Automatic inspection of aeronautical mechanical assemblies using 2D and 3D computer vision

Hamdi Ben Abdallah, Igor Jovančević, Jean-José Orteu, Benoit Dolives, Ludovic Brèthes

Automatic inspection of aeronautical mechanical assemblies using 2D and 3D computer vision

Hamdi Ben Abdallah[1,2], Igor Jovančević[2*], Jean-José Orteu[1], Benoît Dolives[2], Ludovic Brèthes[2]

[1] Institut Clément Ader (ICA) ; Université de Toulouse ; CNRS, IMT Mines Albi, INSA, UPS, ISAE ; Campus Jarlard, 81013 Albi, France

[2] DIOTASOFT, 201 Rue Pierre et Marie Curie, 31670 Labège, France

*corresponding author, E-mail: ijo@diota.com

Abstract

Quality control is of key importance in the aerospace industry. This paper deals with the automatic inspection of mechanical aeronautical assemblies. For that purpose, we have developed a computer-vision-based system made of a robot equipped with two 2D cameras and a 3D scanner. The 3D CAD model of the mechanical assembly is available. It is used as a reference and it describes the assembly as it should be. The objective is to verify that the mechanical assembly conforms with the CAD model. Several types of inspection are required. For instance, we must check that the needed elements of the assembly are present, and that they have been mounted in the correct position. For this kind of inspection we use the 2D cameras and we have developed inspection solutions based on 2D image analysis. We have found that some types of inspection cannot be performed by using only 2D image analysis. A typical example of such types is detecting the interference between elements. It requires to check if two flexible elements (e.g. cables, harnesses) or a flexible and a rigid element (e.g. pipe, support) are at a safe distance from each other. For this type of situations, we use the 3D data provided by the 3D scanner and we have developed an inspection solution based on 3D point cloud analysis. We have also developed a method to compute the best viewpoints for the sensor held by the robot, in order to obtain an optimal view of each component to be inspected. The view-point selection is performed off-line (before the on-line inspection) and it exploits the CAD model of the mechanical assembly.

The proposed automatic computer-vision-based inspection system has been validated in a context of industrial applications. Our software solution for 2D image analysis has been deployed on the robot platform as well as in a hand-held tablet. Since it requires a 3D sensor, our approach based on 3D point cloud has been tested in the robotic context only.

1. Introduction

To automate the inspection process, improve its traceability and repeatability, and to reduce the human error, many aerospace companies aim to automate numerous and complicated operations of quality control of aircraft's mechanical assemblies to address the various and growing security requirements.

We address the problem of automatic inspection in two parts: first, automatic selection of informative viewpoints before the inspection process is started (offline preparation of the inspection, section 2), and second, automatic processing of the 2D images or 3D point clouds acquired from said viewpoints (online inspection process, section 3 and section 4).

Several types of verification can be carried out on a mechanical assembly. For instance, check that the elements of the assembly are present and that they have been mounted in the correct position (the CAD model is the reference), check if two flexible elements (e.g. cables, harnesses) or a flexible and a rigid element (e.g. pipe, support) are at a safe distance from each other, and check that the bend radius of cable complies with safety standards.

According to the type of verification to be performed, one of the two following general strategies has been developed:

1. Model-based 2D image analysis (also called 2D inspection). This method is easy to deploy since it only uses the two RGB cameras (camera with a wide field of view and camera with a narrow field of view) mounted on the robot (see Fig.1) and the CAD model of the object to be inspected. This method is presented in our recent paper [1].

2. Model-based 3D point cloud analysis (also called 3D inspection). This method uses one RGB camera (wide field of view camera) and the 3D scanner mounted on the robot, and the CAD model.

First strategy is preferred whenever possible, for time reasons (3D point cloud acquisition and analysis are more time consuming than those in the case of 2D image).

Rest of the paper is organised as follows. In section 2 we present our scoring function for evaluating possible viewpoints for our inspection tasks. In section 3 our approach based on 2D image analysis has been detailed and in section 4 we explain our methodology based on 3D point cloud analysis. Finally the paper is concluded in section 5. State of the art in the field is summarized in our recent papers [1], [2] and [3] so will not be elaborated in the present paper.

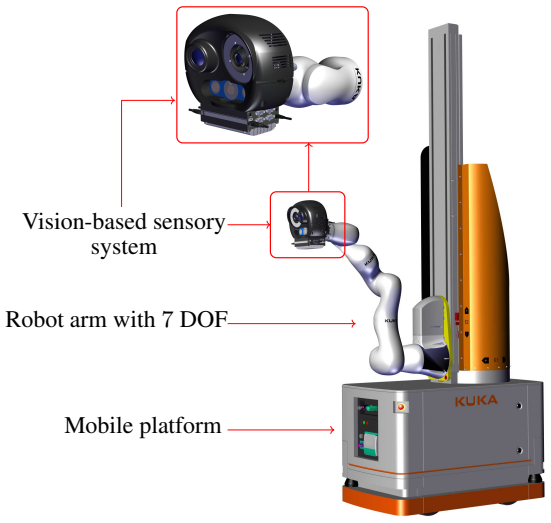


Figure 1: Our inspection system: robot with its vision-based sensory system

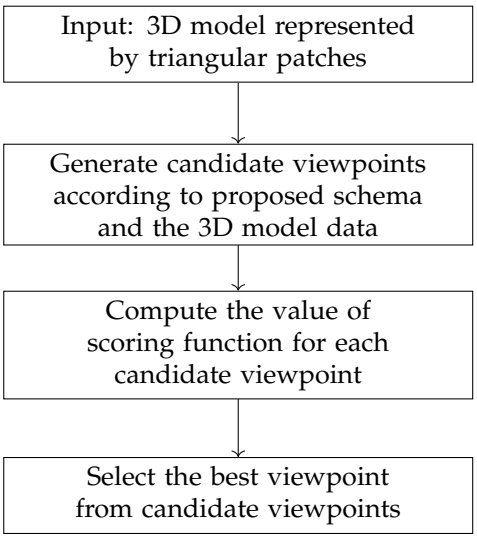


Figure 2: The Overview of viewpoint Selection Scheme

2. Viewpoints selection: offline process

The initial setup of camera viewpoints cannot be done manually, because a human operator cannot define with sufficient accuracy the camera position that will allow him to get the best viewpoint of the element to be inspected. Therefore, we need a (semi-)automatic offline configuration process that is used to compute the best viewpoints which can help to improve the quality and the efficiency of inspection. Later, during the online inspection process, the camera will be placed at those calculated viewpoints.

The strategy proposed to find the best viewpoint (computing the 6D location – position and orientation – of the camera with respect to the scene to be observed), based on the 3D model of the assembly is illustrated in Fig. 2. For each of two types of inspection, a scoring function is constructed. These functions are designed in a way to maximize observability of the element to be inspected. Finally, the best viewpoint can be selected from all of candidate viewpoints according to the scoring function.

Candidate viewpoints are generated by sampling a fixed radius virtual sphere around the element to be inspected. This radius is chosen in such a manner to comply with industrial security standards for ensuring security of the inspected element as well as of the inspecting robot.

The candidate viewpoints are evenly distributed on the sphere surface according the range of longitude θ and colatitude φ (Fig. 3). The best viewpoint is selected according to the value of the scoring function.

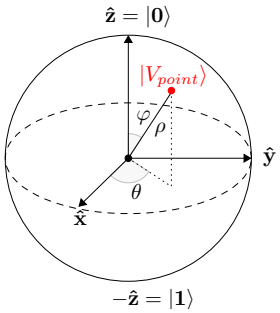


Figure 3: A viewpoint on the visibility sphere

2.1. Scoring function for 2D inspection

Our scoring function for 2D inspection combines three criteria:

- Visibility of the information ($f_{visibility}$)
- Ambiguity of observed shape due to parasite edges ($f_{parasite}$)
- Similarity of observed shape to expected shape (f_{shape})

$f_{visibility}$ - each viewpoint yields a render of our element of interest with a certain number of pixels. The $f_{visibility}$ function takes into account this value and favorizes those viewpoints with higher number of pixels (area in the image). It should be noted that the occlusion is taken into account in the rendering pipeline (see Fig. 6 for ex.).

$f_{parasite}$ - this function favorizes the viewpoints with less rejected edgelets due to parasite edges in their vicinity. For more details about parasitic edges see section 3.4.1.

f_{shape} - finally, each candidate viewpoint is also evaluated by computing the similarity between the shape of filtered projected edgelets of the inspection element (obtained

by rejecting those prone to parasitic edges and occlusion) and the unfiltered projected edgelets.

The three criteria presented before for evaluating each viewpoint are combined using a scoring function f_{score}:

$$f_{score} = w_v \times f_{visibility} + w_p \times f_{parasite} + w_s \times f_{shape}$$

$$w_v + w_p + w_s = 1$$

w_i's are weights that assign importances to the different viewpoint evaluation criteria, and they can be adjusted by the user depending on the importance of the criteria for a specific inspection application. $f_{visibility}$, $f_{parasite}$ and f_{shape} are normalized.

2.2. Scoring function for 3D inspection

For 3D inspection, the scoring function is based on the visibility criteria only. This criteria is explained in the previous section and represented by a function $f_{visibility}$. Hence in the case of 3D inspection the following holds:

$$f_{score} = f_{visibility}$$

3. Inspection by 2D image analysis

In this section we will detail our main contribution when relying on 2D images and CAD model of the assembly. The goal of this inspection task is to verify that a rigid element of the assembly is present and well mounted. Examples of the CAD models of the assemblies we have been inspecting can be seen in Fig. 4.

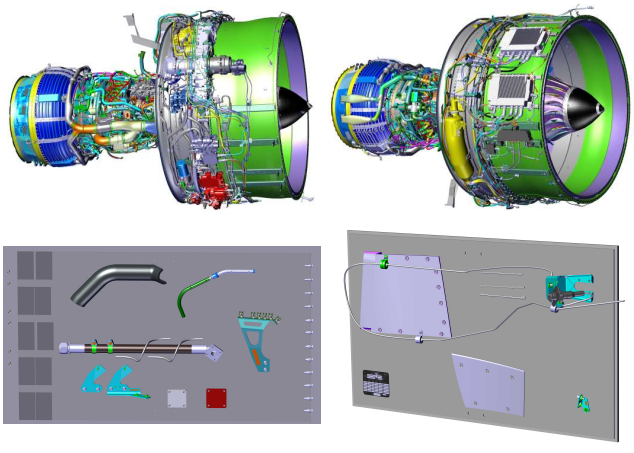


Figure 4: Some of our CAD models: (1^{st} column) aircraft engine, (2^{nd} column) two testing plates with several elements

Our inspection process for the defect detection illustrated in Fig. 5 consists of four main steps. First, from the CAD model of the observed object, 3D contour points (edgelets) are extracted. Further, the edgelets with their direction vectors are projected onto the real image using a pinhole camera model (extrinsic and intrinsic parameters). Used image is acquired by the camera with narrow field of view. It should be noted that the pose of the effector is known thanks to the calibration process but also thanks to an in-house developed localisation module which performs pose refinement. This module is processing images acquired by the wide field of view camera. Finally, we perform the matching between the set of projected edgelets and the set of real edges extracted in the image.

We explain each building block of our strategy in the following sections.

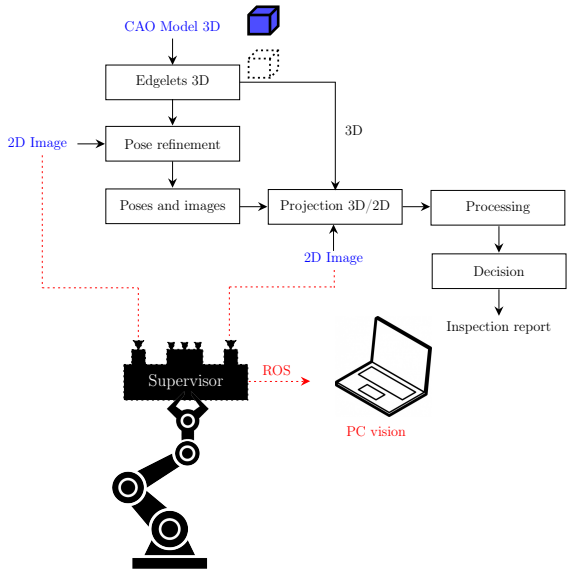


Figure 5: Overview of our online robot-based inspection pipeline

3.1. Step 1 (edgelet generation)

An edgelet E_i is defined as a 3D point $p_i = (x_i, y_i, z_i)$ and a normalized direction vector $\vec{d}_i$ of the edgelet. These points p_i represent the 3D contour points of an object (see Fig. 6). The edgelets are extracted offline by rendering a CAD model as explained in [4].

The output of this step is a set of edgelets that are evenly distributed.

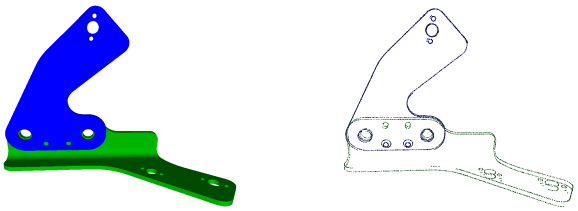


Figure 6: Edgelets extraction: (left) example of CAD part, (right) edgelets extracted from this CAD part

3.2. Step 2 (camera pose estimation)

Our strong pre-assumption is that the moving camera is localized with respect to the complex mechanical assembly being inspected. Namely, we rely on an in-house developed system for initial pose estimation based on 2D-3D alignment, our inspection camera being then considered as moving in a partially known environment. The mentioned algorithm accurately estimates the camera's pose with respect to the object by including the geometric constraints of its CAD model in an optimization process.

3.3. Step 3 (3D/2D projection)

In this phase, we project 3D points of edgelets onto the image using known extrinsic and intrinsic camera parameters (see Fig. 7). The input of this step are an image, the camera pose with respect to the assembly (CAD) and a set of evenly distributed edgelets of the element to be inspected.

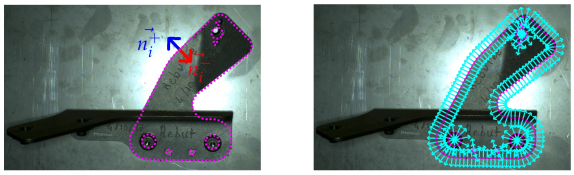


Figure 7: (Left) projection of edgelets (3D points p_i), (right) inward-pointing normals n_i^+ and outward-pointing normals n_i^-, generation of search line l_i

3.4. Step 4 (matching projected edgelets with real edges)

The goal of this step is the matching between the set of projected edgelets and the set of real edges according to the direction of the normal vectors calculated in step 3 (section 3.3). If at least one real edge is found on the search line, the edgelet is considered matched. Otherwise, it is considered not matched.

Initially we employed the well known and widely used Canny edge detector for extracting meaningful real edges in the real image. The results reported in our recent paper [1] are obtained by this algorithm.

Nevertheless, we have identified more recent edge detection approaches such as [5] based on neural networks. Fig. 8 illustrates a quantitative comparison of Canny edge detector and the edge detector proposed in [5]. In our analysis conducted by this moment, we have noticed that the latter one outperforms Canny, notably in treating highly reflective as well as highly textured areas. Moreover, by using machine learning based detectors we avoid the sensitive phase of parameters tuning unavoidable in the case of Canny approach. In the present paper, the inspection results have been obtained based on 2D edges extracted by the machine learning technique.

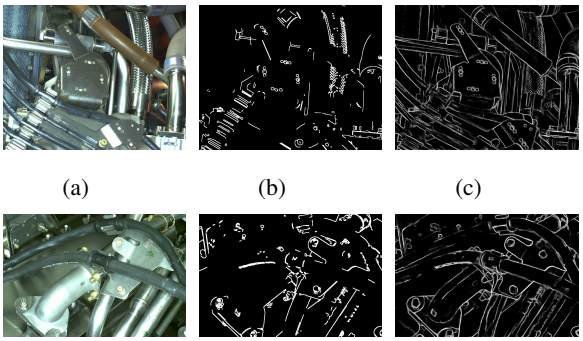


Figure 8: Quantitative analysis of Canny edge detector and edge detector based on neural network (a,d) input images, (b,e) results by Canny edge detector, (c,f) result by neural network edge detector

3.4.1. *Parasitic edges handling*

In this section, we describe our principal contribution, introduced in order to remove as much as possible irrelevant edges. Namely, by using CAD render, we anticipate the existence of some portion of parasitic edges coming from other elements mounted in the vicinity of the inspected element. Indeed, often there are edges very close to the external boundary of the element to be inspected. We call them *parasitic edges* or unwanted (irrelevant) edges. To solve this kind of problem, we introduce a new notion called *context image* of an element to be inspected. From the CAD data and the particular camera position, we form a synthetic view (context image), which has exactly the same perspective as the real image, only that the inspected element is *excluded* from the assembly (see Fig. 9).

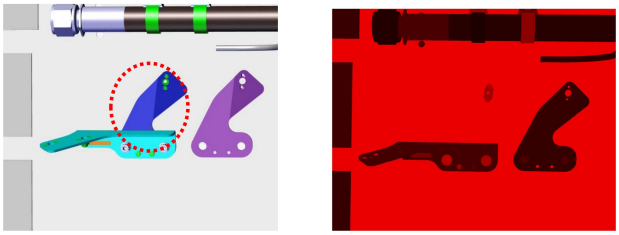
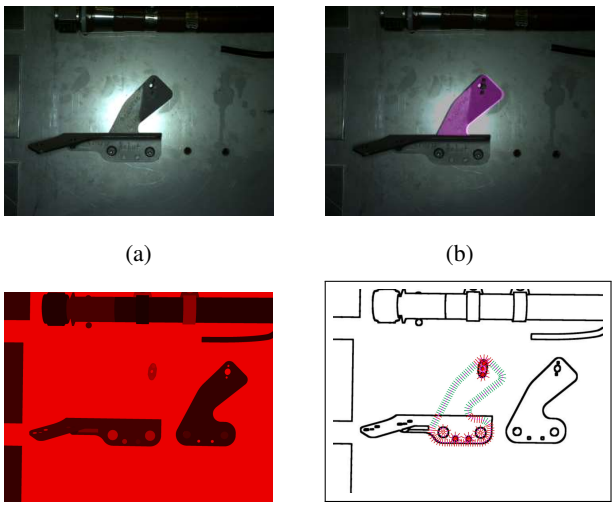


Figure 9: (Left) example of CAD model - inspected element (dark blue) is in the red circle, (right) context image of the inspected element

The process of decreasing number of parasitic edges is illustrated in Fig. 10. First, we project the edgelets of the element of interest onto the context image, as explained in step 3 (section 3.3).

Further, we form search lines (see Fig. 10d) and look for the intensity change in this virtual context image. Intensity changes are shown in Fig. 10d. If such a change in context image is found, it is very probable that this edge will be present in the real image as well. Therefore, we will

not consider this edgelet for matching. These edgelets are shown in red in Fig. 10d. Other edgelets, shown in green in Fig. 10d, are considered in the matching phase.



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

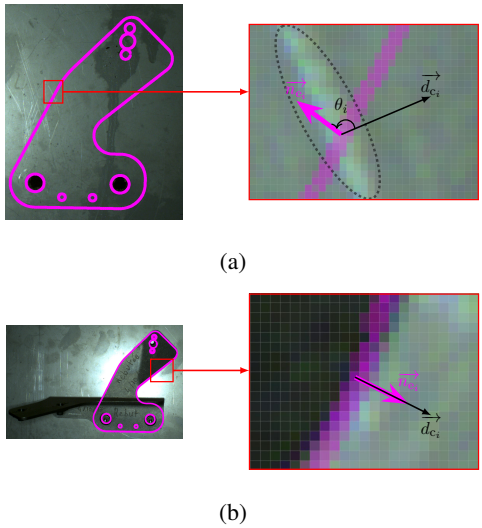
Figure 10: Illustration of the parasitic edges handling: (a) input image, (b) projection of the element to be inspected, (c) context image of the element to be inspected, (d) gradient changes in the context image and considered edgelets (green) and rejected edgelets (red)

3.4.2. *Edges weighting*

For each edgelet in the 3D CAD that is projected onto the image plane, we are interested in finding an image location that contains a real change in image intensity (edge). For this, we first produce an edge image and search for candidate locations along a search line l_i that is obtained as described in section 3.3. Each edge location along the search line that is within a pre-specified distance from the location of the projected edgelet is weighted according to a Gaussian function of the distance of the edge point from the location of the projected edgelet. Instead of simply counting matched edgelets, these weights will be added up in the final score. We do this in order to favorize the edgelets matched with very close real edges and penalize those which are matched with real edges that are far away.

3.4.3. *Gradient orientations*

When we search for an edge in the image, we may encounter edges due to image noise or geometric features that arise from the parts different from the inspection element. To reject such candidates, at each encountered edge location, we compute the gradient of the edge and find the angle it makes with the search line θ_i (see Fig. 11). If this angle is greater than a threshold ($\theta_i > \theta_{th}$) , we reject the candidate and continue our search. If no edges are found within the search limits that satisfy the orientation criteria, we decide that no edge corresponding to the inspection element of interest is present. In our experiments, we have set $\theta_{th} = 20°$.



(a) (b)

Figure 11: Criteria of orientation similarity. (1^{st} column) projection of edgelets corresponding to the inspecting element onto the 2D image, (2^{nd} column) a zone of the image zoomed in 3200 times. (a) Contour point c_i and edgelet e_i have too different orientations ($\theta_i = 120°$), (b) Contour c_i and edgelet e_i have similar orientations ($\theta_i = 0°$)

3.5. Step 5 (making decision on state of the element)

Producing an inspection decision based on matching filtered edgelets independently with edges in the real image can lead to false positives since the matched edges in the real image may as well arise from an element with not the same shape as the inspected one. To take the global shape of the element into account, we propose to characterize the filtered edgelets and the edges detected in the real image using the Shape Context framework [7]. The Shape Context (represented by a set of points sampled from the edges of the object) is a shape descriptor that captures the relative positions of points on the shape contours. This gives a globally discriminative characterization of the shape and not just a localized descriptor. These are then used to measure similarities between shape context representations of the two sets of edge locations.

The workflow of the decision-making step is shown in Fig. 12. The decision is taken using two indicators: the similarity score ($\text{similarity}_{\text{score}}$) based on Shape Context and the matched ratio ($\text{matched}_{\text{ratio}}$) represented by the ratio between matched and not matched edgelets.

In Fig. 13, we show the CAD model, the RGB image acquired by the inspection camera, and the final result on two different cases (element OK and element NOK).

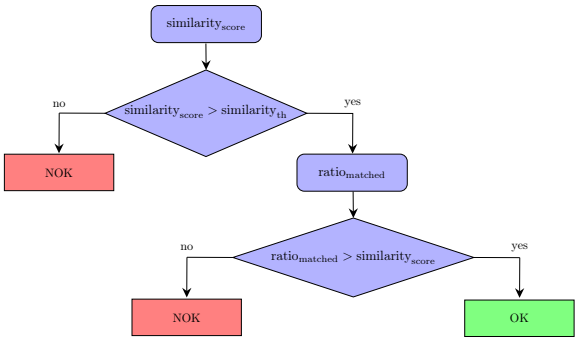


Figure 12: Decision making

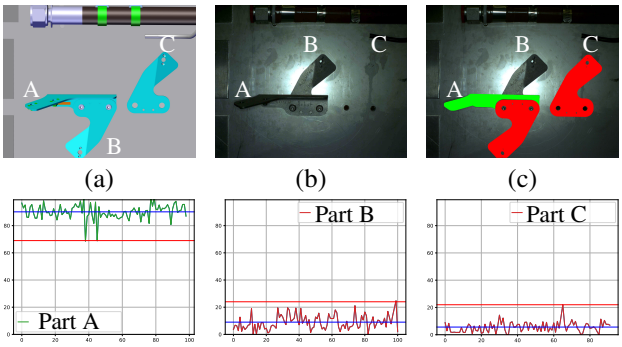


Figure 13: (1^{st} row) (a) CAD model, (b) 2D inspection image, and (c) result of inspection with green OK (element present) and red NOK (element absent or incorrectly mounted), (2^{nd} row) respectively the matched$_{\text{ratio}}$ (curve) and threshold matched$_{\text{th}}$ (red horizontal line) computed on the three different cases 100 times: element A (present) in green, element B (incorrectly mounted) in red, and element C (absent) in red

3.6. Metrics used for evaluation

3.6.1. Accuracy, sensitivity, specificity

Table 1 outlines definitions of the true positive (TP), the false positive (FP), the true negative (TN) and the false negative (FN) in defect detection.

Table 1: Definitions of TP, FP, TN, FN in defect detection

	Actually defective	Actually non-defective
Detected as defective	TP	FP
Detected as non-defective	FN	TN

Based on these notations, the detection success rate (known also as detection accuracy) is defined as the ratio of correct assessments and number of all assessments. Sensitivity is defined as the correct detection of defective samples. Similarly, the specificity can be defined as the correct detection of defect-free samples. Finally, the overall accuracy could be calculated based on the metrics of sensitivity and specificity. The accuracy, sensitivity and specificity are defined as:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}}$$

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

3.6.2. Precision and Recall

Precision and recall metrics are used to validate the performance of defect detection. Precision and Recall are defined as:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F_{\text{score}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The F_{score} is calculated to measure the overall performance of a defect detection method. The highest value of the F_{score} is 100% while the lowest value is 0%.

3.7. Evaluation

The approach was tested on a dataset acquired in a factory with a total number of 43 inspected elements and 643 images. Elements are inspected one by one and decision on each of them is based on one acquired image. The distance between camera and inspected element has been set at 600 mm with a tolerance of $\pm 6mm$. Some examples of our dataset used to evaluate our approach are shown in Fig. 14. It can be noted that our dataset is acquired in a context of industrial robotized inspection in conditions of very cluttered environment.

The overall performance of our method is represented by $F_{\text{score}} = 91.61\%$. The values of specificity, accuracy, sensitivity, precision and recall of our method are shown in Tables 2 and 3.

Table 2: Result of the evaluation (TP, TN, FP, and FN)

Nbr of elements	Nbr of images	TP	TN	FP	FN
43	643	344	236	63	0

The experimental results show that our proposed approach is promising for industrial deployment.

The main challenge we faced lies in the fact that our dataset is acquired in a factory, on a real and very complex assembly at the very end of the production line (almost all the elements are mounted). Therefore, the environment

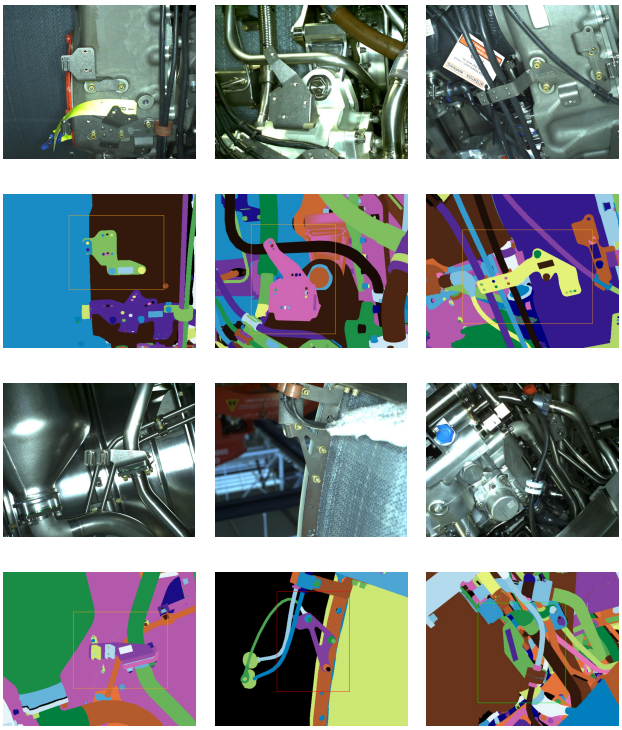


Figure 14: Some examples of our dataset used to evaluate our approach in a context of robotized inspection in conditions of very cluttered environment: (1^{st} and 3^{rd} rows) real images, (2^{nd} and 4^{th} rows) corresponding renders of CAD models

Table 3: Performance measures

Accuracy	Sensitivity	Specificity	Precision	F_{score}
90.20%	100%	78.93%	84.52%	91.6%

around inspected element is very cluttered and the matching task becomes more difficult.

Please refer to our recent paper [1] for more detailed explanation of the experiments and especially the comparison of the evaluation of the hand-held tablet inspection system and the robotic inspection platform.

4. Inspection by 3D point cloud analysis

The strategy based on 3D point cloud data is used when 2D image processing is not sufficient. Indeed, sometimes, using 3D data is necessary, for example for checking if the distance between two cables is conform to the security standards. These types of defects are undetectable with an RGB camera because the cables have the same color. Moreover, obtaining distance measurements is challenging in the absence of depth information.

More precisely, it is required to check if two flexible elements (e.g. cables, harnesses) or a flexible and a rigid element (e.g. pipe, support) are at a safe distance from each other. Fig. 15 illustrates that 2D image analysis is not sufficient to solve this type of inspection problem.

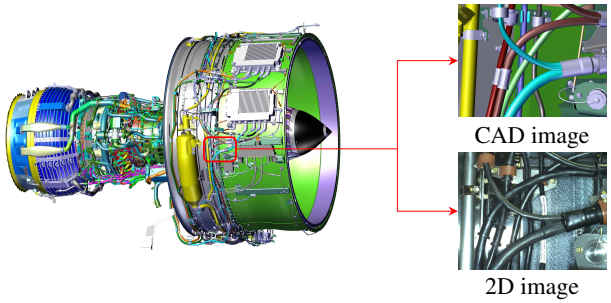


Figure 15: Example of 3D inspection: (left) airplane engine (right) the interference between some cables cannot be determined by 2D image analysis

The online inspection process consists of several steps, which are detailed in this section: (1) a 3D point cloud is acquired using the pre-calibrated Ensenso N35 3D scanner mounted on the robot arm (Fig. 16), (2) a two-step global-to-local registration procedure that allows for alignment of the 3D data with the 3D CAD model is performed, (3) the cables are segmented from the 3D point cloud, (4) the distance between each cable and its surrounding environment is computed in order to detect a possible interference problem, (5) the bending radius of each cable is computed.

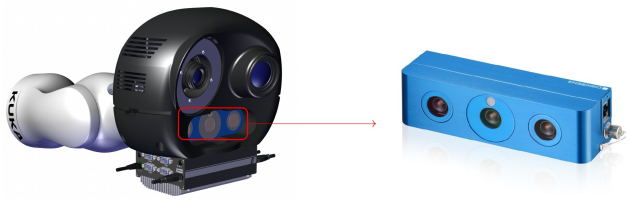


Figure 16: Pre-calibrated Ensenso N35 3D scanner

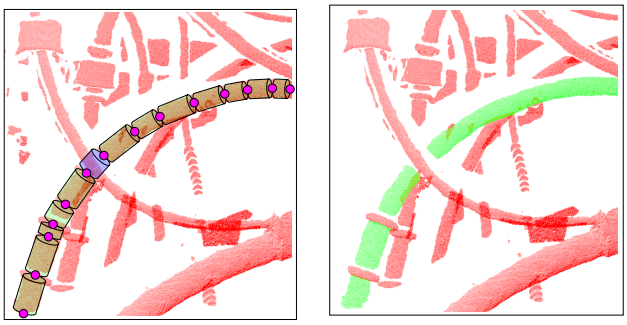
4.1. 3D cable segmentation

Cables are designed to be flexible. The registration or matching with 3D CAD model becomes problematic when the target objects are non-rigid, or prone to change in shape or appearance. Nevertheless, cable connection points, where the cable meets connectors or clamp, are rigid. These connection points are the most important elements for aligning with CAD model. Indeed, these points make it possible to guarantee a rigidity of a portion of cable.

After some preprocessing steps such as known noise removal techniques as well as subsampling CAD model mesh, we proceed to a global 3D/2D alignment phase. The results of the global registration are refined by a local registration process that uses an iterative non-linear ICP proposed by Besl and McKay [8] and Levenberg-Marquardt algorithm [9].

We further perform segmentation of a cable by modeling it as a set of cylinders. We propose a search structure that allows to find the cables present in the real point cloud and to reconstruct them as collections of cylinders of varying radius, length, and orientation. Each cylinder, called sub-cable is fitted, using the point-to-point ICP algorithm [10], to the point cloud data corresponding to a cable. More details about our method can be found in our recent papers [2] [3].

The final result of the cable segmentation phase can be seen in Fig. 17.



(a) (b)

Figure 17: (a) the final cylinder model (made of 12 sub-cables) in green and, in blue, a detected hole in the cable due to two overlapping cables (1 sub-cable) and (b) the segmented point cloud

Fig. 18 and Table 4 illustrate the final cylinder model shown in Fig. 17 made of 13 consecutive sub-cables labeled sc_1 through sc_{13}. Fig. 18 shows the details of the 13 sub-cables: their radius, the length, the angle between two consecutive main axis and the direction of the propagation, where '+' denotes first direction and '-' denotes second direction. The angle between two consecutive main axis θ_i can easily be computed by considering each main axis to be a vector and by taking the dot product of the two vectors:

$$v_1 \cdot v_2 = |v_1||v_2| \cdot \cos(\theta_i) \quad (1)$$

4.2. Interference detection and bend radius computation

The global objective of the inspection process is to measure (1) the minimum distance between each segmented cable and the other elements in the mechanical assembly (in order to check a possible interference problem), and (2) the bend radius of each segmented cable (in order to check that it complies with safety standards).

4.2.1. Inputs

After the segmentation process, we have two main point clouds: the segmented point cloud, which we will call $\mathcal{P}_s$, and all the other points, excluding the segmented point cloud, which we will call $\overline{\mathcal{P}}_s$ (see Fig. 19). If we call $\mathcal{P}$ the filtered input 3D point cloud provided by the 3D scanner, we have $\mathcal{P} = \mathcal{P}_s + \overline{\mathcal{P}}_s$.

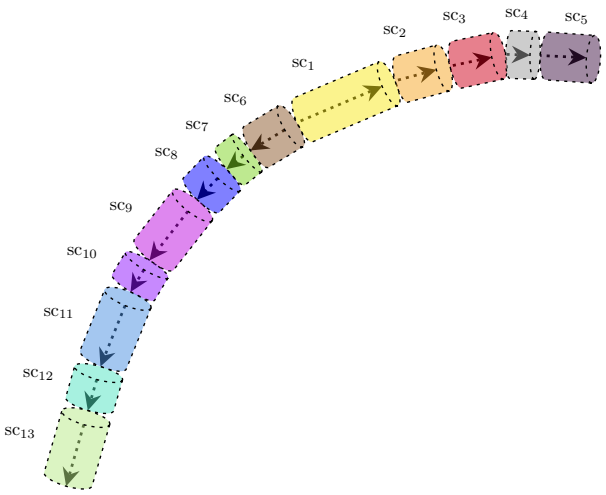


Figure 18: An illustration of the final cylinder model shown in Fig. 17 made of 13 consecutive sub-cables labeled sc_1 through sc_{13} (different colors denote the sub-cable models)

Table 4: The details of the 13 sub-cables: their radius, the length, the angle between consecutive main axis and the direction of propagation, where '+' denotes first direction and '-' denotes second direction. It should be noted that sc_8 is a detected hole due to the overlapping cables (see Fig. 17) and its radius is equal to the radius of previous sub-cable sc_7

Sub-cable	Radius	length	Angle	Direction
sc_1	5.81	33.55	–	+
sc_2	5.93	22.09	8.51	+
sc_3	5.89	17.83	2.12	+
sc_4	5.97	17.97	11.4	+
sc_5	5.91	19.26	3.7	+
sc_6	5.90	16.43	4.12	-
sc_7	5.91	10.01	2.02	-
sc_8	–	13.20	3.22	-
sc_9	5.93	22.01	1.7	-
sc_{10}	5.98	10.66	1.1	-
sc_{11}	5.92	19.44	4.3	-
sc_{12}	5.79	11.81	12.3	-
sc_{13}	5.80	18.02	11.2	-

4.2.2. Interference detection

In this section we will focus on the detection of a possible interference between a cable and the other surrounding elements (another cable, a support, etc.) present in the mechanical assembly.

The objective is to determine if the cables are at a safe distance $d_{\mathcal{T}}$ from the other surrounding elements present in the mechanical assembly. The result of the interference

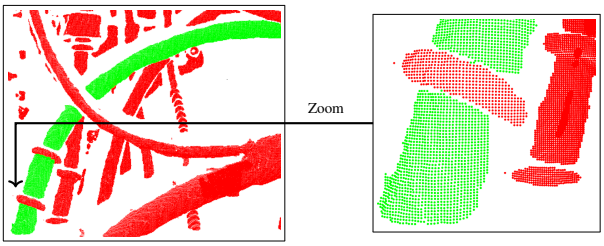
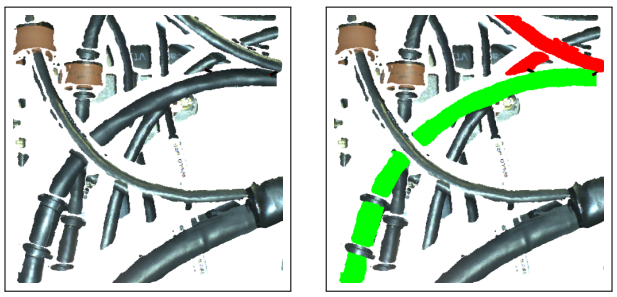


Figure 19: The segmented point cloud, which we will call $\mathcal{P}_s$ in green and all the other points, which we will call $\overline{\mathcal{P}}_s$ in red

detection process is shown in Fig. 20.



(a) (b)

Figure 20: Result of interference detection process: (a) the input point cloud, (b) segmented point cloud in green and representative clusters which cause interference problem in red

Our experiments have been detailed extensively in our recent paper [3].

For one of the experiments we use the available CAD model. We identified 4 cables and measured the distance between each cable and its surrounding elements in the assembly. We do this with a specialised modelling software CATIA©. Further, we sampled the cables and the elements and added noise to the sampled point cloud. This step simulates the noise introduced by the process of scanning. We added three different levels of noise. Finally, we run our algorithm on all generated clouds and obtain the minimal distance between each segmented cable and the other elements. We compare this result with the measure obtained with CATIA©. Table 5 shows that the obtained results are satisfactory.

4.2.3. Bend radius calculation

The other objective of the inspection process is to measure the bend radius of each segmented cable (in order to check that it complies with safety standards).

The minimum bend radius is the smallest allowed radius the cable is allowed to be bent around. The 3D segmentation method presented in section 4.1 produces almost immediately a cable model segmented from the measurement data (see Fig. 21a). Using this model, we can carry out a quantitative analysis of the bend radius of the cable.

Table 5: Interference measurement results compared with the CATIA© modelling software

Standard deviation of added noise	Measure obtained with CATIA	Measure obtained by our algorithm
0	13.5	13.34
0.4	15.5	14.63
0.8	14	12.23
1.2	18.5	16.38

In order to estimate the bend radius of a cable, we fit a 3D plane to the set of start-points P_s^k and end-points P_e^k for all the estimated sub-cables sc_k (see Fig. 21a). Further we project all the points onto the fitted plane (see Fig. 21b). Finally, the bend radius is estimated by fitting a circle to the set of projected 2D points (see Fig. 21c). Therefore, the accuracy of the measure is directly related to the robustness of the segmentation algorithm, fitting plane algorithm and fitting circle algorithm.

Since we do not have the needed instrumentation to measure precisely the bend radius of the cables within the mechanical assembly, we have decided to test our approach on synthetic data generated from the CAD model. A few examples of our dataset are shown in Fig. 22.

The minimum allowed bend radius is based on the diameter and the type of cable. By industry standards, the recommended bend radius for harnesses should not be lower than 10 times the diameter of the largest cable.

The results are evaluated on the basis of the difference between the ground truth bend radius R_c and the estimated bend radius $\widetilde{R}_c$:

$$MSE_{\text{bend\_radius}} = \frac{1}{n}\sum_{i=1}^{n}(R_{c_i} - \widetilde{R}_{c_i})^2 \qquad (2)$$

On a set of 3 different cables with different, radius, resolutions and different bend radius, we found $MSE_{\text{bend\_radius}}$ = 1.23 mm.

As expected, we have found that the results are better when the number of sub-cables found is high. Indeed, the fitting of plane or circle are more accurate when the set of start-points P_s and end-points P_e contains more points.

5. Conclusion

We proposed an automated inspection system for mechanical assemblies in an aeronautical context. We address the problem of inspection in two parts: first, automatic selection of informative viewpoints before the inspection process is started, and, second, automatic treatment of the acquired images and 3D point clouds from said viewpoints by matching them with information in the 3D CAD model.

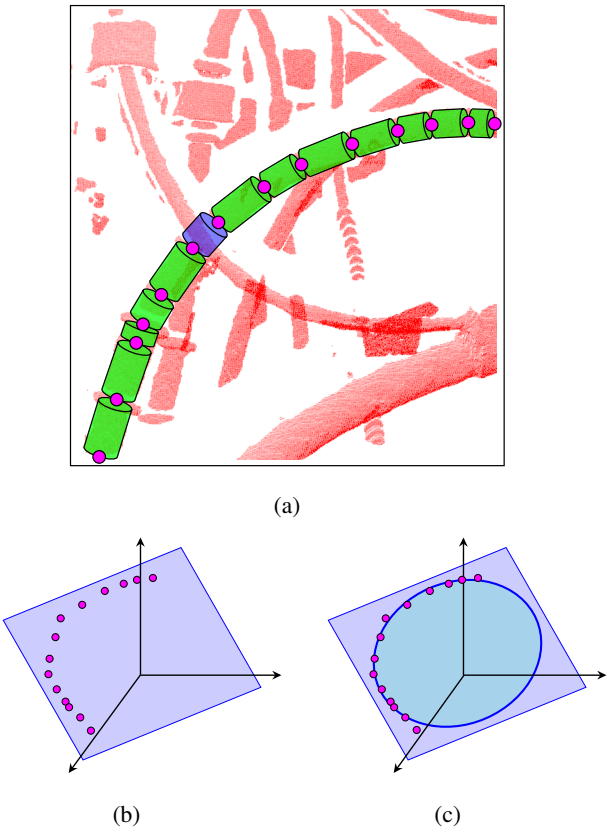


Figure 21: Steps for calculating the bend radius: (a) segmentation modeling result, where each sub-cable sc_k (green) has an initial point P_s^k and a final point P_e^k (both in pink), (b) projection of all the points P_s and P_e onto the 3D plane previously obtained by fitting a plane to those same points with a least squares method, and (c) fitting a circle by least squares method to these projected 2D points

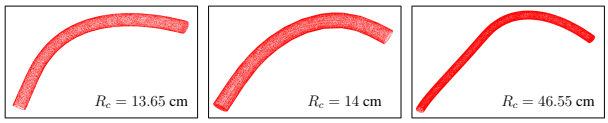


Figure 22: A few examples of our dataset. Three different synthetic cables with different bend radius (R_c)

Depending on the inspection type and speed/accuracy trade off, we propose two strategies: the first one is based on 2D image analysis and the second one relies on 3D point cloud analysis.

With our 2D strategy we are focusing on detecting missing and badly mounted rigid parts by comparing sensed information with the CAD model.

In the second part of the paper we give an overview of our 3D based method for cable segmentation based on cylinder fitting. Using the segmentation result and the CAD model, we can carry out quantitative analysis of potential interferences and the computation of the bend radius of each cable.

The experimental results show that our method is highly accurate in detecting defects on rigid parts of the assembly as well as on aircraft's Electrical Wiring Interconnection System (EWIS).

Acknowledgements

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] H. Ben Abdallah, I. Jovancevic, J.-J. Orteu, L. Brèthes. Automatic inspection of aeronautical mechanical assemblies by matching the 3D CAD model and real 2D images. *Journal of Imaging*, 8(10):81-110, October 2019.

[2] H. Ben Abdallah, J.-J. Orteu, B. Dolives, I. Jovancevic. 3D Point Cloud Analysis for Automatic Inspection of Aeronautical Mechanical Assemblies. *14th International Conference on Quality Control by Artificial Vision (QCAV)*, Mulhouse, France, May 2019.

[3] H. Ben Abdallah, J.-J. Orteu, I. Jovancevic, B. Dolives. 3D Point Cloud Analysis for Automatic Inspection of Complex Aeronautical Mechanical Assemblies. *Journal of Electronic Imaging* (submitted in 2019).

[4] A. Loesch, S. Bourgeois, V. Gay-Bellile, M. Dhome. Generic edgelet-based tracking of 3D objects in real-time. *International Conference on Intelligent Robots Systems (IROS)*, pp. 6059-6066, 2015.

[5] S. Xie, Z. Tu, Holistically-Nested Edge Detection. *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[6] J. Canny, A Computational Approach to Edge Detector. *IEEE Transactions on PAMI*, pp. 679-698, 1986.

[7] S. Belongie, J. Malik, J. Puzicha. Shape matching; object recognition using shape contexts. *IEEE Transactions on Pattern Analysis; Machine Intelligence*, *24*, pp. 509-522, 2002.

[8] P.J. Besl, N.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 239-256, 1992.

[9] C. Kim, H. Son, C. Kim. Fully automated registration of 3D data to a 3D CAD model for project progress monitoring. *Image and Vision Computing*, pp. 587-594, 2013.

[10] S. Rusinkiewicz, M. Levoy. Efficient variants of the ICP algorithm. *Proceedings of Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145-152, 2001.