A digitized approach to reduce assembly conflicts during aircraft cabin conversions

Fabian Laukotka[1], Jan Oltmann[1], Dieter Krause[1]

[1] Institute of Product Development and Mechanical Engineering Design (PKT), Hamburg University of Technology (TUHH), Germany

Abstract

Aircrafts are fitted with a new cabin every five to seven years. Because each individual aircraft is slightly modified during its life span, detailed information to be used during the design of the cabin are rarely available. Therefore, many conflicts between the cabin and the aircraft's body arise during the actual conversion. While modern technologies, like 3D-scanners, are becoming more common, their comfortable use within the CAD-Workflow is often not possible or require extensive processing and additional work. This paper presents an approach, integrated into the CAD-workflow, to directly use pointclouds provided by 3D-scans of aircrafts to identify possible conflicts with the cabin already during its design-phase by performing a clash-analysis between the points and the CAD-solids.

Keywords: Aircraft Cabin Conversion, 3D-Scans, Clash Analysis, CAD vs pointcloud

1 Motivation

In modern aviation, about every five to seven years an aircraft is refitted with a new cabin interior. Prior to the actual refitting, the new cabin will be designed but the planning data for the design often lacks digital 3D models and is limited to 2D drawings. In many cases the design is only based on general available data about the type of aircraft and detailed information about the specific aircraft are missing. As a result, the information used for the design will not necessarily match those of each of the single aircrafts. Thus, during the actual conversion on-site, much time is spent on adapting the designed cabin to the actual structure of the aircraft's body. To shorten the time spent on ground, a new way of determining these conflicts even before the aircraft arrives at the facility is needed.

2 Research Problem und Research Aim

To gather detailed information about the structure of an aircraft a multitude of measurements with state-of-the-art 3D scanning devices is possible. 3D-Scanners can generate an accurate digital model, in the form of a dataset of thousands of single points. As already has been shown [1], scanning aircraft cabins result in detailed information that are usable for 3D-CAD-operations.

A test of different CAD-Software has shown, that functions for clash-analysis are already included in many of them. An assembly consisting of the designed aircraft-cabin and the dataset from the 3D-Scanner could show conflicts that otherwise would just be shown during the actual refitting. Nevertheless, currently these analyses are not able to process pointclouds but rely on fully defined CAD-solids. The pointclouds would have to be enhanced to solids, which is generally possible, but especially in case of aircraft-scans requires manual interaction and a lot of computing power and is for this application not feasible (Figure 1). Using additional knowledge about the structure during this process, the enhanced CAD-data will represent a more detailed model of the aircraft. However, depending on the algorithms used for these operations, other details may be lost for example due to interpolations.

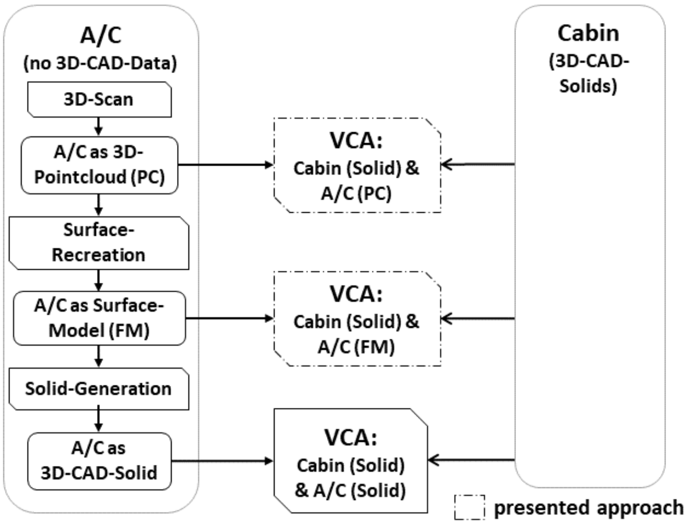
A/C
(no 3D-CAD-Data)
3D-Scan
A/C as 3D-Pointcloud (PC)
Surface-Recreation
A/C as Surface-Model (FM)
Solid-Generation
A/C as 3D-CAD-Solid
VCA: Cabin (Solid) & A/C (PC)
VCA: Cabin (Solid) & A/C (FM)
VCA: Cabin (Solid) & A/C (Solid)
Cabin
(3D-CAD-Solids)
presented approach

Figure 1: Sate-of-the-art and provided workflow of performing a virtual clash analysis (VCA) using pointclouds and solids

The virtual clash-analysis can also be done between the cabin's 3D-CAD-solid and a pointcloud or surface-model of the aircraft's body. In current researches, approaches to analyse a set of pointclouds can be found. Albeit, the combination of 3D-solid and pointcloud seems to be fairly new. This is especially true, regarding the convenient usage within CAD-environments. Nevertheless, using pointclouds instead of surface models or solids may lead to a possible less detailed representation of the aircraft, depending on the density of the pointcloud produced by the 3D-scanner. For further analyses a reconstruction of selected local areas of the pointcloud may also be possible.

This paper presents a new approach to perform a clash-analysis between CAD-solids and pointcloud-datasets within a CAD-environment, that can be generated by a 3D-Scanner without the need of extensive preparations of these datasets. Another approach using the surface model of the aircraft to perform the clash analysis is currently in the works, yet, the presented approach focuses on the usage of pointcloud-datasets. The state-of-the-art workflow as well as the two described adaptions are visualised in Figure 1.

The clash-analysis itself will be complemented with tools to directly use the detected interferences to adapt the CAD-construction, thus, preventing the subsequent occurrence of these interferences. The objective is to provide an integrated tool that can assist not only to identify these interferences but also allow for the adaption of affected components based on the analysis-results directly from within the user interface (UI) of the CAD environment.

3 Methodical Approach

Considering the basic task of performing a clash-analysis between a point-cloud and CAD-solid, the processing results in the repeated geometric task to determine whether a single 3D-point is within a closed polyhedron or not. This process needs to be repeated for every point and every solid. The mathematical approach used for this task as well as the approach to implement it into CAD-environments are described in the following chapters.

3.1 Mathematical Approach

In two-dimensional geometrics, the task to determine whether a point lies within a closed area is also known as the point-in-polygon-analysis and occurs in many computing situations. Therefore, different approaches to solve the task were already established [2].

3.1.1 Point in Polygon Tests using the Ray-Casting-Method

The ray-casting-method is a well-established procedure to perform a 2D point-in-polygon-test. The premise to use this method is that the polygon's boundary is a closed Jordan Curve as defined in the Jordan Curve Theorem [3][4]. Based on this premise, a semi-infinite ray starting at the test-point is casted in an arbitrary direction. Whenever this ray hits the boundary of the polygon a counter i is incremented. This counter i can attain the following values:

$$i\begin{cases} 0 \\ 2n \quad | \, n \in N \\ 2n+1 \, | \, n \in N \\ \infty \end{cases}$$

Following the value of i can be used to determine whether the test-point lies within the polygon or not: In case the ray never intersects the polygon, the test-point is positioned outside of the polygon. The same applies whenever there is an even number of intersections, because the ray enters and leaves the polygon a number of times. On the contrary, whenever there is an odd number of intersections, the test-point lies inside of the polygon. In case the test-point lies directly on the polygon's boundary, an infinite number of intersections can be found [3]. In Figure 2 a 2D-polygon is shown next to a 3D-polyhedron. Additionally, several test-points are added to both objects. Starting from these test-points the ray-casting-method is visualised using arrows indicating the ray-direction as well as crosses indicating the intersection with the object's boundary.

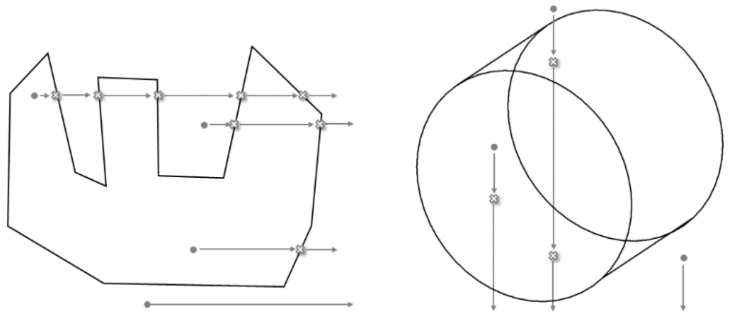
Figure 2: Visualisation of the ray-casting-method in 2D and 3D

As a 3D-object has no simple equivalent to a Jordan Curve that is applicable to this theorem, this previously described method cannot be applied to 3D-polyhedra in general. At the present moment, no equivalent methods to be used in 3D are known. However, the basic steps of this method can be adapted. Similar to a Jordan Curve that defines an object with a closed boundary in 2D, a 3D-CAD-solid can be described as an homogenic object that has a fully defined, closed boundary allowing for a differentiation between inside and outside [5]. Using this approach, the 2D point-in-polygon-analysis using the Ray-Casting-method can be adapted to a 3D point-in-polyhedron-analysis.

A 3D-polyhedron consists of faces, edges and vertices. Again, the ray is casted from the 3D-testpoint in an arbitrary direction. In case the test-point lies within the polyhedron, the casted ray needs to cross at least one of the previously named boundary-types once. Whether the ray intersects a specific face of the polyhedron can be determined by testing if the test-point can be projected onto the face along the ray. If the projection P′ of the test-point P along the ray R, $P' \in R$, lies on the face F, $P' \in F$, an intersection was found. In the rare case that the ray does not intersect a face but rather an edge or vertex of the polyhedron, additional tests are required. These tests are needed to determine if the ray enters or leaves the polyhedron via this edge or vertex, and thus, the counter needs to be incremented, or if the ray only scrapes the edge or vertex without entering or leaving the body. This can easily be done by performing additional ray-casting tests with temporary test-points directly before and after the determined projected point P′ along the ray. If the results of these two test-points differ, the polyhedron's boundary is crossed and the counter needs to be incremented.

3.1.2 Optimisation using an Octree-Structure

As standard pointclouds usually do not store the points in a known structured way but rather simply store the coordinates for each point, this procedure theoretically needs to be done for each single point of the pointcloud. Even if one point is known to not be interfering, there is no way of eliminating other points without testing them each individually.

To speed up this process the algorithm was enhanced by structuring operations, that allow the premature exclusion of complete areas of the pointcloud. This is realised by clustering the points of the pointcloud into neighbourhoods. By splitting this sub-area into smaller and smaller boxes, a tree structure is created. Because each split creates a new level of eight sub-areas, this commonly used technique is called Octree-Structure and can be found in several others geometrical tasks. Each of the sub-areas of the structure can be represented by its bounding box, which can be used to perform pre-tests against the test-objects bounding box. To test whether two boxes are intersecting can easily be done and is already included into SolidWorks. By performing these bounding box tests, complete subareas of the workspace can be eliminated from the actual ray-casting algorithm as described above. Although this produces more calculations beforehand, these calculations are faster than the actual ray-casting tests and depending on the specific layout of the workspace, test-object and pointcloud this additional time can lead to extensive time-saving because of the premature exclusion of test-points. Certainly, depending on the size of the pointcloud, the creation of the Octree-Structure might also require additional time.

3.2 CAD Approach

As already indicated in [1], the algorithms were implemented in form of an Add-In for the established CAD-Software SolidWorks. A custom UI allows for the easy interaction with the Add-In from within SolidWorks. Because of the integration into SolidWorks its functions can be used to perform basic CAD-Operations. Additional operations are added using the API provided by SolidWorks. As part of the ray-casting algorithm directly interacts with the CAD-data in SolidWorks, these operations cannot be done without using the API. However, other operations, like the Octree-Structuration can be done within the Add-In and benefit von techniques like performing multiple calculations simultaneously (Multi-Threading) which currently is not possible while using the API.

To enable Multi-Threading and reduce the workload of SolidWorks while maintaining as accurate calculations as possible, the Dual-Pointcloud-Method was developed, which splits the available pointcloud data into two separate files with a different level of details. The workflow using this method is visualised in Figure 3. One pointcloud with reduced details will be imported into SolidWorks and used to align it to the cabin's CAD-data as well as verify the dataset visually. The transformations used for this alignment will be stored and applied to the pointcloud with full details during the actual analysis. This second pointcloud will be imported directly into the Add-In without using the SolidWorks-API, thus, bypassing the API's bottleneck. Additionally, a selection-box can be placed within the workspace to define the area to be used for the analysis. Using the UI several settings regarding the analysis, like thresholds, can be set.

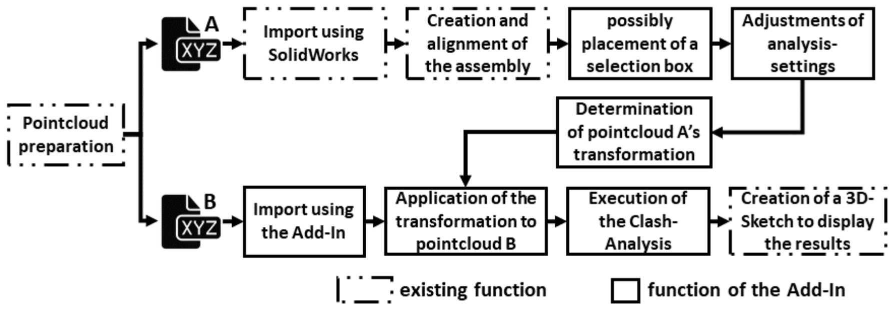
Pointcloud preparation
A
XYZ
Import using SolidWorks
Creation and alignment of the assembly
possibly placement of a selection box
Adjustments of analysis-settings
Determination of pointcloud A's transformation
B
XYZ
Import using the Add-In
Application of the transformation to pointcloud B
Execution of the Clash-Analysis
Creation of a 3D-Sketch to display the results
existing function
function of the Add-In

Figure 3: Flow-Chart of the Dual-Pointcloud-Analysis

The result of the analysis is a set of interfering points, stored within a 3D-Sketch in SolidWorks. This sketch is then part of the saved SolidWorks-File and can be accessed without the need to run the analysis again. Supplementary the interfering points can be exported as a new pointcloud-file using the same coordinate system as the original pointcloud.

The clash-analysis is accompanied by tools that allow further analysis of detected interferences like measurement-tools and the possibility to directly adapt the local geometry of conflicting CAD-solids. These operations use the points of the generated 3D-Sketch. Because they are in form of SolidWorks standard elements, they can be used to directly adapt the cabin's CAD-data, by performing standard CAD-operations of SolidWorks. Screenshots of the different steps using tests-assemblies in SolidWorks are shown in Figure 4.

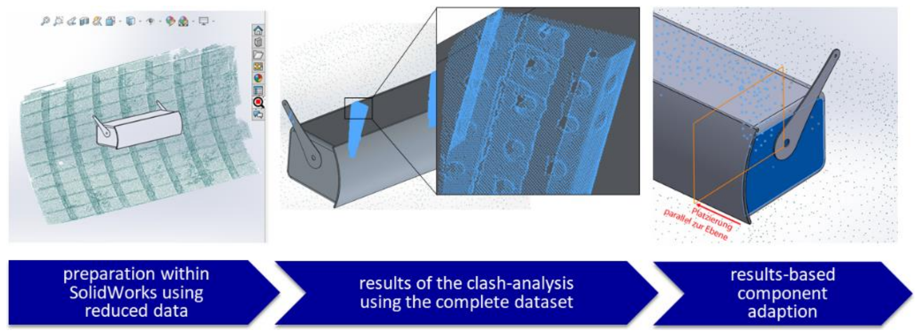
Platzierung
parallel zur Ebene
preparation within SolidWorks using reduced data
results of the clash-analysis using the complete dataset
results-based component adaption

Figure 4: Essential steps of the presented approach as presented in [1]

4 Results

Whereas interference-detections between two pointclouds can already be found in publications, e.g. [6], the approach to enable a direct analysis of a combination of a pointcloud and a solid fully integrated into an existing CAD-Environment was not encountered during the investigation. Multiple aspects of the presented approach were tested using synthetic pointcloud-datasets as well as actual 3D-Scans. The results of the analysis using the described Dual-Pointcloud-Method as well as the structuration of the Dataset into an Octree-Structure are described subsequently.

4.1 Detected interferences using the Dual-Pointcloud-Analysis

The approach has been tested using existing 3D-Scans of parts of an aircraft-body and manually placed interior-solids. The number of points of the tested pointcloud reached from around 150.000 to over 5.700.000. These tests have shown that the direct analysis of pointclouds and CAD-solids is possible and eventuate in a clear identification of conflicts between the solid and the scanned aircraft-body.

A smaller test was conducted using a single Hatrack and a 3D-Scan of part of an aircraft-fuselage. The Hatrack was specially positioned to deliberately create interferences with the fuselage’s structure. As can be seen in Figure 5, these interferences were identified correctly. The different densities of the used pointclouds as described in 3.1.2 are clearly noticeable. The blue dots represent identified interferences stored as a 3D-Sketch in the SolidWorks-Workspace. The smaller green dots are part of the reduced pointcloud, imported into SolidWorks to be used for the alignment.

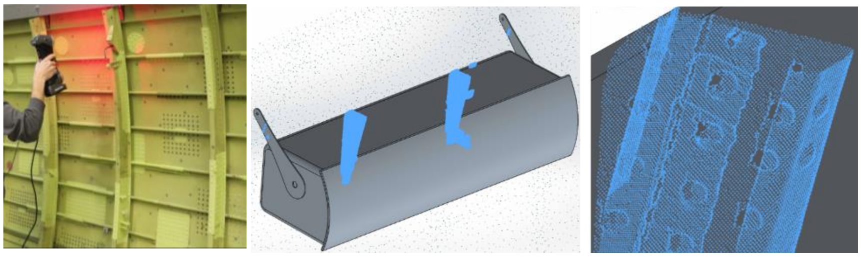
Figure 5: Testing the Clash-Analysis

Another test using two Hatracks positioned at a quarter scan of an aircraft's fuselage leads to similar results. Again, the interferences were identified correctly (Figure 6). The difference in density between the pointcloud used for alignment and the pointcloud used for the analysis is even more noticeable as even the structure of a windows of the fuselage can clearly be seen in the resulting interferences. The upper image shows a coloured 3D-scan of part of an aircraft. In the lower left image, a reduced version if this pointcloud is imported into SolidWorks and two Hatracks in form of CAD-solids are added to the assembly. The results of the analysis between the solids and pointcloud is shown in the lower right image.

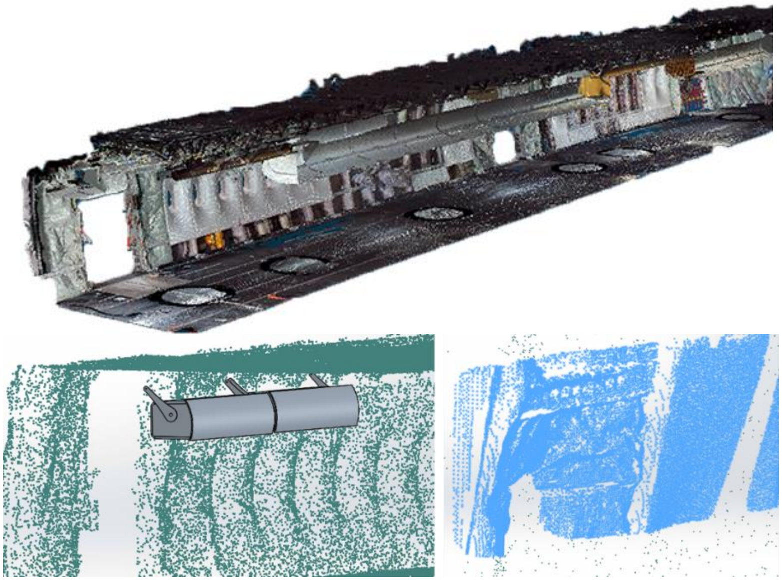
Figure 6: Further Tests of the Clash-Analysis

Besides these two tests, a number of repetitions and other structures were used to verify the Add-Ins functionality. All these tests show, that a clash-analysis using CAD-solids and pointclouds can be realized using the presented approach.

4.2 Impact of the dataset-structuration

To determine the possible effect of the usage of the Octree-Structure, multiple tests using the same CAD-data with and without structuring the points were conducted. For each of these tests the time needed to import the points and perform the analysis were measured. In case the structuration is enabled, the import will require additional time as it is done during this task. The tests were conducted with several pointclouds one of these being an actual 3D-scan of a section of an aircraft. The number of points reaches from 10.000 up to 5.700.000. For the tests with up to 250.000 points a standard notebook utilizing an Intel Core-i5 Processor was used, while the test of the aircraft-scan with 5.700.000 points was done on an CAD-Workstation. As can be seen in Table 1, the usage of the Octree-Structure reduces the time needed to perform the analysis significantly while requiring only minimal additional time during the import. This can be ascribed to the possibility to perform Multi-Threading while staying within the Add-In during the import but relying on the API-Limitations during parts of the analysis.

Table 1: Analysis-Time with and without Octree-Structuring

Number of points	unstructured			structured		
	Import	Analysis	Total	Import	Analysis	Total
10.000	$< 1\,s$	$28\,s$	$< 29\,s$	$1\,s$	$28\,s$	$29\,s$
40.000	$2\,s$	$252\,s$	$254\,s$	$3\,s$	$200\,s$	$203\,s$
250.000	$140\,s$	$4313\,s$	$4453\,s$	$158\,s$	$2880\,s$	$3038\,s$
5.700.000	$60.480\,s\ (16h\ 48\ \mathrm{min})$			$2700\,s\ (45\ min)$		

Summarising, it can be seen, that the use of the Octree-Structure extends the import-process slightly but gains a big time-benefit during the analysis. Another big influence is the actual layout of the assembly.

5 Discussion and Outlook

Using the described approach implemented into a SolidWorks Add-In it was possible to show, that the direct Clash-Analysis between a scanned pointcloud and CAD-solids results in a clear identification of interferences. Thereby, the surface-reconstruction of the aircraft’s CAD-data is not necessary anymore, saving time and computational resources. The identified interferences then can easily be used to modify the cabin’s CAD-data directly from within the same working environment. Nevertheless, there still is room for improvements of the algorithm and overall handling of the big dataset, especially regarding the possible size of complete cabin-scans. The interaction of the Add-In with SolidWorks can also be improved, as the exchange of big datasets with SolidWorks takes a lot of time because the main SolidWorks-API does not benefit from the use of multithreading. However, the inclusion into an already existing CAD-Software provides a benefit for the user as the analysis can be included into already existing workflows and no additional software is needed.

Without further optimisation, the analysis might take several hours, hence it is not feasible to perform many of these analyses during a standard workday. A way to evade this is to automatically perform the clash-analysis during after-hours or using dedicated hardware while manually realising the resulting adaptions during the workday. Nevertheless, an improvement in the overall performance is preferable, especially as the required time increases with the scope of the complexity of the assembly.

As many of the operations are performed multiple times, slight improvements in their required time can lead to big improvements of the overall processing time. During the tests, the limitations of the used interaction with the SolidWorks-API emerges as the main bottleneck of the approach. By optimizing this elementary part of the algorithm, a big speed up can be expected. To improve parts of the analysis by improving the API-capabilities, contact to SolidWorks is already established. Currently the complete processing is done using the CPU of the computers. As some calculation-heavy programs are already using the graphics card for some of their calculations, an investigation into the possibility to use a graphics card for parts of the algorithm might also lead to an overall improvement, especially regarding the new graphic cards support specialised operations for ray-tracing-algorithms. As other programs, can already handle big datasets of pointclouds, the best would be to combine the performance of these with the presented approach to create a professional grade environment to view and analysis pointclouds as well as use the resulting information to perform CAD-design operations.

Despite the need for further improvements, the presented approach has shown, that in the given situation of combining pointclouds from 3D-scanned aircrafts with CAD-solid from the cabin design, assembly conflict that otherwise would just be identified during the actual cabin conversion can be identified early on.

Literature

[1] Deneke C., Oltmann J., Schüppstuhl T., Krause D.: AST: Technology Innovations For A Faster Aircraft Cabin Conversion, Hamburg - Germany, 2019

[2] Shimrat, M., "Algorithm 112: Position of point relative to polygon" Communications of the ACM, Volume 5 Issue 8, 1962

[3] Eric Haines, "Point in Polygon Strategies" in Graphics Gems IV (1994)

[4] Akenine-Möller, T., Haines, E., & Hoffman, N.: "Real-Time Rendering - Third Edition". Boca Raton: CRC Press - Taylor & Francis Group, 2008.

[5] Hoffmann, C. M.: "Geometric and Solid Modeling", West Lafayette: Purdue University College of Science, 1992

[6] Eberly, D. H., & Schneider, P. J.: "Geometric Tools for Computer Graphics", San Francisco: Morgan Kaufmann Publishers, 2003

[7] Figueiredo, M., Oliveira, J., & Araújo, B.: An efficient collision detection algorithm for point cloud models, Faro - Portugal, 2010