**QUEEN'S
UNIVERSITY
BELFAST**

Automated mesh deformation for CAD models which exhibit boundary topology changes

Sun, L., Yao, W., Robinson, T. T., Armstrong, C., & Marques, S. (2020). Automated mesh deformation for CAD models which exhibit boundary topology changes. *AIAA Journal*. <https://doi.org/10.2514/1.J058760>

Published in:
AIAA Journal

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
[Link to publication record in Queen's University Belfast Research Portal](#)

Publisher rights

© 2020 American Institute of Aeronautics and Astronautics. This work is made available online in accordance with the publisher's policies. Please refer to any applicable terms of use of the publisher.

General rights

Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

Automated mesh deformation for CAD models which exhibit boundary topology changes

Liang Sun,¹ Weigang Yao,² Trevor T Robinson,³ Cecil G Armstrong⁴
Queen's University Belfast, School of Mechanical and Aerospace Engineering, Belfast, UK

and
Simão P Marques⁵
University of Surrey, Department of Mechanical Sciences, Guildford, UK

This paper describes an automated approach to deform an analysis mesh applied to a CAD model in response to a change in the shape of the CAD model. The process has been used to overcome a key issue that occurs during the shape optimization process. It is an advance over existing techniques because it works even when the boundary topology of the CAD model changes during the shape change, which other approaches cannot generally accommodate. To achieve the deformation, an analysis topology representation is created for each model, such that there is a one-to-one mapping between the analysis topology representing the models before and after the shape change. Strategies are presented to parameterize the superset entities in the analysis topology which have no global mathematical definition. The surface mesh is deformed in a way that allows the underlying surface/edge parameterizations to change and it is shown how the domain mesh can be updated based on the surface mesh deformation. If the models before and after shape change have the same genus, then a small dimensional change in the shape of the model will result in meshes with the same topology, even if the topology of the underlying CAD representation has changed. This is highly desirable when evaluating sensitivities to a design change during an optimization.

¹ Research fellow, School of Mechanical and Aerospace Engineering, and AIAA Member.

² Research fellow, School of Mechanical and Aerospace Engineering, and AIAA Member.

³ Reader, School of Mechanical and Aerospace Engineering, and AIAA Member.

⁴ Professor, School of Mechanical and Aerospace Engineering, and AIAA Member.

⁵ Senior lecturer, Department of Mechanical Sciences, and AIAA Member.

I. Introduction

Shape optimization requires a parameterization strategy to change the design's shape and an optimizer to drive the parameters to improve performance. There are different options for selecting the design parameters. CAD-free methods (such as using the mesh nodes as optimization parameters) have a rich design space but need a subsequent fitting process to transfer the optimized mesh representation to a geometrical CAD representation which is usually necessary to progress to manufacturing stages. This fitting operation will result in the shape created in the CAD system approximating the optimized shape, and the inherent inaccuracy in the process could reduce the performance of the optimum reached. In this work the ambition is to use the CAD model feature parameters [1][2][3], i.e. those created with the construction of the CAD model, as the optimization parameters, thereby optimizing the CAD model directly and eliminating the need for the fitting step. It has been applied to models created in the likes of CATIA V5 or Siemens NX. This approach means that the CAD model is available throughout the entire design, analysis and manufacture process without losing information about the geometric construction history. Modern CAD systems have significant modeling capabilities and allow the designer to easily create and modify models of complex shapes. The careful choice of features used to build the model can also be used to embed design intent in the model, where the features chosen will limit the shapes that the model can assume during update (e.g. a cylinder created by sweeping the sketch of a circle normal to the sketch plane will only ever be cylindrical).

Feature-based CAD models of complex components can contain thousands of parameters. Gradient-based optimization is increasingly popular in the aerospace and automotive industries for such models. This approach minimizes the objective function by modifying the parameters based on gradients, which predict the effect of each parameter on performance. One of the key roadblocks to optimization is that current design tools do not provide gradients of performance with respect to parameter values, meaning these must be computed as an additional process. In a direct method, the gradient of the objective function with respect to each parameter is calculated through time-consuming finite-difference evaluation for problems defined by large systems of ordinary or partial differential equations (PDE), such as those found in Computational Fluid Dynamics (CFD). The cost is especially large when the number of optimization parameters is large. A more efficient method can be the adjoint method, which removes the necessity of solving the flow field for each parameter. The adjoint solution is independent of the number of design parameters and can be calculated at a cost comparable to the corresponding CFD analysis [4][5][6][7].

Adjoint solvers can provide a surface sensitivity which gives a prediction of how the objective function will change when the model boundary (represented using the surface mesh) moves. To get the sensitivity for each parameter it is necessary to calculate the gradient of boundary position with respect to design parameters and combine this information with the adjoint sensitivity. Currently, commercial CAD systems do not provide this information directly. Haimes et al. [8][9] developed an open-source software called the Electronic Geometry Aircraft Design System (EGADS), which provides analytic parametric sensitivity. Lazzara [10] presented a methodology for constructing CAD models using concepts of multi-fidelity/multi-disciplinary geometry and design motion and similarly analytic parametric sensitivity can be achieved. Mykhaskiv et al. [11] applied automatic differentiation routines to the open source CAD kernel Open Cascade [12] to achieve this, but this was a significant amount of work since there are thousands of classes and functions in the CAD system source code. It also requires the user to have access to the CAD system source code, which is not common for the large CAD systems. Agarwal et al. [2] demonstrated a CAD-based approach which computes the boundary movement (referred to as design velocity) due to a CAD feature parameter change, and successfully demonstrated linking it with adjoint sensitivities in a CAD-based optimization framework to compute gradients. One of the challenges encountered in such an optimization process is that when the CAD parameter values change, causing a shape change, it is necessary for the analysis mesh to be updated correspondingly. A common approach to achieve this is through mesh deformation, where the existing mesh coordinates are modified to match the shape of the updated CAD model, but the mesh topology is kept the same. It is preferred over mesh regeneration because:

- Maintaining a constant mesh topology is helpful to keep the numerical error consistent while regeneration may introduce large changes in the distribution of numerical error
- For the adjoint method, the gradient of the objective function with respect to the volume/surface grid is needed, which requires the mesh to have constant topology
- There are no effective and robust algorithms to automatically generate structured mesh or the boundary layers of unstructured mesh, especially for complex models.
- Mesh regeneration can be very expensive compared to mesh deformation. However, this depends on the size of the model, the mesh software and the requirement for mesh quality

Mesh deformation in this context is usually achieved in two steps: first by deforming the mesh on the boundary of the component model, and then by using algorithms such as linear elasticity [13], spring analogy [14] or radial basis

function (RBF) [15] [16] to propagate the change of the surface mesh to the volume mesh. Using current engineering design tools, the CAD parameters cannot be directly related to the surface mesh movement, which means that change in the values of the CAD parameters doesn't inform how the surface mesh should move. Deforming the mesh on the surface requires mapping between entities in the initial and updated model. This is complicated when using a feature-based CAD system as the CAD model's boundary topology (the number of faces and how they are connected) or the face naming labels may change as the model updates after the parameter values change. As this makes it difficult to map the original to the updated CAD model it makes it difficult to update the mesh [2], [3], [17]. Fig. 1 illustrates how during the optimization of an aircraft, moving the position of the wing changes which of the boundary entities on the fuselage it intersects with. The inability to track these occurrences through the CAD system makes surface mesh deformation difficult.

Fig. 1 The wing-fuselage intersection across the boundary of a face

This paper describes an approach to move the mesh between two CAD models regardless of whether there are changes to face labels or boundary topology. As will be shown the key enabler is to obtain a consistent analysis topology for both models. Herein this is achieved using the concept of virtual topology [18]. Once the consistent analysis topology is available for both models, the surface mesh deformation is achieved based on constrained Delaunay triangulation (CDT). The remainder of the paper is organized as follows: Section 2 reviews the existing state of the art; Section 3 details the process of mapping topology using “virtual topology”, where existing face or edge entities may be subdivided or combined into sub- or super-sets; Section 4 introduces the parameterization of the superset entities; Section 5 details the surface mesh deformation process; Section 6 shows some results of the topological mapping and surface mesh deformation; Section 7 demonstrates how the surface mesh movement can be propagated into the volume; Section 8 gives the discussion and finally Section 9 draws the conclusions.

II. The existing state of the art

The research question for this work is how to robustly map the mesh applied to a CAD model to its updated form after it has undergone a parametric update. The key challenge is how to build the mapping between the boundary

representations for the two models when they are different in each model, or when the underlying surface/edge parameterizations change. In this section, some related work of surface mesh deformation is reviewed.

The goal of surface mesh deformation is to create a body-fitted surface mesh on an updated model after a shape parameter change. Herein the deformation is performed on the parametric space of the boundary representation (B-rep) and then converted to the 3D physical space to guarantee the mesh nodes lie on the surfaces. Most approaches assume that the perturbation is small and consequently the bounding topology of the model remains consistent after a model update. In such cases, the correspondence between different versions of the model can be built based on the names of boundary entities or the features used [17][19][20][21].

The simplest way of surface mesh deformation is to map the nodes from the original model to the updated model based on the normalized parametric position of a node on a surface. This is efficient, but not robust. For example, it is possible that the point with the same parametric position on the updated model does not lie on the face of the model. Another solution is to use the movement of the edges as input and propagate the movement into the face mathematically [22],[23],[24]. These methods work well on the condition that the boundary topology of the model does not change, which for a general CAD-based optimization framework cannot be guaranteed. There are other situations where the surface mesh is created on top of several faces, e.g. a block face in ANSYS ICEM CFD [25] stepping over several geometric faces. In this case, there are no global parametric definitions for the block faces. Truong et al. [26] gave an example of deforming surface mesh for block faces. In [26], a simple wing model is used. The correspondence between block faces in the original and updated model is constructed based on the z-component of the normal vector in the middle of the face. The surface mesh of the original face and a CAPRI tessellation [27] (triangular mesh) of the corresponding new face are reparametrized onto the same region in a 2D plane, aligning the corresponding boundaries. For each surface node, barycentric coordinates on the triangular element are calculated. The coordinates of nodes on the new face are obtained based on the barycentric coordinates, and the coordinates of the triangular mesh on the new face. In this method, the faces need to be four-sided and a regular triangular mesh is preferred. The irregularity of the triangular mesh affects the quality of the deformed mesh. That work did not deal with situations where new boundary topology entities appear in the updated model.

The contributions of this paper are three-fold:

- It proposes a novel approach to build a one-to-one mapping between the analysis topology of the models before and after the shape change

- It introduces strategies to parameterize the superset entities (groups of faces) in the analysis topology which have no global mathematical definitions
- The surface mesh is deformed in a way that allows the underlying surface/edge parameterizations to change, including ranges and directions

III. Mapping analysis topology

B-rep is the scheme used in almost all mainstream mechanical CAD packages (e.g. CATIA V5, Siemens NX, CREO) to represent the shape of the components. There are two aspects to a B-rep description, namely the geometry and topology. Geometry defines the shape of the entities, while topology describes how the entities are trimmed and connected. The geometrical entities include point, curve, and surface, while the topological entities mainly include vertex, edge, face and body. The one-to-one correspondence between the topological entities in an original and updated model is clear if the boundary topology does not change after the parameters update, even if the shape has changed. Where the boundary topology does change, it is described here how the concept of virtual topology can be applied to both models such that the resulting topological definition of each model (referred to herein as the analysis topology) is the same.

Virtual topology has been widely used for model simplification in the preparation of analysis models for FEA [18][28]. Most commercial FE tools include virtual topology capabilities, although the implementation differs from package to package. The key aspect of the technology is that it allows a new topological description of the model to be defined without modifying the underlying B-rep geometry and topology. Creating an analysis topology was introduced in [29]. The key process used in this work is to apply virtual topology merge operations to groups of adjacent faces, creating virtual superset entities. Fig. 2 shows an example where the virtual merging operations are performed on three B-rep faces (F_1 , F_2 and F_3) to create one superset face, vF_1 . The process to generate this also requires the creation of two superset edges $ve1$ and $ve2$.

Fig. 2 An example where virtual topology is used and meshes with improved quality are achieved. (a) three original faces; (b) an example of the topology hierarchy for face F_1 ; (c) generated mesh with poor quality due to the two sliver faces; (d) one superset face created by virtual merging operation; (e) the analysis topology hierarchy for the superset face; (f) generated mesh with improved quality.

One of the key requirements to be able to automatically build a one-to-one mapping between the topological entities in two models is the ability to distinguish the individual topological entities in the model. Different CAD systems use different approaches to name or tag the different entities. In some CAD systems, where the boundary topology does not change as a result of the parametric update, the mapping between the corresponding entities in the two models can be achieved by using the faces' entity names, or where name information is not available based on the order the entities appear in the model description. However, some CAD systems do not have a robust naming convention, or do not provide access to face names through the package's application programming interface (API). Where the CAD system does not have a robust or accessible naming convention it is often possible for the user to apply tags to the model. In this work, to identify each of the faces, a unique color flag was applied to each face, and this information was used to build the mapping. Any unique tag can be applied, but using color had the advantage that it provided both a unique flag for each face and allowed a visual verification of the results. At the outset of the process a different color is applied to each face of the baseline model (here CATIA V5 was used). The face color can be easily accessed in the CAD package through the API and it contains information about the inherited faces. This means faces with the same color before and after the parameters change are usually related, and in this work is used to guide the mapping process.

In the following description the models before and after the parametric change are referred to as m_1 and m_2 respectively. V/v , E/e , and F/f are used to represent the topological vertex, edge and face entities in the initial and

changed models respectively. The prefix letter “m-”, “v-” and “i-” before the topology entities represent mapping entities, virtual entities, and ignored entities.

A. Extracting topology and face colors

Once the faces have color tags applied in their native package, they are exported using the neutral format STEP. The process from this point on is therefore independent of the CAD system used to build the model. From each STEP file the topology of the model and color information is extracted using the open source CAD kernel Open Cascade. This information is stored in a SQL database [30]. The database is used to keep track of the unique identifiers assigned to each entity to maintain a record of the superset entities which will be created during the process, and to store the virtual topology relationships. The following tables are created in the database:

- Entity: stores a global database ID for each topology entity and the manifold dimension of the entities (0 for vertex, 1 for edge and 2 for face).
- Color: stores the global database ID and the RGB value assigned to a face.
- Topology: stores the ID of an entity, along with the IDs of its bounding entities and the relative orientations of the entities.
- Virtual topology: stores the ID of the superset entity and the ID of the entities which make it up, e.g. vF_1 is a superset of F_1 , F_2 , and F_3 in Fig. 2.
- Analysis topology: stores the analysis topology after applying the virtual topology operations, e.g. vF_1 is bounded by ve_1 , e_2 , ve_2 and e_9 in Fig. 2.

Fig. 3 Relations in the SQL database

Note that this database builds on the work in [29], where a detailed description is provided for using virtual topology merge and split operations to create a fit-for-purpose analysis topology. In this work, color is added as a new relation rather than as an additional attribute in the entity table.

B. Building an initial mapping

If boundary topology does not change, then the face color can be used to define the faces that are likely to correspond between the models, although this on its own does not provide a robust mapping. The conditions for mapping face F in m_1 to a face f in m_2 is that it satisfies:

- F and f share the same color tag
- F and f have the same topology (i.e. same number of bounding edges and vertices)
- F and f have the same connectivity to their adjacent faces, i.e. the adjacent faces' colors are the same

Fig. 4 shows a part of a CAD model created in CATIA V5 after tens of geometric operations, including Boolean operations, blending and so on. The model before and after parametric update is shown in Fig. 4(a) and (b) respectively. This example demonstrates different scenarios which need to be considered when building a mapping. For example, for a face F in m_1 , it is possible that there is a face of the same color for it to map to in m_2 , but the actual mapping required would be to a superset of faces where the face with the same color is merged with its neighbor, e.g. $F_4 \rightarrow f_4 + f_5$. In this work a one-to-one entity map is created between superset faces and edges created in either models.

The process starts by identifying faces with the same color in m_1 and m_2 . For models undergoing a parametric update the shapes will still be largely similar and most faces in m_1 will have a unique face to map to in m_2 which shares the same color as them, e.g. $F_1 \rightarrow f_1$ in Fig. 4. Where a face F in m_1 has no face with the same color in m_2 , such as F_2 and F_3 in Fig. 4 these faces form a set F_{dis} (named as these faces disappear in m_2). Similarly, some faces appear in m_2 with a color which is not applied to any face in m_1 , e.g. f_2 , f_3 and f_4 in Fig. 4. These faces form a set F_{new} . In CATIA v5 all new faces will have the same default color (set as red in this work). Sometimes, the color applied to a face F in m_1 is applied to more than one face in m_2 . In this case, a virtual superset is created for the adjacent faces in m_2 (e.g. the superset example of merging faces in Fig. 2), so there is only one virtual face in m_2 corresponding to F .

Fig. 4 An example of the faces' color of a CAD model which has undergone a parametric update: (a) m_1 ; (b) m_2 . F_1 and F_4 in m_1 have a unique face in m_2 which shares the same color, i.e. f_1 and f_5 respectively. F_2 and F_3 have no faces with the same color in m_2 . f_2 , f_3 and f_4 in m_2 have no faces with the same color in m_1 .

In this work, for each pair of faces with the same color, they are mapped only if:

- Their topologies are the same
- F is not adjacent to a face in F_{dis}
- f is not adjacent a face in F_{new}
- F and f are not adjacent to any face whose topology changes

During the work it was observed that in some special cases, a face which is present in m_1 corresponds to a face with the new entity color applied in m_2 , e.g. $F_2 \rightarrow f_2$. In this case if the two faces' topologies are the same, and the colors of all of their adjacent faces are the same, they are mapped to each other.

The mapping faces can then be used to map edges. While some CAD systems will allow names or tags to be applied to edges others do not, and so edge identifiers from the CAD system cannot be relied on for mapping. Here, the mapping of edges is achieved based on the topology of the faces. For an edge in m_1 , if its two bounding faces have been mapped, and the faces only share one common bounding edge, e.g. e_1 in Fig. 5(a), then this edge is mapped to the common edge of the mapped bounding faces. If two faces share more than one bounding edge, e.g. e_2 - e_4 in Fig. 5(b), the wire entity (terminology used in Open Cascade, to refer to a list of oriented edges) of the mapped faces is queried and the edges in the wire are reordered to make a mapped edge the first of the wires, e.g. $me_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4 \rightarrow me_5 \rightarrow me_6$. All unmapped edges can then be mapped based on their indices in the wire. A vertex will be mapped if its bounded edges are mapped and share one common vertex, as the v_1 shown in Fig. 5(c). It can also be mapped if its bounded faces' colors are all mapped and only share one common vertex, such as the v_1 shown in Fig. 5(d). This is referred to as a top-down mapping process.

Fig. 5 Top-down mapping process (a) map the common edge of two mapping faces; (b) map other edges of a mapping face; (c) map vertex based on its bounding edges; (d) map vertex based on its bounding faces

C. Updating the mapping

After the process above has been used to create a mapping, a bottom-up process is used to complete the mapping for all entities. For example, an edge which is not mapped during the process in the previous section, will be mapped if its two bounding vertices are mapped, as e_1 shown in Fig. 6(a). Similarly, two faces will be mapped if all their bounding edges are mapped, Fig. 6(b).

Fig. 6 Bottom-up map: (a) map an edge based on its bounding vertices; (b) map a face based on its bounding edges

Once mappings have been defined based on this bottom-up process, the new mapping faces may allow the process in the previous section to identify yet more new mappings. Both processes are cycled iteratively until there are no vertices or faces left to map.

D. Checking mapping vertices of faces in F_{dis} and F_{new}

For models where the parameter value update does not cause the topology to change, the above process will map all the entities in the model. Where the topology has changed, a superset will be created based on the local information of the vertices. Fig. 7 shows an example of the process being applied to a model where the top row (a) represents the different stages of virtual topology being applied to the original model and the bottom row (b) shows the application of virtual topology to the parametrically updated model.

First, a mapping process is carried out for any mapped vertices which bound faces in the F_{dis} set, e.g. mv_1 which bounds face df_1 in Fig. 7(a). To achieve this mapping, if the number of bounded faces is the same for the two mapped vertices (4 in this case), and there exists two pairs of faces which have been mapped (i.e. mf_1/mf_1 , mf_2/mf_2), the faces bounded by the two vertices can be ordered. Here, in the corresponding lists for both models, df_1 will have the same

position for the list corresponding to m_1 as f_1 does for the m_2 list. This information, along with the fact that F_1 which has the same color as f_1 suggests that virtual face vF_1 should be generated. A new color is then assigned to vF_1 and f_1 . The merging of face dF_1 and F_1 means that the common edge between them is ignored in the analysis topology, i.e. iE_1 . If the valence of a vertex bounding an ignored edge becomes two (as a consequence of the ignored edge no longer being present), the vertex is also ignored, i.e. iV_1 , and a superset edge is created, i.e. vE_1 .

Fig. 7 Mapping process based on mapped vertices bounding a face in the set F_{dis} : (a) m_1 ; (b) m_2

Then the mapped vertices of a face in the set F_{new} are checked in a similar way as shown in Fig. 8. A superset face is created by merging nf_1 and f_2 in m_2 . The common edge e_1 is ignored and a superset edge ve_1 is created.

Fig. 8 Mapping process based on mapped vertices bounding a face in the set F_{new} : (a) m_1 ; (b) m_2

After the above two steps, the procedures in section C are then performed to update the mapping result.

E. Mapping remaining vertices

For any vertices in m_1 that remain unmapped, a set S_1 is created containing its bounding faces, excluding those in F_{dis} , $S_1 = \{F_i\}$. From this, a set S_2 is created of the pairs of adjacent faces in S_1 , $S_2 = \{(F_1, F_2), \dots\}$. A vertex v_i in m_2 is defined as a candidate to map with V_1 if the following conditions are satisfied:

- The number of faces incident at v_i (i.e. its valence) is the same as that of V_1
- v_i is the common vertex of two faces whose colors are the same as that of a pair in S_2

For each candidate identified, the same check is performed to identify its candidate in m_1 . A vertex V_1 is mapped to v_1 only if they are the only candidates of each other.

In Fig. 9, F_4 is a face that has a color applied that is present in m_1 and not m_2 , and f_4 is a face represented by a new color. For V_1 in Fig. 9(a), $S_2 = \{(F_1, F_2), (F_1, F_3)\}$. f_1 and f_2 in m_2 share vertex v_2 and f_1, f_3 share vertex v_1 . Based on the criteria above, V_1 has only one candidate v_2 . Similarly, when checking v_2 , V_1 is its only candidate in m_1 . Therefore, V_1 is mapped to v_2 .

For V_1 in Fig. 9(c), $S_2 = \{(F_1, F_2), (F_1, F_3), (F_2, F_3)\}$. In Fig. 9(d), f_1, f_3 share vertex v_1 and f_2, f_3 share vertex v_2 . This means V_1 has two candidates in m_2 , so V_1 is not mapped and the vertex it should be mapped to remains to be determined by later processes.

Fig. 9 Map remaining vertices

To improve the computational efficiency of the process by reducing the number of candidates, an angle tolerance is introduced. This tolerance constrains the change of the corner angles of faces incident at the vertex. This means that the change should be within a defined limit. In this work, this limit is chosen to be 45 degrees. After the identification of new mapped vertices, the earlier procedures are again applied. For the example in Fig. 9(a), a superset will be created for F_4 and F_5 , and a superset will be created for f_3 and f_4 in Fig. 9(b).

The next step is to map a face F in m_1 or f in m_2 (In the following, the notation F/f will be used for short) whose boundary topologies change, e.g. if an inner loop of edges disappears. Faces that F/f should be merged with need to satisfy the following conditions:

- Adjacent to F/f and unmapped
- Merging the face with F/f will recover the topology of f/F

Any faces that satisfy these conditions are tested in turn. After this step, if non-mapped faces still exist, a superset will be created for those connected faces. Note that the edges surrounding the connected groups of the non-mapped faces must have been mapped in the previous process. With this information it can be determined which face/superset

face the non-mapped superset faces should be mapped to. For the model in Fig. 4, two superset faces are created for m_1 and three superset faces created for m_2 , as shown in Fig. 10.

Fig. 10 Face map result of the model in Fig. 4. (a) left: original model m_1 ; right: the analysis topology of m_1 , two superset faces created; (b) left: updated model m_2 ; right: the analysis topology of m_2 , three superset faces created

Cycling through the processes above for a range of pairs of models, the process has been shown to be robust in creating and mapping analysis topologies for the parametrically modified models.

IV. Parameterization of superset entities

Once the analysis topologies have been created, and a mapping between the models exists, the next requirement is to calculate the parametric position of each mesh node on each analysis topology entity in the original model. A strict definition of a parameterized curve is a differentiable mapping of a parameter $t \in I$ to a point $\gamma(t) = (x(t), y(t), z(t)) \in R^3$, where I is an interval of the space R . This means given a parameter t , a unique point in R^3 with the coordinate xyz can be calculated as well as the first and higher order derivatives at that point. Given a point on the curve with xyz coordinate, a unique parameter t can be calculated. These queries can be easily done by calling the APIs in a CAD engine. Similarly, a parameterized surface is a differentiable mapping of a parameter $(u, v) \in R^2$ to a point $(x(u, v), y(u, v), z(u, v)) \in R^3$. As a superset entity is comprised of more than one parameterized geometric entities, there are no global mathematical definitions of the shape of a superset entity. In this work, it is the bijective mapping between a parameter space (t for an edge and (u, v) for a face) and a physical space R^3 that is required and

there is no requirement for the differentiability. A parameterization technique has been applied to both the superset edges and superset faces.

A. Parameterization of a superset edge

In differential geometry, any parametrized curve has a natural parametrization, defined using the arc length of the curve, i.e. $t = \frac{l}{L}, t \in [0, 1]$ where L is the arc length of the whole curve and l is the arc length from the starting point of the curve to a querying point on the curve. This means that no matter how the curve is defined, it can always be re-parameterized using its arc length. Here, the same idea is applied to a superset edge to get a parameterization based on the arc length. This is explained with respect to the example in Fig. 11, where a superset edge ve_1 is created on top of two subset edges e_1 and e_2 . This superset edge ve_1 has two vertices v_1 and v_3 which are the non-shared vertices of e_1 and e_2 respectively. It is oriented from v_1 to v_3 (the details of the creation and orientation of the superset entity can be found in [29]). Note the orientation of the subset edges could be different from the superset edge, e.g. the orientation of e_2 is opposite to that of the superset edge ve_1 . Therefore, some information needs to be obtained in order to create the parameterization:

- A list of the ordered vertices when walking along the superset edge from its starting vertex, i.e. v_1, v_2, v_3
- A list of ordered edges when walking along the superset edge from its starting vertex, i.e. e_1, e_2
- A list of the orientation of the above ordered edges relative to the superset edge. It will have a value of +1 if its orientation is same as that of the superset edge, i.e. +1, -1 in Fig. 11
- A list of arc lengths of the ordered edges, i.e. L_1, L_2
- The arc length of the superset edge, L

Fig. 11 (a) subset edges; (b) superset edge

This information can be obtained from a query to the SQL database or a query to the CAD engine, which in this case is Open Cascade. With this information, given a parameter $t \in [0, 1]$, its xyz coordinate on the edge can be obtained using Algorithm 1.

Algorithm 1: Parameterizing a superset edge, $t \rightarrow xyz, t \in [0, 1]$

$t \rightarrow xyz, t \in [0, 1]$

1. Calculate the arc length from the querying point to the starting vertex of the superset edge $\rightarrow tL$
 2. Identify which subset edge that the point lies on $\rightarrow e_i$
 3. Calculate the arc length on $e_i \rightarrow c = tL - \sum_{m=1}^{i-1} L_m$
 4. Calculate its parameter t' on e_i
 5. If e_i has the same orientation as the superset edge $\rightarrow t' = \frac{c}{L_i}$
 6. Else $\rightarrow t' = 1 - \frac{c}{L_i}$
 7. Calculate xyz from t' and e_i
-

Similarly, given a point on the edge with coordinate xyz , a parameter $t \in [0, 1]$ based on arc length can be obtained using Algorithm 2.

Algorithm 2: Parameterizing a superset edge, $xyz \rightarrow t, t \in [0, 1]$

$xyz \rightarrow t, t \in [0, 1]$

1. Identify which subset edge the point lies on $\rightarrow e_i$
 2. Calculate the arc length from the vertex of e_i , which appears the first in the ordered list of vertices $\rightarrow L'$
 3. Calculate the parameter $\rightarrow t = \frac{L' + \sum_{m=1}^i L_m}{L}$
-

B. Parameterization of a superset face

The objective of the parameterization of a superset face is to set up a bijective mapping between a parameter $(u, v) \in R^2$ and a point on the superset face with coordinate $xyz \in R^3$. This is not as straightforward as the superset edge if the underlying geometries of the subset faces are different. However, in the past decade extensive methods have been developed to build a bijective mapping between a parameter (u, v) and a discretized representation of a surface [31], [32], e.g. a piecewise-linear triangulated surface. In general, this kind of parametrization will incur some metric distortions since it is trivial to parameterize individual tri element, but for a set of tri elements, they will not fit together on the plane. In this work, a technique called as-rigid-as-possible (ARAP) parameterization is used, which first flattens the triangles on a plane and then minimizes the stretch as much as possible by restricting the local transformations to rotations only [33]. Herein, an implementation of the ARAP technique in CADfix [34] is used,

which includes an untangle function to guarantee a bijective mapping in complex cases [35]. This function is called through the CADfix python API. Fig. 12(a) shows a superset face colored in green, which is comprised of two subset faces. Fig. 12(b) shows the discretized representation of the superset face and two set of lines representing the u and v directions which are calculated using the ARAP parameterization. As can be seen on the detailed view of Fig. 12(b), the uv lines step over the two subset faces.

Fig. 12 (a) a superset face colored in green and comprised of two subset faces; (b) the discretized representation of the superset face and the uv lines calculated using the ARAP parameterization

V. Surface mesh movement

The surface mesh movement is performed in the parametric space of the geometry entities and then mapped back to the physical space. This guarantees that the deformed surface mesh always lies on the CAD geometry. However, the ranges or directions of the parametric space of the two mapped entities could be different. The following section introduces how the surface mesh nodes of m_1 are mapped to the updated model m_2 in this case. Let N_x represent a surface mesh node of m_1 and n_y be its mapped mesh node of m_2 . N_x is classified into one of the three types: a node lying on a mapped vertex, a node lying on a mapped edge (excluding the vertices of the edge) and a node lying in a mapped face (excluding the boundary of the face). The surface mesh of m_1 are deformed in a hierarchal way by deforming nodes lying on vertices, then those lying on the edges, and finally those lying in faces.

A. Deforming nodes on vertices

Deforming nodes on the mapped vertices is very straightforward. For a node N_x , the vertex that it lies on is identified, V_i . From the mapped analysis topology, the corresponding vertex of V_i is queried, v_j , which is then the location of n_y .

B. Deforming nodes on edges

For a mesh node on a mapped edge E_i , its parametric position, t , on that edge can be calculated. If the parameter is not given as a measure of arc length, within the process the edge is reparametrized such that it is. This makes a normalized range of the parameter $t \in [0, 1]$. The mapped edge of E_i is then queried, e_j . The orientations of the mapping edges are checked to ensure the correct position is calculated. It is possible that after the parameter change the orientation of an edge will be reversed. Fig. 13 shows an example where this happens for the highlighted edges. Without considering the orientations of the mapped edges, the resulting edge mesh can be tangled, as shown in Fig. 13(b). The orientations of the edges can be easily retrieved from the database. If the orientations of the two mapped edges are the same, the new mesh node n_y will lie on e_j and with a parameter t . Otherwise, the location of the new mesh node n_y will lie on e_j and with a parameter $1-t$.

Fig. 13 The orientations of the mapped edge are different for the highlighted edge after the parameters change. (a) the edge mesh of m_1 ; (b) the edge mesh of m_2 if not considering the edge orientations

C. Deforming mesh nodes on faces

As mentioned before, the uv ranges and uv directions of parametric space of a face could be different, e.g. when using the ARAP to parameterize a superset face, there are no guarantees that the uv directions will be same as its mapped faces. This is also true even for a single geometric face. The change of uv range can be easily solved by normalizing the range to $[0, 1]$. However, it is difficult to make changes to make the uv directions the same. This means a consistent reference system is required to indicate the location of a mesh node on the face. Here, a constrained Delaunay Triangulation (CDT) is created in the parameter space of a face in m_1 , which is then used for this purpose.

The deformation process is explained with respect to the example in Fig. 14 (a) and (b) shows the parametric space of a face in m_1 and m_2 respectively. In Fig. 14 (a), a CDT is created using nodes on the boundary of the face.

Those nodes have been mapped in the previous steps, e.g. $N_1 \rightarrow n_1, N_2 \rightarrow n_2, N_3 \rightarrow n_3$. This means any CDT triangle in m_1 will have a mapped triangle in m_2 . Therefore, for any mesh node on a face in m_1 , if the CDT triangle that it lies in is identified as well as its relative location in that triangle, it can then be mapped to a corresponding triangle in m_2 with the same location. Here, the location of a node in a triangle is defined using its barycentric coordinates rst in the triangle. Let $N_1 = (u_1, v_1)$, $N_2 = (u_2, v_2)$ and $N_3 = (u_3, v_3)$ be the vertices of a CDT triangle and $N_x = (u_x, v_x)$ be a mesh node to be mapped. From the definition of the barycentric coordinates

$$\begin{bmatrix} u_x \\ v_x \\ 1 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} r \\ s \\ t \end{bmatrix} \quad (1)$$

where $0 \leq r, s, t \leq 1$. The barycentric coordinates can then be calculated by

$$\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} u_b - u_a & u_c - u_a \\ v_b - v_a & v_c - v_a \end{bmatrix}^{-1} \begin{bmatrix} u_p - u_a \\ v_p - v_a \end{bmatrix}. \quad (2)$$

The location of mapped mesh node of N_x, n_y , is then calculated using the rst and the parametric coordinates of n_1, n_2 and n_3 .

Fig. 14 (a) the parametric space of a face in m_1 and CDT created using nodes on the boundary of the face; (b) the parametric space of the mapped face in m_2

To provide an efficient way of identifying which CDT triangle N_x lies in a multi-dimensional binary search tree (a KD-tree) is created using the CDT vertices. Given a mesh node N_x , the KD-tree can efficiently return the closest node in the tree to that mesh node. One of the CDT triangles “ T_{abc} ” bounded by the closest node is selected to test if N_x lies in the triangle by checking its barycentric coordinate rst relative to the triangle, as shown in Fig. 15. If not, the next triangle to be tested is determined in a way same as that used in [2]:

- If $r < 0$, the triangle adjacent to the edge “ bc ” will be tested
- If $s < 0$, the triangle adjacent to the edge “ ac ” will be tested
- If $t < 0$, the triangle adjacent to the edge “ ab ” will be tested

The test process continues until it finds a triangle that N_x lies in.

Fig. 15 Use barycentric coordinates rst to determine the next CDT triangle to test

VI. Results

This section shows the results of applying the surface mesh movement process to several models. Fig. 16 shows a wing model created using the “multi-section surface” command in CATIA. The wing surface is swept through airfoil profiles guided by two curves passing the leading and trailing edges. The guiding curves are created in a way so that it allows a blend feature to be created as shown in Fig. 16(b). A parameter is defined to control the movement of the airfoil profiles towards the wing tip along the z direction. For the example in Fig. 16(b), the parameter value is changed so that the wing tip is rotated upwards by 50° . In this model the boundary topology does not change as a result of the parameter update, but the mesh fitting on m_2 is shown to be good and its structure is shown to be the same on each model. A comparison of the mesh quality between the two meshes in terms of the scaled Jacobian is given in Fig. 17. As it shows, the mesh qualities are very similar.

Fig. 16 Surface mesh deformation of a wing model (the topology does not change): (a) surface mesh before the parameter changes; (b) the deformed surface mesh after the parameter changes

Fig. 17 A comparison of the scaled Jacobian of the surface meshes before and after the parameter changes

The second test case is a model of the Onera-M6 wing created in CATIA V5. Like the test case in Fig. 16, the wing is created by sweeping through four airfoil profiles which are equally distributed along the span wise direction. Each airfoil profile is defined by two 6-order Bezier curves. When the z coordinate of the first control point after the leading edge on each profile increases above a certain value, the single upper surface becomes two surfaces, with an extra sliver face created close to the leading edge, as the m₂ shown in Fig. 18. A superset face is created for the upper surface colored in green. The result of surface mesh deformation is shown in Fig. 19. The mesh on the updated model fits the updated model well and the mesh structure is the same on both models. Note that the sliver surface is green, suggesting it should be merged with the top surface, rather than the bottom surface.

Fig. 18 Onera-M6 wing model, sliver face generated close to the leading edge in m₂

Fig. 19 Surface mesh deformation of the Onera-M6 model shown in Fig. 18: (a) surface mesh before the parameter changes; (b) the deformed surface mesh after the parameter changes

The third test case is a model of a ventilation duct which is highly curved and contains many fillet features. With a complex boundary topology. As shown, the yellow face becomes two faces after the parameter change (one light green and one red in Fig. 20 (b) left) due to the significant change of the blend curvature. A superset is created as shown in Fig. 20 (b) right. The surface mesh deformation result is given in Fig. 21.

Fig. 20 A ventilation duct model (a) left: original model m_1 ; right: the analysis topology of m_1 (b) left: the model after parameter changes m_2 ; right: the analysis topology of m_2

Fig. 21 Surface mesh deformation of the s-bend model in Fig. 20: (a) surface mesh before the parameter changes; (b) the deformed surface mesh after the parameter changes

The surface mesh deformation for the model shown in Fig. 10 is given in Fig. 22.

Fig. 22 Surface mesh deformation result of the test case shown in Fig. 10: (a) surface mesh before the parameter changes; (b) the deformed surface mesh after the parameter changes

In Fig. 23, F_1 - F_5 in m_1 disappear after the parameters change and two new faces f_1 and f_2 appear. After the mapping process, f_1 and f_2 are mapped to F_1 and F_2 respectively. Also, three superset faces are created for m_1 as shown in Fig. 23 (a) right. The mesh result is given in Fig. 24.

Fig. 23 Mapping of the analysis topology of a ventilation duct: (a) left: original model m_1 ; right: the analysis topology of m_1 (b) left: the model after parameter changes m_2 ; right: the analysis topology of m_2

Fig. 24 Surface mesh deformation result of the test case shown in Fig. 23: (a) surface mesh before the parameter changes; (b) the deformed surface mesh after the parameter changes

In the last test case, the intersection of the two objects moves across the boundary of a face, as shown in Fig. 25

. This

example is a representative of a wing intersecting a fuselage body. In the example shown the fuselage is made up of many surfaces and moving the wing forward on the aircraft will change the surfaces it intersects with. During the process a superset face is created for the F_1, F_2 in m_1 and f_1, f_2 in m_2 respectively, which then results in a one-to-one mapping between the topology. The surface mesh deformation result is given in Fig. 26. It is shown how the mesh structure remains the same after the update, but it should be noted that the edge between f_1 and f_2 is not respected in the deformed surface mesh. The nodes which were position along the edge between the faces F_1 and F_2 in the original model now bend around the leading edge, as shown in Fig. 26(b).

Fig. 25 Mapping of the analysis topology when the intersection moves across a face's boundary: (a) left: original model m_1 ; right: the analysis topology of m_1 (b) left: the model after parameter changes m_2 ; right: the analysis topology of m_2

Fig. 26 Surface mesh deformation result of the test case shown in Fig. 25: (a) surface mesh before the parameter changes; (b) the deformed surface mesh after the parameter changes

VII. Volume mesh deformation

After the surface mesh deformation has been computed, this information can be used to update the domain mesh applied to the model. The new surface mesh (or surface mesh displacements depending on the algorithms) can be used as an input to a volume mesh deformation step. There are many algorithms that can be used to move a volume mesh based on the movement of its boundary mesh, the most popular of which are listed in the introduction. The following shows an example using the linear elasticity function in the open source package SU^2 [5] to propagate the surface mesh deformation into the volume.

The test case is a wing model used for aerodynamic analysis. The wing is comprised of 15 faces and the wing tip is rotated upwards 20 degrees, as shown in Fig. 27.

Fig. 27 Wing test case: (a) original wing; (b) wing tip rotated upwards 20 degrees

The fluid domain is meshed with 843,253 tetrahedral elements and there are 185,044 triangle elements on the wing surfaces, as shown in Fig. 28.

Fig. 28 (a) the volume mesh; (b) the surface mesh on the wing

Moving the surface mesh using the approaches in this paper took 34 seconds. Afterwards, the volume mesh is deformed using the linear elasticity method in SU^2 the process of which takes 115s. The mesh results before and after deformation are given in Fig. 29 .

Fig. 29 Example of using linear elasticity to propagate the surface mesh deformation into the volume: (a) mesh before parametric change; (b) mesh after the parametric change

VIII. Discussion

This paper has described an approach to move an analysis mesh between two dimensionally different versions of a CAD model. The context of this work is where the model has been parametrically perturbed during an optimization process, but there are many examples of when such a capability would be useful (e.g. two different variants or design iterations of a product or moving a mesh from an analysis model to a scanned geometry of a manufactured model).

The process is an advance on current processes as it does not impose any restriction on the consistency of the boundary topology between the models, but instead uses a virtual topology-based procedure to create an analysis topology representation of each model, such that there is a one to one correspondence between the analysis topologies for each. As this process uses virtual topology it does not edit the underlying geometric descriptions in any way. The analysis topology and mappings are stored in an SQL database, and as such are easily exploitable and modifiable. For all the examples in the paper the creation of the analysis topology, mapping and mesh movement processes were carried out on a desktop workstation with 32 GB RAM. The processing time is thought to be small compared to the cost of determining how to move the mesh manually. It also saves time on having to export/import and set-up a new case from the meshing software and into SU^2 .

In an analysis of desirable criteria for assessing mesh generation methods, Sabin [36] identified *continuity*, ensuring that the mesh on an updated geometry will have the same connectivity and small changes in nodal coordinates as a desirable property. This is the implied assumption in this work. In the context of the work this process was created for, where the difference in the model will be due to parametric updates with small changes in numerical value, this assumption is thought to be valid. However, in scenarios where the parametric updates are large, or where there is a

significant difference in the shape of the model, then the suitability of this approach may have to be questioned. Two examples have been identified where the change in geometry cannot be treated as small. The first is when there is a change in the genus of the two models. The second is when the mapping of the CDT from m_1 to m_2 produces triangles with a reversed orientation (e.g. rotating the wing through 180 degrees for a wing-fuselage model or moving a circular inner loop of a long-slender rectangle face from one end to the other end). If these cases happen, it would not be possible to get a valid mesh with the same mesh structure to fit both models and a remeshing operation would be required. As with all other mesh deformation methods, it is advisable to check the mesh quality and mesh connectivity (e.g. to avoid self-intersection) for the deformed mesh, especially when the change in the model shape becomes large to determine the suitability of the resulting mesh for the application it is being used for.

One of the key requirements for this work is the ability to identify each face in a CAD model. As already stated, different CAD systems have different approaches to this, both in terms of how they apply unique tags/names/labels to entities in the models, and how they export them to the STEP format. In this work, a unique tag was applied through application of a unique color to each face, but the visual aspect of color is not important for the process and it should be easily adaptable to any naming/tagging/labelling process depending on the CAD system it is applied to.

In this work, Open Cascade was used for extracting the color attribute of faces from the STEP files and for querying topological information, such as edges bounding a face, and geometrical information, such as parameter of a node on an edge. The reason for choosing Open Cascade rather than the package where the model is originally built is that it keeps the mapping work independent of the modelling software. The work here can be easily adapted from one CAD package to another.

This work also used the novel step of parameterizing the superset entities in the model to both compute the parametric position of a node on a superset in the original model, and to then compute the position of the node on the perturbed model base on the parametric position. ARAP and CDT were both used for this, the former providing the parameterization of a superset face to a planar domain and the latter providing the relative position of nodes on faces to the triangles defined by nodes on edges and vertices.

It was demonstrated in this paper that the surface mesh was moved from one model to another parametrically modified model, and that this in turn was used to update the volume mesh as would be required to conduct the computational simulation of the second model. This paper has not contributed any advances to the volume mesh

moving algorithms themselves. In future, the work in this paper will be integrated into a CAD-based shape optimization process.

IX. Conclusions

The paper describes an approach to mesh deformation in CAD-based shape optimization in the context of allowing the boundary topology of the model to change. It shows:

- A strategy to build a one-to-one mapping analysis topology between models before and after parametric change using the virtual topology operations
- A parameterization strategy for superset edges and faces, which have no global mathematic definition, to have a bijective mapping between parametric and xyz coordinates
- A method to deform the surface mesh hierarchically from vertex to edge to face, which works even when the parametric space range and orientation changes
- Successful applications on a few representative single-part industrial test cases. The proposed methods bring a significant step forward to use complex industrial CAD models for shape optimization. It allows the designed CAD model to be directly used in shape optimization and keeps the geometry of the underlying CAD model unchanged.

Acknowledgments

We are thankful to the UK Engineering and Physical Sciences Research Council (EPSRC) for supporting this work through grant EP/P025692/1.

References

- [1] E. Hardee, K.-H. Chang, J. Tu, K. K. Choi, I. Grindeanu, and X. Yu, “A CAD-based design parameterization for shape optimization of elastic solids,” *Adv. Eng. Softw.*, vol. 30, no. 3, pp. 185–199, 1999.
- [2] D. Agarwal, T. T. Robinson, C. G. Armstrong, S. Marques, I. Vasilopoulos, and M. Meyer, “Parametric design velocity computation for CAD-based design optimization using adjoint methods,” *Eng. Comput.*, pp. 1–15, 2017.
- [3] T. T. Robinson, C. G. Armstrong, H. S. Chua, C. Othmer, and T. Grahns, “Optimizing parameterized CAD geometries using sensitivities based on adjoint functions,” *Comput. Aided. Des. Appl.*, vol. 9, no. 3, pp. 253–268, 2012.

- [4] M. B. Giles and N. A. Pierce, “An Introduction to the Adjoint Approach to Design,” *Flow, Turbul. Combust.*, vol. 65, no. 3/4, pp. 393–415, 2000, doi: 10.1023/A:1011430410075.
- [5] T. D. Economou, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso, “SU2: An Open-Source Suite for Multiphysics Simulation and Design,” *AIAA J.*, vol. 54, no. 3, pp. 828–846, 2016, doi: 10.2514/1.J053813.
- [6] A. Jameson, “Aerodynamic design via control theory,” *J. Sci. Comput.*, vol. 3, no. 3, pp. 233–260, Sep. 1988, doi: 10.1007/BF01061285.
- [7] D. Jones, J.-D. Müller, and F. Christakopoulos, “Preparation and assembly of discrete adjoint CFD codes,” *Comput. Fluids*, vol. 46, no. 1, pp. 282–286, Jul. 2011, doi: 10.1016/J.COMPFLUID.2011.01.042.
- [8] R. Haimes and J. Dannenhoffer, “The engineering sketch pad: A solid-modeling, feature-based, web-enabled system for building parametric geometry,” in *21st AIAA Computational Fluid Dynamics Conference*, 2013, p. 3073.
- [9] R. Haimes and M. Drela, “On the construction of aircraft conceptual geometry for high-fidelity analysis and design,” in *50th AIAA Aerospace sciences meeting including the new horizons forum and aerospace exposition*, 2012, p. 683.
- [10] D. S. Lazzara, “Modeling and sensitivity analysis of aircraft geometry for multidisciplinary optimization problems,” Massachusetts Institute of Technology, 2012.
- [11] O. Mykhaskiv *et al.*, “NURBS-based and parametric-based shape optimization with differentiated CAD kernel,” *Comput. Aided. Des. Appl.*, vol. 15, no. 6, pp. 916–926, 2018, doi: 10.1080/16864360.2018.1462881.
- [12] “OpenCASCADE.” [Online]. Available: <https://www.opencascade.com/>.
- [13] T. Baker and P. Cavallo, “Dynamic adaptation for deforming tetrahedral meshes,” in *14th Computational Fluid Dynamics Conference*, 1999, p. 3253.
- [14] J. T. Batina, “Unsteady Euler airfoil solutions using unstructured dynamic meshes,” *AIAA J.*, vol. 28, no. 8, pp. 1381–1388, 1990.
- [15] A. De Boer, M. S. Van der Schoot, and H. Bijl, “Mesh deformation based on radial basis function interpolation,” *Comput. Struct.*, vol. 85, no. 11–14, pp. 784–795, 2007.
- [16] G. A. Strofylas, G. N. Lygidakis, and I. K. Nikolos, “An agglomeration strategy for accelerating RBF-based mesh deformation,” *Adv. Eng. Softw.*, vol. 107, pp. 13–37, 2017.
- [17] D. Fudge, D. Zingg, and R. Haimes, “A CAD-free and a CAD-based geometry control system for aerodynamic shape optimization,” in *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005, p. 451.
- [18] A. Sheffer, T. Blacker, J. Clements, and M. Bercovier, “Virtual Topology Operators for Meshing,” *Int. J. Comput. Geom. Appl.*, vol. 10, no. 3, pp. 309–331, 2000, doi: 10.1142/S0218195900000188.
- [19] A. Ronzheimer, “Aircraft Geometry Parameterization with High-End CAD-Software for Design Optimization,” in *Proceedings ECCOMAS*, 2012.

- [20] J. J. Alonso, J. Martins, J. J. Reuther, R. Haimes, and C. A. Crawford, "High-fidelity aero-structural design using a parametric CAD-based model," *AIAA Pap.*, vol. 3429, p. 2003, 2003.
- [21] W. E. Brock *et al.*, "Adjoint-based design optimization using CAD parameterization through CAPRI," in *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics, Nashville, TN, USA, 2012*, pp. 2012–2968.
- [22] M. J. Martin-Burgos, D. González-Juárez, and E. Andrés-Pérez, "A novel surface mesh deformation method for handling wing-fuselage intersections," *Chinese J. Aeronaut.*, vol. 30, no. 1, pp. 264–273, Feb. 2017, doi: 10.1016/J.CJA.2016.12.026.
- [23] O. Mykhaskiv, P. Mohanamurthy, ... J. M.-35th A. A., and undefined 2017, "CAD-based shape optimisation of the NASA CRM wing-body intersection using differentiated CAD-kernel," *arc.aiaa.org*.
- [24] M. Schwalbach, T. Verstraete, J. D. Müller, and N. Gauger, "A comparative study of two different CAD-based mesh deformation methods for structural shape optimization," in *Computational Methods in Applied Sciences*, vol. 49, 2019, pp. 47–60.
- [25] Ansys, "ICEM CFD." [Online]. Available: <http://resource.ansys.com/Products/Other+Products/ANSYS+ICEM+CFD>.
- [26] A. H. Truong, D. W. Zingg, and R. Haimes, "Surface Mesh Movement Algorithm for Computer-Aided-Design-Based Aerodynamic Shape Optimization," *AIAA J.*, vol. 54, no. 2, pp. 542–556, 2016, doi: 10.2514/1.j054295.
- [27] R. Haimes and G. J. Follen, "{CAPRI}: Computational Analysis Programming Interface, a Solid Modeling Based Infrastructure for Engineering Analysis and Design.," in *6th Numerical Grid Generation in Computational Field Simulation*, 1998.
- [28] W. Wang, H. Fan, and G. Xi, "An extension of advancing front technique on new target surface after virtual topology operations," *Adv. Eng. Softw.*, vol. 124, pp. 42–52, 2018.
- [29] C. M. Tierney, L. Sun, T. T. Robinson, and C. G. Armstrong, "Using virtual topology operations to generate analysis topology," *Comput. Des.*, vol. 85, pp. 154–167, 2017.
- [30] "MySQL." [Online]. Available: <https://www.mysql.com/>.
- [31] M. S. Floater and K. Hormann, "Surface Parameterization: a Tutorial and Survey," in *Advances in Multiresolution for Geometric Modelling*, Berlin/Heidelberg: Springer-Verlag, 2005, pp. 157–186.
- [32] A. Sheffer, E. Praun, and K. Rose, "Mesh Parameterization Methods and Their Applications," *Found. Trends® Comput. Graph. Vis.*, vol. 2, no. 2, pp. 105–171, 2006, doi: 10.1561/06000000011.
- [33] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler, "A Local/Global Approach to Mesh Parameterization," *Comput. Graph. Forum*, vol. 27, no. 5, pp. 1495–1504, Jul. 2008, doi: 10.1111/j.1467-8659.2008.01290.x.
- [34] TranscendData, "CADfix." [Online]. Available: <https://www.iti-global.com/cadfix>.

- [35] S. H. Whyman, M. R. Gammon, and C. D. Fellows, “Enabling virtual topology for high quality CFD surface meshing of complex CAD geometry,” in *2018 Fluid Dynamics Conference*, 2018, p. 3721, doi: 10.2514/6.2018-3721.
- [36] M. Sabin, “Criteria for comparison of automatic mesh generation methods,” *Adv. Eng. Softw. Work.*, vol. 13, no. 5–6, pp. 220–225, 1991.