



Automatic cabin virtualization based on preliminary aircraft design data

Jan-Niclas Walther¹ · Bahadır Kocacan¹ · Christian Hesse¹ · Alex Gindorf¹ · Björn Nagel¹

Received: 26 February 2021 / Revised: 15 December 2021 / Accepted: 16 December 2021 / Published online: 8 January 2022
© The Author(s) 2022

Abstract

Preliminary aircraft design and cabin design are essential and well-established steps within the product development cycle for modern passenger aircraft. In practice, the execution usually takes place sequentially, with the preliminary design defining a basic cabin layout and the detail implementation following in a subsequent step. To enable higher fidelity assessment of the cabin early in the design process—for example by means of virtual reality applications—this paper proposes an interface, which can derive detailed 3D geometry of the fuselage from preliminary design data provided in the Common Parametric Aircraft Configuration Schema (CPACS). This is a key step towards integration of cabin analysis and preliminary design in automated collaborative aircraft design chains, not only in terms of passenger comfort, but also manufacturability or crash safety. Like the TiGL Geometry Library for CPACS, the interface presented acts as a parameter engine, which translates data from CPACS into CAD geometry using the Open Cascade Technology library. However, the scope of TiGL is expanded significantly, albeit with an explicit focus on the fuselage, by including more details such as extruded frame and stringer profiles and floor structures. Furthermore, advanced knowledge management techniques are employed to detect and augment missing data. For virtual reality applications, triangulated representations of the CAD geometry can be provided in established exchange formats, creating an interface to common visualization platforms. Additionally, a new evolution of the cabin definition schema in CPACS is presented, to incorporate models of cabin components such as seats or sidewall panels enabling immersive virtual mock-ups.

Keywords CPACS · Cabin · Structures · Virtual reality · Open Cascade Technology

1 Introduction

Cabin design is inherently an essential aspect of the development of passenger aircraft. The number of passengers, as well as the arrangement of seats and aisles have an immediate effect on fundamental design parameters such as length and cross-section of the fuselage. Similarly, a good understanding of the mass distribution within the fuselage is important for analysis of the flight mechanics. However, a full three-dimensional mock-up is rarely available in the early stages of the design process.

Due to the increasing tendency towards individualized cabin experience between carriers, the timely evaluation of a

variety of different cabin concepts from a passenger perspective is becoming an essential capability. Allowing potential customers to experience a selection of concepts quickly could prove to be a key advantage for original equipment manufacturers (OEMs). Virtualization, i.e., the realistic virtual reproduction of the physical product using e.g. virtual reality (VR) technology, is a promising and cost-effective method to implement such a service. In addition, it can provide engineers with a tool to gain an intuitive understanding of complex product data in an interactive environment. Representative applications for human centric design or engineering exploration have been described by De Crescenzo et al. [1] and Fuchs et al. [2], respectively.

The applications show, that effective virtualization requires a digital mock-up of the cabin geometry to be provided, which needs to be sufficiently detailed to provide test subjects with the sensation of being in a real cabin. As a result, the requirements for details are high compared to applications from other disciplines such as structural

✉ Jan-Niclas Walther
jan-niclas.walther@dlr.de

¹ Institute of System Architectures in Aeronautics,
German Aerospace Center (DLR), Hein-Saß-Weg 22,
21129 Hamburg, Germany

analysis making cabin virtualization a suitable test case. This paper describes an approach linking preliminary aircraft design, structure and cabin layout generation and geometry modeling using the Common Parametric Aircraft Configuration Schema (CPACS) [3] to provide a mock-up of the fuselage and cabin automatically while maintaining consistency among different disciplines. It is designed to be operated as part of collaborative multi-disciplinary digital design processes and as such is designed to run fully automatically. To this end, product information, which is not provided in the incoming data set but required for the model generation, must be augmented automatically. This is implemented using a knowledge-based engineering approach [4]. The possibility to automatically and dynamically add necessary information based on the available data at a given stage of an interdisciplinary process is a distinguishing factor to similar solutions for cabin design and virtualization found in industry and research such as the commercial tool Pachelab ACE myCabin [5] or the open source CabLab [6], which are monolithic applications operated through a graphical user interface. Establishing these digital capabilities is a key contribution towards the vision of the digital thread, advocated by the German Federal Government, the Helmholtz Association and the German Aerospace Center (DLR) [7–9], to improve digital permeation and interconnection throughout the aerospace industry, reducing lead times and entrepreneurial risks for new, greener aircraft designs.

The following sections introduce an approach, where preliminary design data are augmented based on a limited set of parameters and a virtual mock-up of the fuselage and cabin is derived, which suits the requirements for cabin virtualization. At first, the prerequisites for modeling the necessary product information using CPACS are introduced in Sect. 2. Then, the knowledge-based methods to generate the information required for building the digital mock-up based on preliminary design inputs are discussed. This includes a discussion of the geometric modeling rules for the transformation of parametric data in CPACS to CAD shapes for relevant components of the fuselage structure. Finally, in Sect. 4, it is demonstrated how the methods can be applied to create a digital mock-up from preliminary design data for a mid-range aircraft concept.

2 Fuselage structure and cabin description using CPACS

The cabin design is inevitably linked to a structural layout, the structure being the physical link between the cabin components and the outer mold line. Since such a structural layout may not be available in the beginning of the process, a structural design may need to be created in a preprocessing step. Consistency of design data is paramount when

integrating multiple disciplines in this way. Therefore, it is advisable to use a common underlying data model to link the design competences. A proven option is CPACS, a schema for hierarchically structured XML data sets to describe aircraft. The activities surrounding CPACS tend to be focused on preliminary design, but the format is designed for use across multiple disciplines and levels of fidelity. Therefore, detailed structure and cabin descriptions are available, making CPACS a suitable choice for coupling preliminary and cabin design. Furthermore, it is supported by established overall aircraft [10] and fuselage structure design tools [11], which can provide inputs and form the basis for subsequent developments.

2.1 Fuselage structure definition and initialization

The capabilities for describing primary structures of fuselages in CPACS have been covered in depth by Scherer and Kohlgrüber [12]. Aside from stringers, frames and skin panels, floor structures consisting of crossbeams, struts and longitudinal beams such as seat rails, can be defined. Descriptions of other key components, such as bulkheads and wing-fuselage connection areas are also provided and openings in the structure, e.g. for passenger or cargo doors can be specified. Analogously to beams and shells in structure mechanics, the structural descriptions are divided into profile and sheet based components. Profile-based elements are defined by a topologically one-dimensional curve, along which a two-dimensional cross-sectional profile is extruded. The curve is defined either as the intersection of the fuselage outer surface and a definition surface like in the case of frames and stringers, or as the segment of an infinite or semi-infinite curve that lies within the fuselage as for the floor structure. The profile definitions are stored in a separate node for structural elements, which can be understood as a library of semi-finished structural parts. Sheet-based elements assign a thickness property to a preexisting surface, such as a skin panel, which may be bounded e.g. by frame and stringer curves. Both definitions also provide information on the material.

Methods for initializing a structural layout for a given empty outer fuselage loft, e.g. the output of a preliminary design process have been explained at length by Walther and Ciampa [11]. It shows, how the fuselage structure, which typically consists of a large number of entries in CPACS, can be generated automatically based on very few control parameters such as bulkhead positions and a nominal frame pitch. The capabilities have been extended for the cabin design use case by adding automatic distribution of cutouts and providing more detailed information on the bulkheads.

The cutout nodes are used to describe structural openings such as windows or passenger and cargo doors. The definitions are already provided by CPACS. They are given

in the global aircraft coordinate system, where the x and z coordinates correspond to the aircraft's longitudinal and vertical axes, respectively. As illustrated by Fig. 1, the cutouts are positioned at a point on the fuselage surface. Analogously to e.g. stringer positions, it is defined by a point and a reference angle, which can be used to construct an intersection vector with the fuselage surface. At the intersection point, a rectangle of width deltaX height deltaY is extruded along an orientation vector (orientationVector). The dimensions of the rectangle are defined in a local coordinate system, determined by an alignment vector (alignmentVector). The corners of the rectangle can be rounded off using the filletRadius parameter. Furthermore, a cutoutType node must be specified, the value of which must be one of “window”, “door” or “ramp” in correspondence with the three use cases for structural cutouts in CPACS.

For the augmentation, the door cutouts must be specified explicitly by providing the longitudinal position, height, width and fillet radius. When setting the reference angle to $\pm 90^\circ$, the vertical position of the door can be computed from a given target sill height. For the positioning of the window cutouts, the frame positions are taken into account, as well as to the door positions. A cutout profile and the height of the window above the floor must be given in advance, similarly to the door cutouts. The longitudinal position, however, is set to be the middle between a pair of neighboring frames. Windows are only placed between the first and last door. If a frame bay is occupied by or within a certain tolerance of a door, no window is placed. Given the focus on cabin, cockpit windows are omitted entirely at this stage.

To gain a better understanding of the available space in the cabin, the provision of information on bulkheads is also desirable. The necessary parameters are available in CPACS and have also been described by Scherer and Kohlgrüber [12]. When initializing the structure for the

preliminary design data set, geometric details of the bulkheads can be specified, along with the desired longitudinal position, a corresponding sheet element type and the bulkhead type.

2.2 Cabin definition

Compared to the structure, cabin design activities using CPACS have received little attention in literature [13–16]. Nonetheless, a comprehensive node called decks is provided in its current release 3.2, which largely dates back to Fuchte [15]. Based on a simplified description of the cabin space boundary using contour lines, which are defined by a structured grid of sample points, seats and large cabin floor components such as galleys and lavatories can be placed. Further information e.g. on door positions and evacuation spaces can be provided, as well.

A weakness of the current parameterization is, that it permits inconsistencies, e.g. if changes are made to the outer geometry, the cabin boundary contours become invalid. Furthermore, it contains redundant information, such as door definitions, which overlap with the cutout definitions described in the previous section. Finally, definitions for key cabin components including sidewall panels and hat racks, which are essential when building an immersive virtual cabin mock-up, are missing altogether. Therefore, a modified version of the decks node is proposed at this point and applied in the following.

A key modification is the introduction of the deckElements node. Analogously to the $\text{structuralElements}$ node, which, among other things, contains all the structural profile cross-sections, this node provides definitions of standard cabin components, that are used as building blocks when assembling a cabin layout for a given configuration. Data on seats, sidewall and ceiling panels or luggage compartments can be stored, but also on larger monuments such as galleys and lavatories. In the simplest case, a

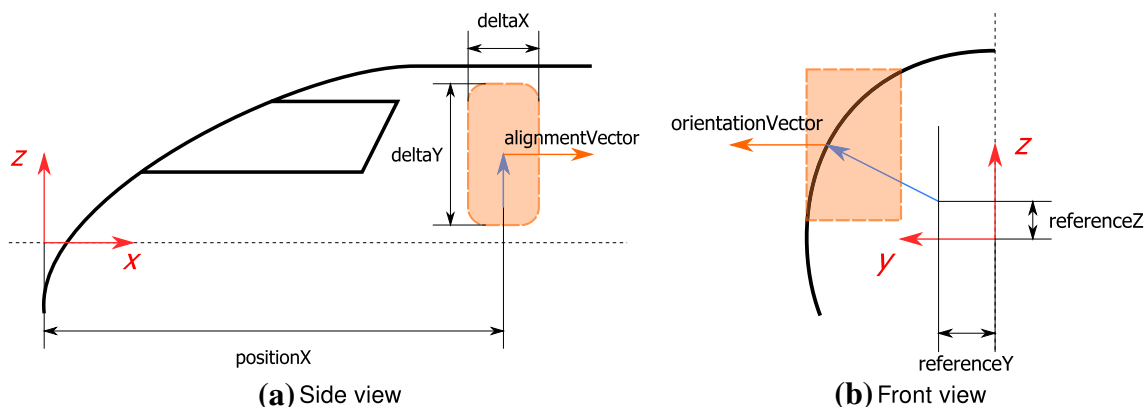


Fig. 1 Schematic depiction of the cutout definition

component is described by its bounding box, i. e. its length, width and height. Apart from that, a component mass for a subsequent mass estimation can be specified, as well as a `genericGeometryComponentUID`. Unique Identifiers (uIDs) can be assigned to most nodes and CPACS, and enable referencing said node from any other node within CPACS. The `genericGeometryComponentUID` refers to a particular type of node in CPACS, that allows the integration of geometry from external sources including CAD models in STEP or IGES format and triangulated 3D geometry in STL format. In this way, dummy geometry for more detailed and immersive visualization can be embedded in CPACS very easily. Depending on the component type, additional metadata can be provided for downstream analysis, such as the number of seats in a seat row, or the number of trolleys in a galley module.

Outsourcing the component definitions results in a much leaner `decks` node, which serves merely to instantiate a cabin assembly in a given fuselage. To this end, four types of entries are necessary:

- Floor-based component placement (seats, galleys, lavatories, class dividers): These components are placed in the local cabin coordinate system, which is given by a global transformation node for each deck, by providing an x and y coordinate using the `transformation2D` type from CPACS. Since the component is to be mounted on the floor, the vertical coordinate of the lower end of the bounding box defaults to $z = 0$. In addition to the coordinates, an element uID referencing an entry in the `deckElements` node must be specified. Furthermore, local offsets and scaling of the model can be manipulated using an optional additional transformation node. UIDs of longitudinal floor beams can be specified to provide information on structural mount points.
- Component placement in 3D space (linings, luggage compartments): The above formulation must be generalized for components, which are not mounted on the floor. Since the x and y coordinates are insufficient, the `transformation2D` node is replaced by a regular `transformation` node, allowing positioning in all dimensions. Consequently, component positions must also be computed in z direction for each component placed during the layout synthesis in addition to x and y positions, e.g. via offsets as described in Sect. 3.3. The transformations are given w. r. t. the local aircraft coordinate system. The longitudinal floor beam uIDs are generalized to parent uIDs, which can refer to an arbitrary structural element.
- Description of free spaces (aisles, evacuation spaces): Knowledge of free spaces in the cabin is essential not only for certification, but also boarding and evacuation simulation [13, 14, 16]. Even though they can in princi-

ple be deduced from the floor-based components, a separate definition is provided to allow for the formulation of required or desired spaces. Like floor-based components, these are defined in the local xy coordinate system of the cabin floor, however, aisles and other generic spaces are described in slightly different ways. The basis of both definitions is a polyline, which is closed to form a polygon and extruded to a 3D body using an additional height parameter for generic spaces. In contrast, the key parameter for the aisles is the width, which is given at each point of the polyline, defining a 2D surface on the floor.

- Door definitions: As mentioned previously, the current release of CPACS (3.2) contains redundant door definitions. In the scope of this publication, the definitions have been modified in such a way, that door geometry can be specified exclusively using cutouts. The alternative description in CPACS using width, height and sill height is no longer valid. However, as outlined above, conversion between the two is feasible.

Another important change is the pending elimination of the cabin geometry (`cabGeometry`) node, which was used to specify an outer boundary for the cabin as described above. Since the points given in this definition can be rendered invalid, even by minor changes to the fuselage geometry, which can quickly result in inconsistent data sets, a long term storage of the geometry points in CPACS is precarious. Instead, an ad hoc generation of the cabin geometry, as described in Sect. 3.3, should be preferred.

The modified cabin description presented here is used in the following to store the results of the cabin synthesis. At the same time, it serves as a guideline for the required range of components and the level of detail for the design process.

3 Implementation of a design and modeling environment for fuselages

To build a digital mock-up, the parameters and definitions given in CPACS must be translated to CAD geometry using a geometric modeling or parameter engine. Due to the multi-fidelity scope of CPACS, it is possible for some pieces of information critical to modeling certain components to be missing. In this case, additional design functionality must be applied to augment the missing data, before the geometry can be generated.

3.1 Knowledge modeling of CPACS data

As illustrated by the uID references in the cabin definition, as described in the previous section, CPACS is a hierarchical format at first glance only. In truth, the possibility to

interlink virtually any two nodes results in a far more complex network of relationships and dependencies. Therefore, correctly modeling the dependencies between given pieces of information in CPACS is essential to ensure a consistent design.

To this end, the CPACS interface originally introduced by Walther et al. [17] has been restructured using a knowledge-based object oriented approach as proposed e.g. by La Rocca [4]. Each CPACS data node is augmented by a unique `xpath`, `requires` and `value` attribute.

The `xpath` attribute contains a generalized notation of the address used for accessing nodes in the XML data set. It also serves as a unique label for the node in the network of CPACS parameters. The generalization is necessary to align nodes, which are semantically identical but differently named, such as `/cpacs/vehicles/aircraft` and `/cpacs/vehicles/rotorcraft`.

The `requires` attribute is usually a set and leverages the `xpath` nomenclature to reference other nodes, which are required for the interpretation of the present node.

Finally, the `value` attribute is used to access the actual data stored in the node. Despite the name, it does not usually contain a singular value but a table of all the entries of the same type beneath a collection node, such as the `frame` entries beneath the collection node `frames`. The information is stored as a pandas data frame [18], a popular structure for tabular data in Python. It enables efficient processing and provides powerful SQL-style merge operations between nodes which is useful down the line during modeling. The tables are read from CPACS and stored in the `value` attribute which is initially empty and remains so if no data are found in the data set. The `value` can only be set through a dedicated `compute` method.

Using the `xpath` and `requires` attributes from all node objects, a directed acyclic graph (DAG) of the dependencies can be constructed. Assuming acyclicity simplifies the handling of the data, but is also justified, since cyclic `uID` references are not found in the CPACS schema. The NetworkX package [19] for Python not only provides a DAG data structure, but also many common graph algorithms. The understanding of dependencies provided by the DAG yields several interesting opportunities. Most importantly, it provides the means, to query the ancestors and descendants to any given node. Any ancestor to a given node represents another node that imperatively must be known to compute the value of the given node. The descendants, on the other hand, are nodes that will not be computable unless the given node is known. Figure 2 provides an example of this for frames. It can be seen, that the `value` attributes of the `fuselage` and the `profileBasedStructuralElements` nodes must be known (i.e. not empty) to correctly interpret and model the information contained in the `frame` node. The `fuselage` node provides the surface loft used to

compute intersection curves with the frame planes, whereas the `profileBasedStructuralElements` node provides the cross-section. In turn, information from the `frame` node is required e.g. to interpret the information on the side-wall panels, which can reference frames via their `parentUID`, as explained in the previous section.

The information stored in the graph can also be used to read nodes just in time as they are required (lazy evaluation), or to schedule parallel execution of independent nodes. Another interesting feature is the identification and handling of missing or empty nodes, which is a common problem given the multi-fidelity nature of CPACS. In the case of the aforementioned `frame` node, if the `value` parameter remains empty after the CPACS readout, it can be deduced, that the CPACS data set does not provide any information on the frames. This means that design rules in the form of Python functions need to be applied to generate the frame information based on the available information. In the case of the frames, this means that a frame distribution must be generated from the known mainframe positions as described by Walther and Ciampa [11].

To this aim, the system is modified by overloading the `frame` node with a dedicated design node, which provides the frames as a function of the main frames. As shown in Fig. 3, this extends the system graph with several auxiliary nodes, which are not stored in CPACS, but essential to the design process nonetheless. An example for such an auxiliary node is the `mainFrames` node. The positions of the main frames are deduced e.g. from the bulkhead and door positions. The profile cross section definitions are copied from a surrogate CPACS file. In this way, the known inputs required to compute the value of the `frame` node can be reduced to only the `fuselage` and the tool-specific `fuselageAugmentation` node, which contains the structural design parameters mentioned in Sect. 2.1. By implementing many knowledge rules, a system is built, which can adapt to a large variety of input states from CPACS and automatically augment missing information accordingly. In the following subsection, the rules for translating relevant structural CPACS parameters from Sect. 2.1 into CAD geometry are described, which in turn provide necessary inputs for the cabin layout synthesis rules outlined in Sect. 3.3.

3.2 Fuselage structure CAD model

As outlined in Sect. 2.2, the CPACS schema provides means for detailed fuselage structure descriptions. The most common use case is to create global finite element models of the structure, consisting of shell elements for the skin panels and beam elements for the stiffeners. In this context, the TiGL Geometry Library [20] can be employed to generate the outer aircraft geometry from CPACS, which is sampled using intersection vectors to build a discretized geometry

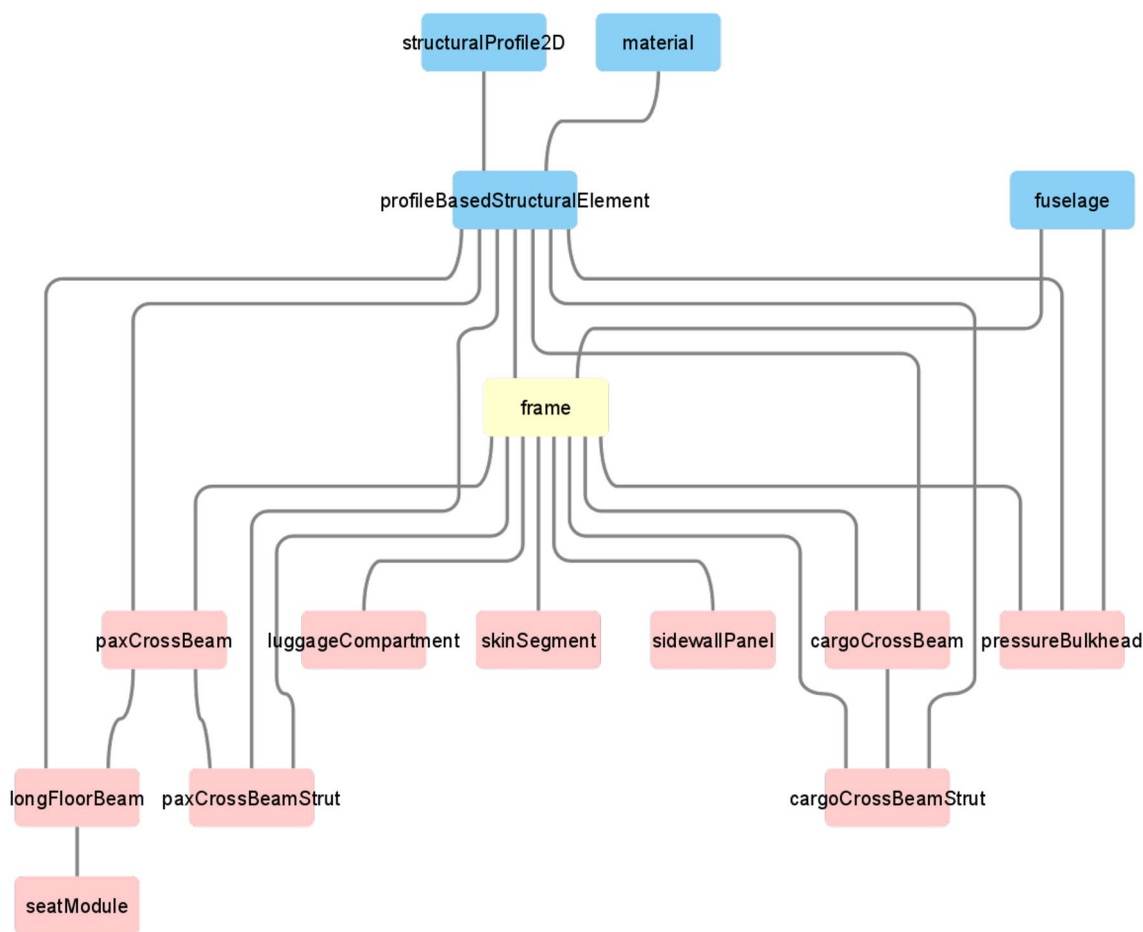


Fig. 2 Graphical dependency map for the frame node in CPACS

model. The cross-section of the profile-based elements is primarily used to compute the properties of the beam elements, although some finite element analysis environments like ANSYS also support the visualization of the extruded beams, if provided with suitable profile definitions.

Nonetheless, for both the purposes of virtualization, and to generate a cabin layout that is consistent with the structure, it is necessary to provide geometry not just in a discretized form, but on the basis of smooth, mathematically accurate surfaces. The CAD geometry can be discretized as required for a given use case and also provides surface normal information which is essential for smooth shading even of coarse meshes.

Many of the structural features necessary for the cabin design are not provided by the TiGL library in this level of detail and need to be constructed directly from the CPACS data. This can be accomplished using the Open Cascade Technology (OCCT) library [21], an open-source CAD kernel, which is also the basis of the TiGL library. In particular, detailed information on the geometry of the structural reinforcements, such as frames, is necessary for the cabin layout

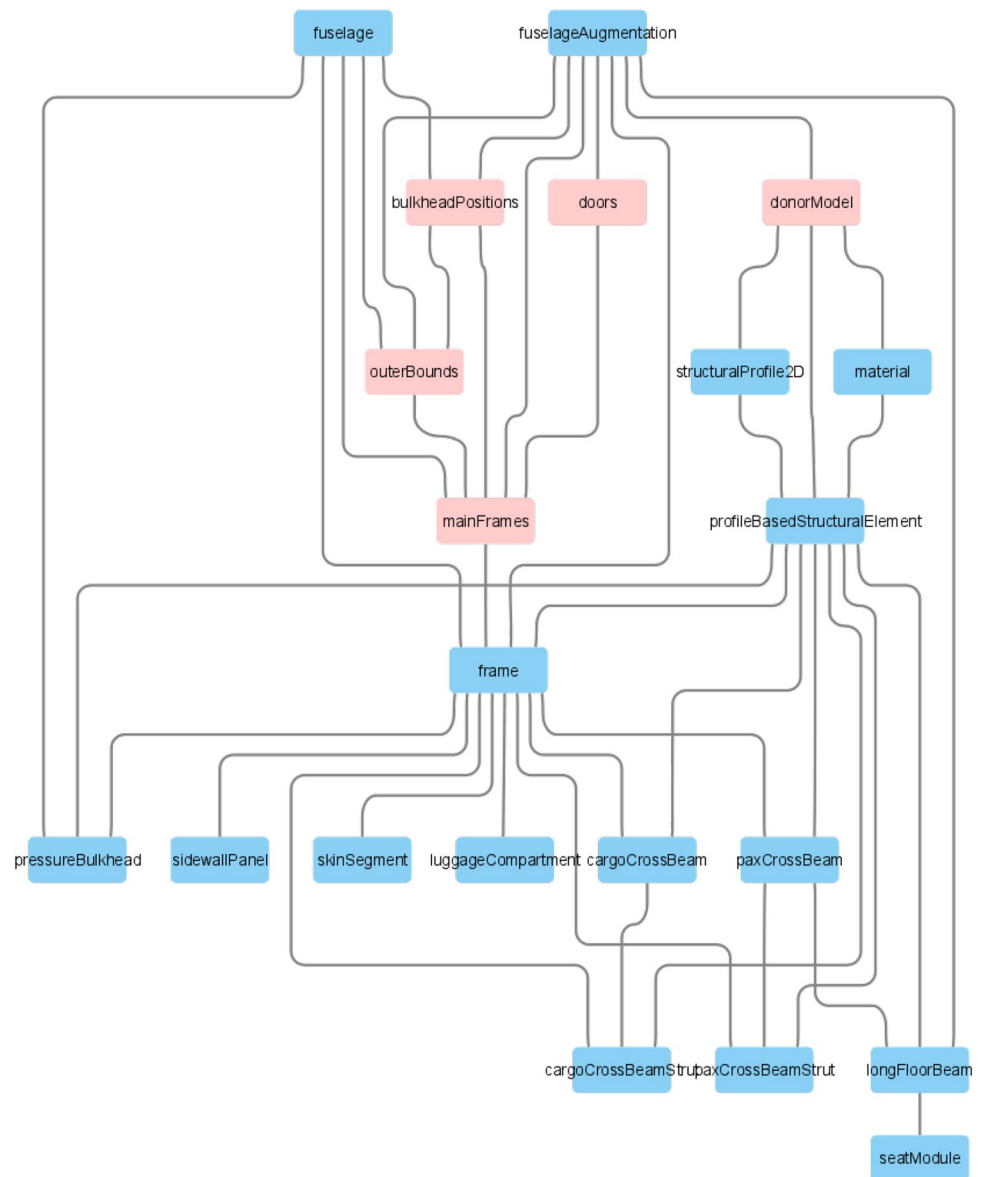
generation. Therefore, the profile-based structural elements from CPACS must be turned into CAD solid models by extruding their profile along their definition curve.

Taking the frames as an example, the definition curve is specified by the intersection of the fuselage surface and a frame definition plane. The fuselage surface from CPACS can be generated using the TiGL library or by interpolating the cross-section profile points as shown by Walther et al. [17]. The frame definition plane is usually orthogonal to the longitudinal axis of the fuselage and can thus be derived from the longitudinal position given in CPACS. The intersection curve—typically a B-spline curve—is computed using OCCT. Along this trajectory curve $\mathbf{T}(v)$, the structural profile polygon, reformulated as a first degree B-spline curves $\mathbf{C}(u)$, can be extruded as a swept surface as described by Piegl and Tiller [22] using

$$\mathbf{S}(u, v) = \mathbf{T}(v) + \mathbf{A}(v)\mathbf{S}(v)\mathbf{C}(u), \quad (1)$$

where $\mathbf{A}(v)$ is a transformation matrix and $\mathbf{S}(v)$ is a scaling matrix. Since no scaling is performed on the profiles, the

Fig. 3 Expansion of the CPACS graph with frame design rules



corresponding matrix turns to $\mathbf{S}(v) = \mathbf{I}$. The transformation matrix at a given position v can be computed based on the tangent vector of the trajectory $\dot{\mathbf{T}}(v)$ and the normal vectors of the fuselage surface, which span a local coordinate system. The tangent of a B-spline curve is simply computed by deriving the basis functions. For computing the normals, on the other hand, the trajectory curve $\mathbf{T}(v)$ must be projected onto the fuselage surface, which yields a compatible curve $\mathbf{T}_{uv}(v)$ in the surface's parameter space (commonly referred to as p-curve). This is illustrated by Fig. 4b for the case of a conical segment and a skew plane shown in Fig. 4a.

A canonical solution for constructing a swept surface is to apply the skinned surface algorithm on a finite number n_v of transformed profile sections at the positions \mathbf{v} [22]. Since the trajectory curve $\mathbf{T}(v)$ and its projection $\mathbf{T}_{uv}(v)$ are compatible, the latter can be evaluated at \mathbf{v} to compute

the surface uv coordinates corresponding to the Cartesian points $\mathbf{T}(\mathbf{v})$. At these coordinates, the surface normals can then be evaluated. In some cases, such as the frames, the normal vectors must be manipulated, e.g. by projecting them onto the curve definition plane. Due to the way frames are defined in CPACS, this usually causes the x component to turn to zero. CPACS imposes a convention with respect to the sense (mathematically positive) and ordering (front to back) of the fuselage profiles, which is enforced during readout and propagated to the lofted surfaces. Consequently, it can be asserted that the normals will always be facing inward.

Aside from the transformations $\mathbf{A}(\mathbf{v})$, the global translation prescribed by the trajectory curve point $\mathbf{T}(\mathbf{v})$ must be applied to the profile. Due to the affine invariance property of B-splines, any transformation of the curve corresponds to

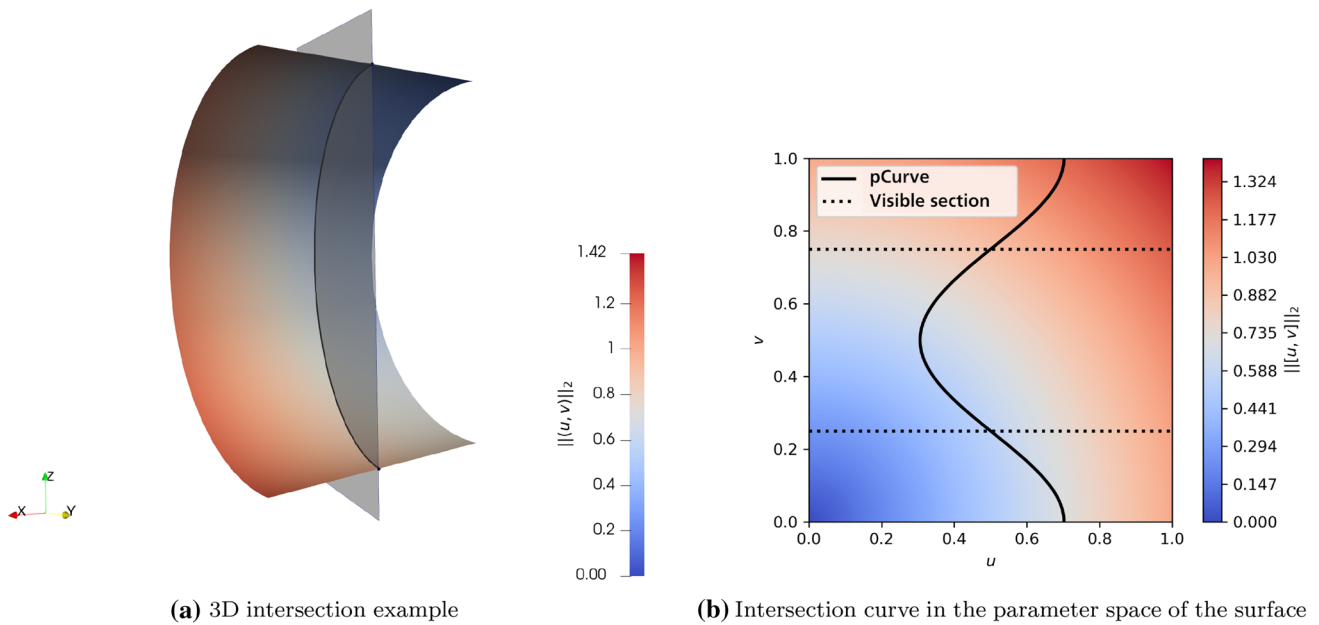


Fig. 4 Frame curve determination for a conic fuselage segment and a skew definition plane

a transformation of the control points, which can be implemented very efficiently. The transformed profile sections for the example of the conic section are given in Fig. 5a.

For the skinned surface algorithm [22], the transformed control point arrays of the sections are collected in a $n_c \times n_v \times n_{dim}$ array, where n_c is the number of profile control points and n_{dim} is the dimensionality of the points. In addition, the B-spline basis functions of the trajectory curve $N_T(\mathbf{v})$ are evaluated at the sample positions \mathbf{v} , which results in a $n_v \times n_t$

matrix, where n_t is the number of control points of the trajectory curve. By choosing the number of sample points to be $n_v = n_t$, $N_T(\mathbf{v})$ becomes a square matrix and thus, the series of linear systems

$$\mathbf{P}_i = \mathbf{N}_T(\mathbf{v}) \cdot \mathbf{P}_{S,i}, \{i \in \mathbb{N} : 1 \leq i \leq n_c\} \tag{2}$$

can be solved and assembled to yield the control point array \mathbf{P}_S of the extruded B-spline surface $S(u, v)$. The knot vectors

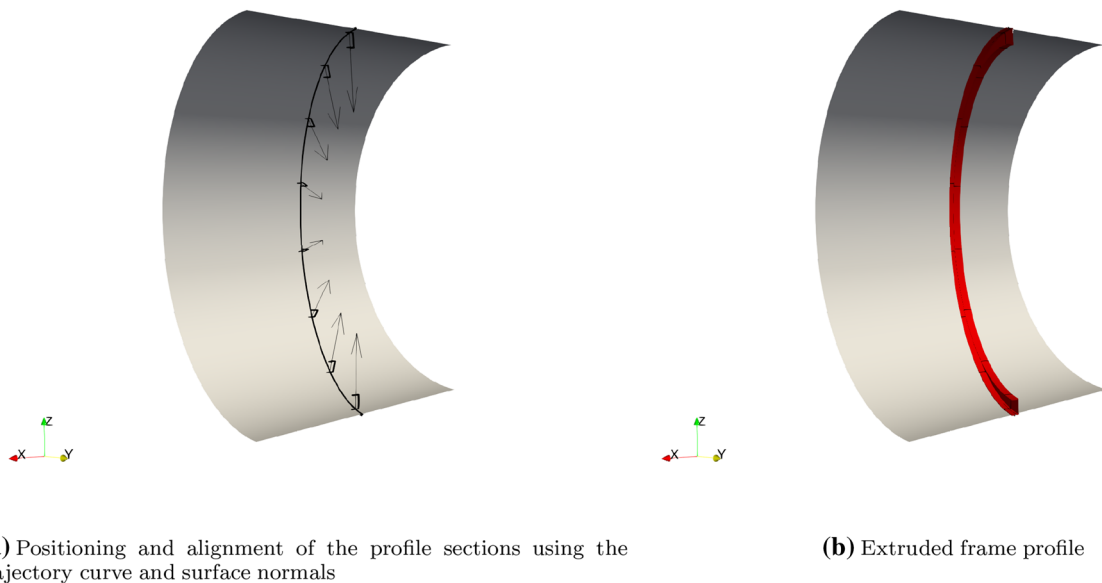


Fig. 5 Profile extrusion using the swept surface algorithm

and polynomial degrees can be carried over from the section curve $C(u)$ and the trajectory curve $T(v)$. For the example of the frame, this results in the surface shown in Fig. 5b.

The swept surface algorithm is also applied for modeling the floor beams elements. However, the trajectory curves are constructed in a different way, which essentially corresponds to the procedure applied during finite element model generation. Using a point and a vector specifying a direction, an infinite or semi-infinite line can be constructed, which is then clipped to yield only the segment that lies within the fuselage. For longitudinal floor beams, which are defined using a multitude of points, a polyline is constructed instead. The profile transformation before extrusion is determined by the type of component: The profiles of the floor grids and the strut profiles are oriented in vertical and longitudinal direction, respectively.

To achieve the correct geometry, profiles should initially be extruded beyond the bounds of the fuselage, before computing the intersection of the fuselage volume and the resulting solid. This operation, however, is very computationally expensive, as two complex geometries must be intersected for each floor element. Instead, clipping the trajectory curve first and performing the extrusion after, as described above, reduces the computational cost significantly, while still returning acceptable results.

Another important modeling function is the creation of door and window cutouts. Based on the cutout-definition, solid bodies are extruded, which are then subtracted from the fuselage surface and the structural elements in a Boolean operation sense. In this context, it is important, to extend the extrusion also towards the inside of the fuselage to account for the curvature of the fuselage surface.

This concludes the CAD modeling rules for the structural components relevant to the cabin layout synthesis. However, to integrate the structural model into the virtual mock-up, the CAD surfaces can be converted to triangle meshes and exported to a suitable format to transfer the geometry to other frameworks more specifically dedicated to 3D visualization. To this end, the OCCT library provides an exporter for the Stereolithography (STL) format, which, however, does not support e.g. export of surface normals. Therefore, the OCCT mesher is used directly to compute a triangulation of a given surface instead. It implements Delaunay triangulation and provides a few mesh control parameters, such as minimum edge length and angular deflection limits. The triangulation not only returns the Cartesian point coordinates and triangular cell connectivity, but also the corresponding surface uv coordinates. This makes it possible to compute the corresponding surface normal vectors in a postprocessing step.

The model components are usually stored in various types of boundary representation (BREP) objects, such as solid bodies and composed surfaces (shells). The mesher,

in contrast, works only on the underlying individual faces, which means it might be necessary to manually iterate the BREP object tree during mesh generation.

For further processing the mesh, including all additional point and cell metadata is converted to data types of the Visualization Toolkit (VTK) library [23]. If required, the hierarchy of the BREP can be retained using multi-block data sets. The library not only provides a wide array of filters, e.g. for mesh decimation, but also interfaces to many common 3D mesh formats.

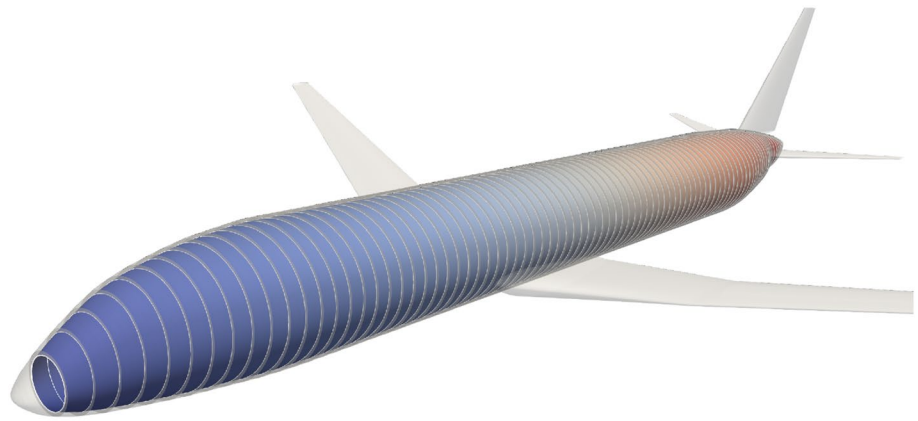
3.3 Cabin layout synthesis

The now known structural layout and geometry provide the basis for the cabin layout synthesis. As a first step, it is important to identify the interfaces between structure and cabin. Most importantly, the structure provides the boundary to the space available for designing the cabin, which is constrained by the skin, frame height, the floor structure, the bulkhead positions and the cabin wall position. As mentioned in Sect. 2.2, CPACS includes a node, which provides a definition for this space. However, no mechanism is in place to enforce consistency with the structure. Therefore, the cabin space is instead constructed ad hoc from the structural definitions. To accomplish this, different algorithms can be applied, depending on the information available for the cabin synthesis.

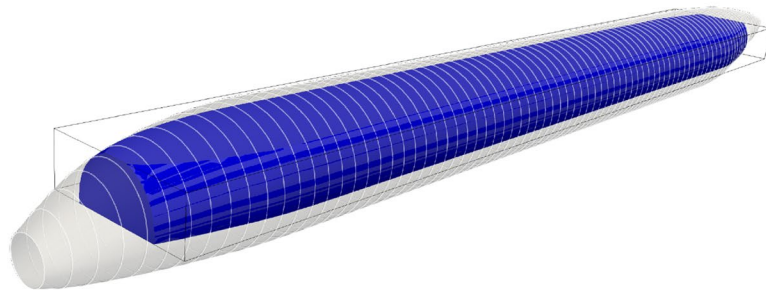
In case no details on frame geometry are provided, a generic frame height can be assumed. Using an OCCT function, an offset surface can be constructed from the fuselage surface based on this information. However, this algorithm should only be applied if no further information is available, since operations on the resulting offset surfaces tend to be computationally expensive.

If the details on the frame geometry are known, the cabin space can instead be constructed directly from the structural definitions. This is accomplished by once more extruding the frames as swept surfaces. Instead of a detailed profile, a simple line representing the frame height is used as an approximation. The inner curve of the frame can be extracted by computing the isoparameter line for the surface parameter value in section direction $u = 1$ for the resulting surface. From the sequence of curves, a surface as illustrated by Fig. 6a can be constructed, once more using the skinned surface algorithm applied in the previous section, albeit under slightly different conditions. Unlike the transformed profiles, the frame tip curves are not necessarily compatible, i. e. they do not share the same order or knot vector. Therefore, they must be made compatible in a preprocessing step using degree elevation and knot insertion algorithms [22]. As a result of the knot insertion, all curves share the same parameter vector, which is the union of all distinct parameter values and multiplicities of the curves. Due to

Fig. 6 Ad hoc cabin space construction



(a) Skinned surface construction from frame heights



(b) Determination of cabin space

numerical inaccuracies, two very close parameter values might be considered to be distinct, resulting in large and computationally expensive knot vectors. By introducing a liberal tolerance of $\varepsilon = 1e - 4$, this issue can be alleviated and surfaces of high quality can be generated, which can also be handled well numerically. Since no trajectory curve is given, the basis functions $\mathbf{N}_T(\mathbf{v})$ must be computed using an interpolation technique [24]. The choice of parameter vector has no impact, since a linear interpolation scheme is applied.

Commonly, the outcome of the interpolation algorithm is a surface, which is closed and periodic in circumferential direction. It can be turned into a BREP solid by closing up the profiles at both ends, enabling more reliable Boolean operations. Using the known floor height and rear bulkhead position, as well as the cockpit wall position, which coincides with the cabin origin, a box can be constructed, which represents further cabin bounds. Intersecting this box and the BREP solid from the frames yields the cabin space as shown in Fig. 6b with a much higher accuracy than the original

CPACS definitions. For backward compatibility, the surface can be sampled using a regular grid of intersection vectors to compute the points needed for the legacy `cabGeometry` definition. For all subsequent steps, the most important feature of the cabin geometry is its base area, which provides the boundary for the placement of seats and floor elements such as lavatories or galleys. On the other hand, it is also a useful reference for the initial placement of paneling and luggage compartments. Aside from the cabin space, information on the available cabin elements as provided by the respective CPACS node introduced in Sect. 2.2 is required, possibly including a reference to an external 3D component file, or at least a bounding box.

Seat elements are placed according to different seating classes, e.g. first, business or economy. Classes are not only distinguished by different numbers of seats abreast and seat pitch, but also potentially different seat models. By taking into account the number of rows in each class, the number of passengers per class and hence the total

number of passengers can be computed. Based on the latter, the number and type of exits is determined based on the regulations found in the certification specification CS-25 [25]. The locations of the exits link back to the structural cutout definitions on the one hand and to the necessary seat spacing on the other.

Furthermore, galleys and lavatories must be placed at an exit using a 3D notation composed of an exit ID, a Boolean indicator of whether the part is placed before or after the exit and the placement along the width. Component counts are determined based on the required level of comfort.

Combining the information on class seat pitch, seat dimension exit width and floor element placement into a seat gap Δx_i , the seat row positions can be computed recursively using

$$x_{i+1} = x_i + \Delta x_i + l_i. \quad (3)$$

The position in y direction can be deduced using an inside-out approach based on the aisle and row widths, or an outside-in approach using the floor boundary instead of the aisle width. Irrespective of the selected approach, a consistency check is necessary, to assert that neither the floor width, nor the minimum aisle width are violated. Similarly, the x positions must be validated against the length of the floor boundary.

For the placement of the sidewall panels, luggage compartments and ceiling panels, the frame distribution is taken into account in addition to the floor boundary. Due to the dummy models used, it is assumed, that a sidewall panel will span two frame bays. Therefore, longitudinal scaling factors can be computed using the known frame positions. This is trivial in the cylindrical section of the fuselage, but in the non-cylindrical parts, the curvature of the floor bound must be taken into account. Here, the installation angle of the sidewall panel and luggage compartment is computed using

$$\gamma_i = \arctan \left(\frac{\Delta y_{\text{bound},i}^0}{\Delta x_{\text{frame},i}} \right), i \in \{1, 3, 5, \dots\}, \quad (4)$$

where

$$\Delta x_{\text{frame},i} = x_{\text{frame},i+2} - x_{\text{frame},i} \quad (5)$$

$$\Delta y_{\text{bound},i}^0 = y_{\text{bound},i+2}^0 - y_{\text{bound},i}^0 \quad (6)$$

with

$$y_{\text{bound},i}^0 := y_{\text{bound}}(x_{\text{frame},i}, 0). \quad (7)$$

$y_{\text{bound}}(x, z)$ is the cabin boundary function. The required part length results to

$$l_{\text{req},i} = \left\| \left[\Delta x_{\text{frame},i} \quad \Delta y_{\text{bound},i}^0 \right]^T \right\|_2, \quad (8)$$

which yields a scaling factor $s_{x,i,\text{part}} = \frac{l_{\text{req},i}}{l_{\text{part}}}$.

The placement of the sidewall panels along the floor boundary is also clearly determined in the cylindrical sections. The position and rotation angle are given by the line between the boundary points at the frame positions $x_{\text{frame},i}$ and $x_{\text{frame},i+2}$. For the non-cylindrical sections, however, different strategies may be applied, as illustrated in Fig. 7. Placing the panels side by side without gaps will result in overlapping panels due to the concave outline. This can be resolved by moving the panels apart along the boundary line at the cost of introducing gaps, which might also be undesirable. A third option is a centered positioning, which is a compromise between the two previous extremes.

This approach is also valid for the luggage compartments. However, whereas the sidewall panels are placed directly on the border of the cabin space, an offset in y and z direction is applied to the luggage compartment, based on the sidewall dimensions and an overlap parameter, which accounts for design details of the parts, that are not reflected by the bounding box. Determining the correct overlaps currently

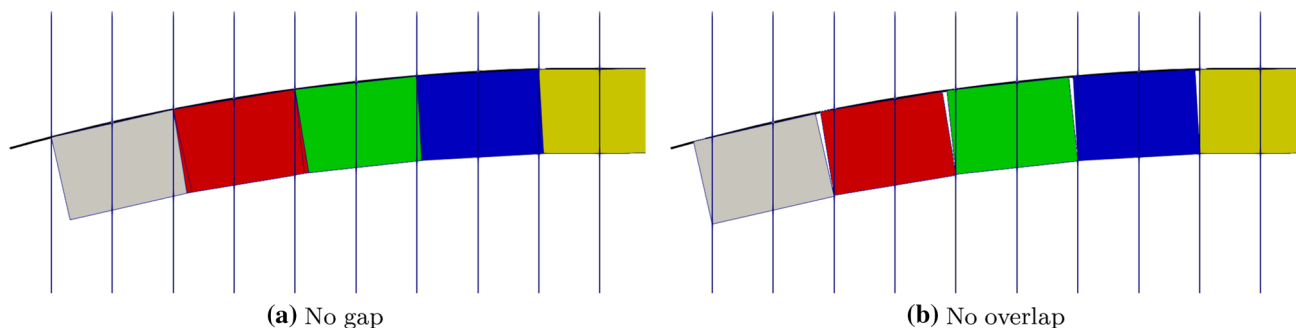


Fig. 7 Positioning strategies for non-cylindrical sections

involves a manual process of comparing and aligning the dummy geometry models. Finally, the computed positions are also compared to the exit layout to discard panels which collide with the exits.

The ceiling panels are not rotated and instead scaled assuming $l_{\text{req},i} = \Delta x_{\text{frame},i}$. In exchange, it might be necessary to also scale the ceiling panels in the width direction. This depends on the dimensions and positions of the side-wall panels and luggage compartments, as well as the width of the cabin space boundary and the number of aisles. For multiple aisles, additional luggage compartments may also be included following the same strategy.

4 Virtual mock-up of the fuselage and cabin for the AVACON research baseline

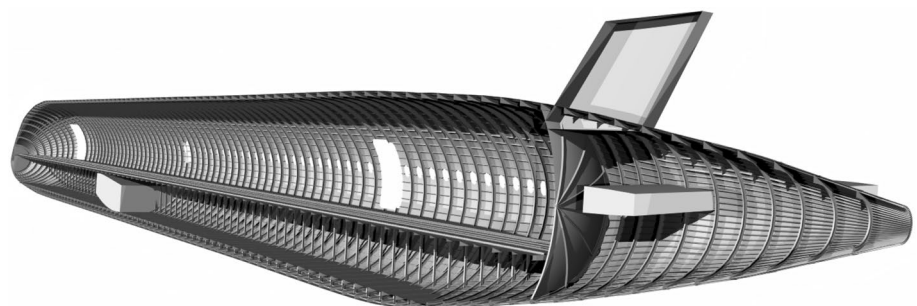
Finally, the methods outlined above are demonstrated in an application case and different visualizations are derived. As an example, the CPACS data set for the AVACON Research Baseline (ARB) [26] is used, a concept for a mid-range aircraft with entry into service in 2028. At a range of 4600nm and a capacity of approximately 250 passengers, it is designed to fill the gap in the civil aviation fleet between large short range aircraft like the Airbus A321 and small long-range aircraft like the Airbus A330, which is often referred to as the *middle of the market*. The top level aircraft requirements are derived from the Boeing 767.

A cabin layout is already included in the data set [16], which is used as reference. Following the example of the Boeing 767, a twin-aisle two-class layout with four seats abreast in business class and seven seats abreast in economy class has been selected.

4.1 Structural design

Since no information on the fuselage structure is given in the CPACS file for the ARB, it needs to be generated first. Attention must be paid, that the floor height given for the structural augmentation matches the position of the cabin, which is already given in the data set.

Fig. 8 Rendering of the ARB fuselage structure



A rendering of the fully modeled fuselage structure is given in Fig. 8. The advantages of the CAD-based model generation and the provision of surface normals are highlighted by the smooth shading e.g. of the spherical rear pressure bulkhead. The cutouts are also easily discernible. However, it is also clear, that some structural components, which are not immediately relevant to the cabin design, such as the landing gear bay or the tail plane attachments are missing in the model.

4.2 Cabin design

Based on the structural layout, a cabin layout is synthesized following the rules in section 3.3. Figure 9 shows both a newly generated single-aisle layout and the twin-aisle reference layout. Inspired by the Airbus A321 cabin layout, the

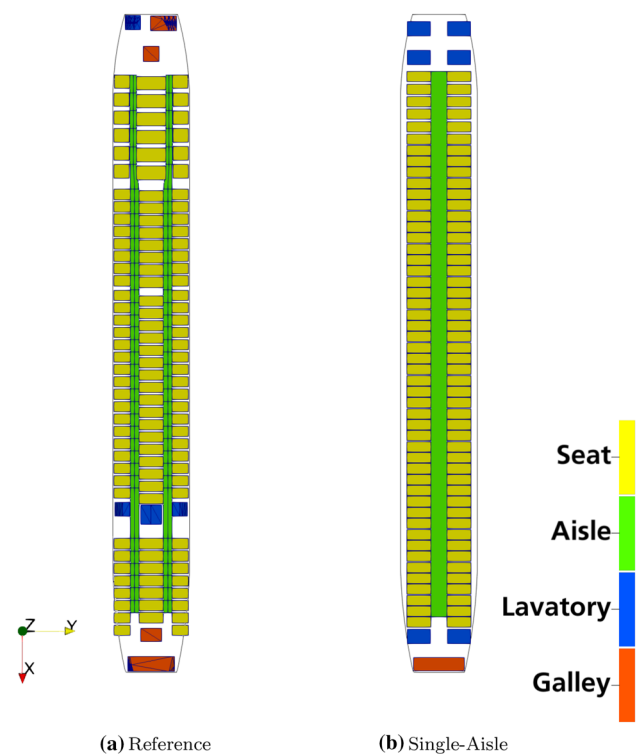


Fig. 9 ARB cabin layouts

single-aisle configuration consists of two classes with four seats abreast in business and six seats abreast in economy class and a seat pitch of 30in across all classes. For a fictitious high-speed and high comfort boarding scenario, the aisle width has also been increased to 30in. This layout seats 242 passengers, while the comfort level due to number of available lavatories and galleys is kept at a level similar to the reference. The exit positions from the structural model are respected by matching spaces. While the occupation of

the available space is far from excellent for this layout, it still showcases the capacities of the design tool at this stage.

Figure 10 shows the panel and overhead storage compartment distribution for the single aisle configuration. The frame distribution is respected, as are the exit positions, where conflicting hat rack and sidewall panels are removed. This leaves a gap in the panel distribution, as matching sidewall panels for doors or emergency exits are not yet available within the process.

4.3 Visualization and virtualization

At this point, the information provided on cabin and fuselage can be fused for combined visualization. To this end, the structural BREP geometry is triangulated and the cabin mock-up assembled to yield two separate, but compatible VTK mesh instances, which can be exported. In Fig. 11, a visualization of the reference cabin of a native VTK polygonal data export in Paraview is shown inside the structural. This fast interface is valuable for quick visualization during development.

The Wavefront OBJ format can instead be used to export the models to Blender e.g. for higher quality renderings as given in Fig. 12. As demonstrated here, the hierarchical structure of the meshes enables selective viewing, e.g. hiding the ceiling panels to expose the underlying structure. At the same time, Wavefront OBJ files can also be used as an interface to the Unity platform [27], bridging the gap from visualization to virtualization. With very little additional effort, the model can be prepared for interactive exploration from a first person perspective, even using VR hardware. An impression from a demonstrator scene for the ARB is given in Fig. 13.

This example illustrates that the methods proposed above can be applied to automatically design and build detailed and consistent cabin models, which are ready to be used

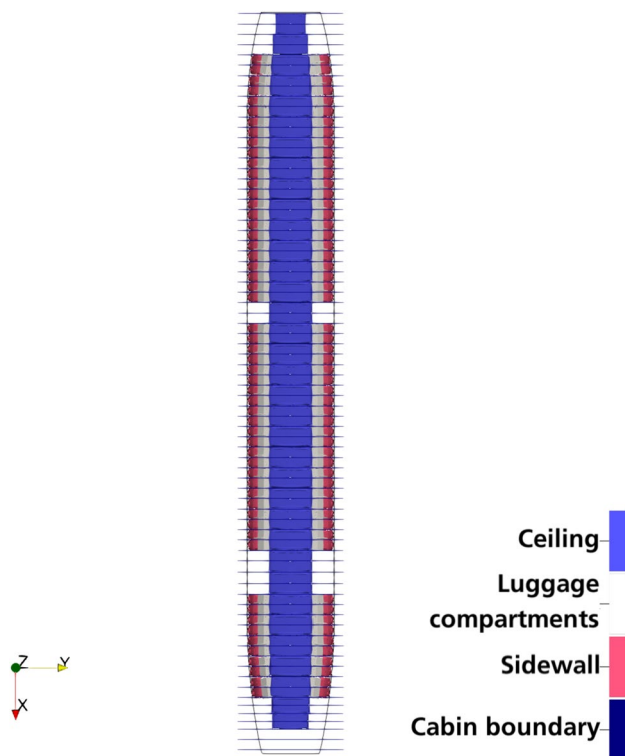


Fig. 10 Distribution of panels and luggage compartments

Fig. 11 Fuselage cabin and structure model (Paraview)

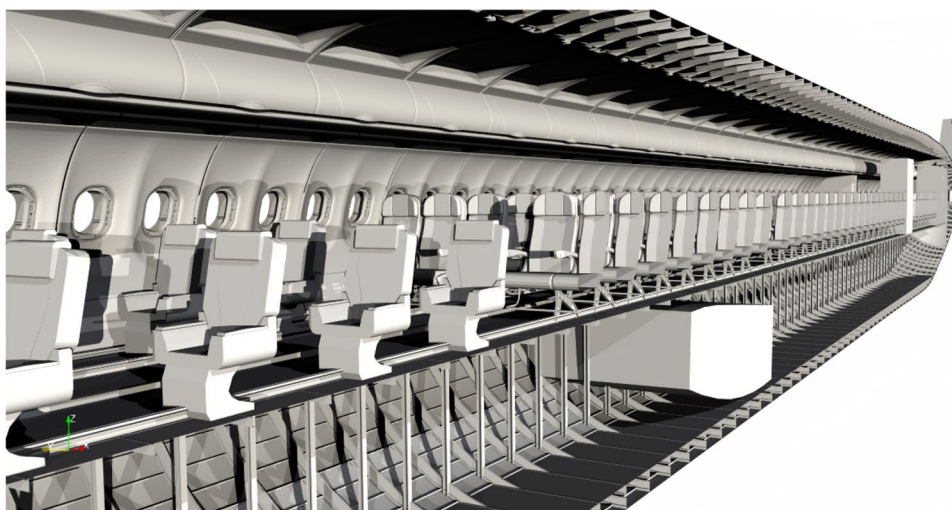


Fig. 12 Reference cabin rendering with visible structure (Blender)

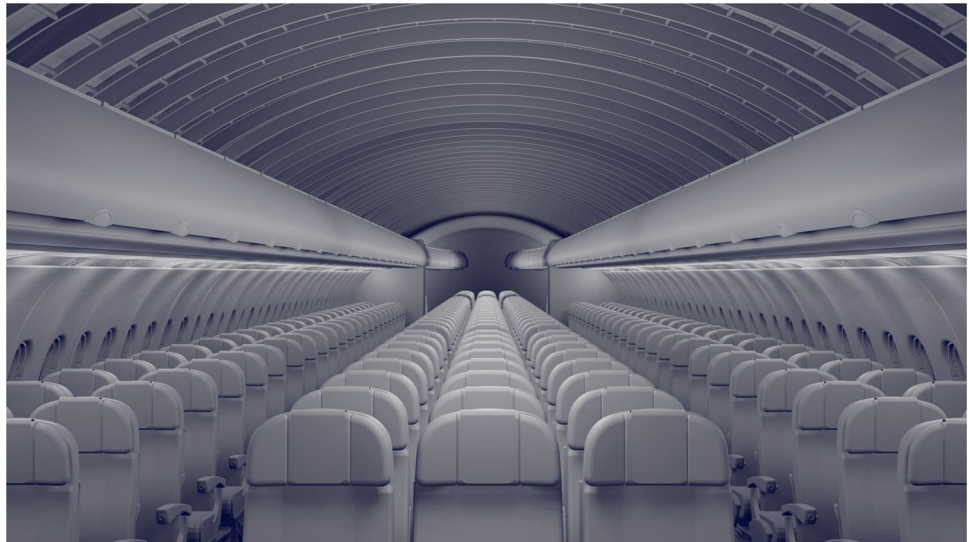
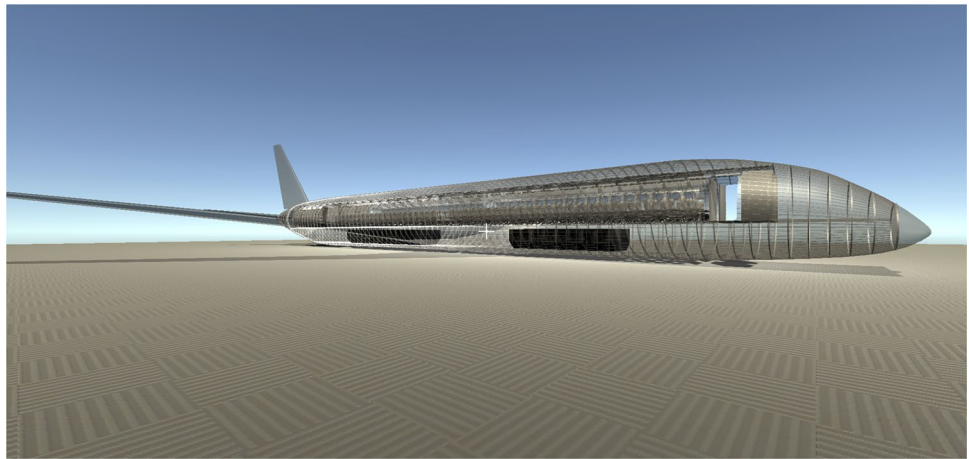


Fig. 13 ARB demonstrator scene (Unity)



in VR applications, based on preliminary data provided in the CPACS format and a very limited number of additional inputs. Due to the high degree of automation and the application of a common central data model like CPACS, this enables the incorporation of detailed cabin design and analysis in automated collaborative aircraft design chains. Furthermore, feedback loops with preliminary design are enabled e.g. to accurately predict the cabin length, which has an effect e.g. on the outer fuselage shape.

5 Conclusion

In the preceding sections, an automated approach for deriving detailed and consistent cabin designs, and VR-compatible 3D models thereof, from preliminary design data provided in the CPACS format has been demonstrated. Moving along a cohesive digital thread, the CPACS data are processed in a knowledge-based design system capable of

augmenting missing cabin and structural information based on a reduced set of control parameters. In addition, a CAD model of the fuselage is generated based on the CPACS parameters using the OCCT library and converted to a triangulated surface mesh for visualization. The structural model is then merged with a cabin assembly, which can process detailed component models provided in addition to CPACS.

The methods have then been demonstrated using the AVACON Research Baseline as an example. Visualizations of the resulting model using various environments including an interactive scene built using the Unity VR platform have been provided.

However, the example also highlights one of the key limitations of the design process at this stage, which is the reliance on 3D dummy models for the design. Even though an impressive level of detail for the visualization can be achieved in this way, such “dead” geometry can only be treated based on its bounding box during the design. In addition, missing component models like the panels for the door areas

can lead to gaps in the model, which is detrimental to the immersion. Therefore, a parametric description schema for cabin elements, similar to CPACS for the fuselage structure, will be essential for future developments. Such a description could not only contain parameters on the geometry of the part but also explicit connection points e.g. to structural components and thus provide valuable additional information for the positioning. This level of detail certainly exceeds the scope of a format like CPACS, which remains aimed primarily at preliminary design activities. At the same time, it presents an opportunity for the development of complementary formats, where new accomplishments in information technology, such as semantic web technologies, can be explored and integrated.

Improved interactivity of the VR application is another interesting trajectory for future developments. Currently, the user can explore the model by walking through it. But given the capabilities of engines like Unity, much more advanced user interactions, such as querying and visualizing metadata or dependencies from within the model or interacting with individual components, are possible. In fact, many of these features have already been demonstrated by [2].

Finally, given the goal of the DLR to become a virtual OEM capable of mirroring entire development cycles digitally, the mock-up introduced in this paper along with its parameterization is well suited to take the role of a virtual product. The geometry can be made available to other critical disciplines, e.g. noise analysis or crash and ditching simulation. Since a detailed finite element mesh is needed, the requirements for which differ substantially from those for the triangle meshes for visualization presented here, it will be necessary to integrate an external mesher such as Gmsh [28] or MeshGems [29]. However, this effort might open up promising new research trajectories, as feeding back the disciplinary analysis results to the virtual product will enable a comprehensive and integrated evaluation and optimization of the cabin within the virtual development cycle.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. De Crescenzo, F., Bagassi, S., Starita, F.: Preliminary user centred evaluation of regional aircraft cabin interiors in virtual reality. In: Scientific Reports 11.1 (May 2021). <https://doi.org/10.1038/s41598-021-89098-3>
2. Fuchs, M.K., Beckert, F., Biedermann, J., Nagel, B.: Experience of conceptual designs and system interactions for the aircraft cabin in virtual reality. In: AIAA AVIATION 2021 FORUM. Am. Inst. Aeronaut. Astronaut. (2021) <https://doi.org/10.2514/6.2021-2773>
3. Alder, M., Moerland, E., Jepsen, J., Nagel, B.: Recent advances in establishing a common language for aircraft design with CPACS. In: Aerospace Europe Conference-AEC2020, Bordeaux, France (2020)
4. La Rocca, G.: Knowledge based engineering techniques to support aircraft design and optimization. PhD thesis. Delft University of Technology (2011) **ISBN: 9789090260693**
5. PACE GmbH. Pcelab ACE myCabin (2021) <https://pace.txtgroupe.com/products/productconfiguration/mycabin/>
6. Gobbin, A.: Numerische Modellierung des Auslegungsprozesses für Passagierkabinen von Verkehrsflugzeugen unter Berücksichtigung der wichtigsten Auslegungsforderungen und Implementierung in MatLab. MA thesis. TU Berlin (2015)
7. Bundesministerium für Wirtschaft und Energie (BMWi). Die Luftfahrtstrategie der Bundesregierung (2014)
8. Helmholtz-Gemeinschaft. Das Programm Luftfahrt (2020). https://www.helmholtz.de/forschung/luftfahrt_raumfahrt_und_verkehr/luftfahrt/
9. Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR). Programm und Strategie: Luftfahrtforschung im DLR (2020). <https://www.dlr.de/content/de/artikel/luftfahrt/luftfahrtforschung/programm-und-strategie-luftfahrtforschung-im-dlr.html>
10. Wöhler, S., Atanasov, G., Silberhorn, D., Fröhler, B., Zill, T.: Preliminary aircraft design within a multidisciplinary and multifidelity design environment. In: Aerospace Europe Conference 2020 (2020) <https://elib.dlr.de/135245/>
11. Walther, J.-N., Ciampa, P.D.: Knowledge-based automatic Airframe Design using CPACS. Transp. Res. Proc. **29**, 427–439 (2018). <https://doi.org/10.1016/j.trpro.2018.02.038>
12. Scherer, J., Kohlgrüber, D.: Fuselage structures within the CPACS data format. Aircr. Eng. Aerosp. Technol. **88.2**, 294–302 (2016). <https://doi.org/10.1108/aeat-02-2015-0056>
13. Fuchte, J.C., Dzikus, N., Gollnick, V.: Cabin design for minimum boarding time. In: Deutscher Luft- und Raumfahrtkongress 2011 (2011). <https://elib.dlr.de/77237/>
14. Fuchte, J. C., Gollnick, V., Nagel, B.: Integrated tool for cabin and fuselage modeling in future aircraft research. In: Workshop on Aircraft System Technology (AST) (2013). <https://elib.dlr.de/82767/>
15. Fuchte, J.C.: Enhancement of aircraft cabin design guidelines with special consideration of aircraft turnaround and short range operations. Tech. rep. Technische Universität Hamburg-Harburg (2014). <https://elib.dlr.de/89599/>
16. Engelmann, M., Hornung, M.: Boarding process assessment of the AVACON research baseline aircraft.en. In: Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V. (2019). <https://doi.org/10.25967/490049>.
17. Walther, J.-N., Petsch, M., Kohlgrüber, D.: Modeling of CPACS-based fuselage structures using Python. In: Aircraft Engineering and Aerospace Technology (2017). <https://doi.org/10.1108/AEAT-01-2017-0028>
18. McKinney, W.: Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science

- Conference. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. <https://doi.org/10.25080/Majora-92bf1922-00a>
19. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th Python in Science Conference. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, pp. 11–15 (2008)
 20. Siggel, M., Kleinert, J., Stollenwerk, T., Maierl, R.: TiGL: an open source computational geometry library for parametric aircraft design. *Math. Comput. Sci.* **13**(3), 367–389 (2019). <https://doi.org/10.1007/s11786-019-00401-y>
 21. Open Cascade SAS. Open Cascade Website. 2020. www.opencascade.com
 22. Piegl, L., Tiller, W.: *The NURBS Book*. Springer-Verlag GmbH (1996). **ISBN: 3540615458**
 23. Schroeder, W., Martin, K., Lorensen, B.: *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, 4th edn. Kitware, Clifton Park (2006)
 24. Park, H.: Choosing nodes and knots in closed B-spline curve interpolation to point data. *Comput. Aided Design* **33**, 967–974 (2001)
 25. European Aviation Safety Agency (EASA). CS-25: Certification Specifications for Large Aeroplanes (2008)
 26. Wöhler, S., Hartmann, J., Prenzel, E., Kwik, H.: Preliminary aircraft design for a midrange reference aircraft taking advanced technologies into account as part of the AVACON project for an entry into service in 2028. In: Deutscher Luft- und Raumfahrt-kongress 2018 (2018). <https://elib.dlr.de/126000/>
 27. Unity Technologies. Unity—Manual: Unity User Manual (2019.4 LTS) (2021). <https://docs.unity3d.com/Manual/index.html>
 28. Geuzaine, C., Remacle, J.-F.: Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numer. Method. Eng.* **79**, 1309–1331 (2009). <https://doi.org/10.1002/nme.2579>
 29. Distene SAS. Distene’s MeshGems suite | Meshing Software Components for CAD and CAE applications from Distene (2020). <http://www.meshgems.com>

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.