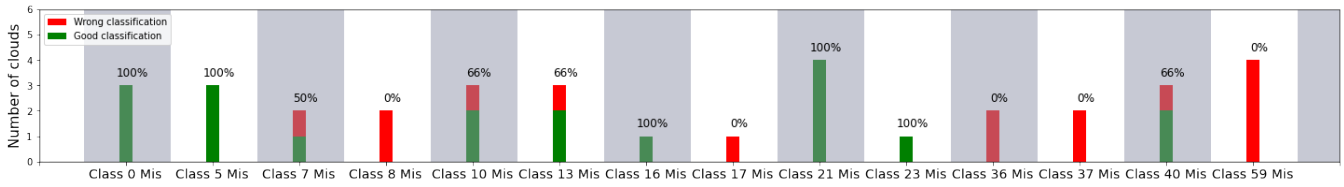


FIGURE 10 – Evaluation sur les 14 classes du dataset Absence set. En vert : le pourcentage de nuages correctement classés. En rouge : le pourcentage de nuages mal classés. Sur l'axe Y : le nombre total de nuages testés. Sur l'axe X : classes réelles testées.

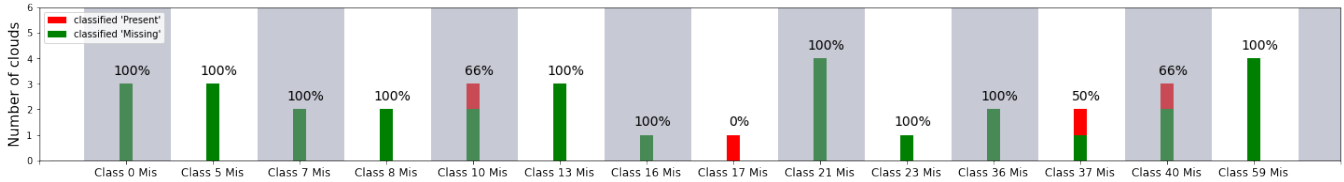


FIGURE 11 – Evaluation sur les 14 classes du dataset Absence set. En vert : pourcentage de nuages correctement détectés comme "support absent" par classe. En rouge : le pourcentage de nuages mal classés. Sur l'axe Y : le nombre total de nuages testés. Sur l'axe X : les classes réelles testées.

Références

[1] Heinz Wörn, Thomas Längle, and Michael Gauß. ARIKT : Adaptive robot based visual inspection. *KI*, 17 :33–, 01 2003.

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

[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] I. Mikhailov, I. Jovancevic, 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.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.