Computer-Aided Design 146 (2022) 103232

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Application paper

A finite-element based mesh morphing approach for surface meshes

Felix Claus [a,*], Bernd Hamann [b], Hans Hagen [a]

[a] TU Kaiserslautern, Computergraphics & HCI Lab, Erwin-Schrödinger-Straße 52, 67663 Kaiserslautern, Germany
[b] Department of Computer Science, University of California, Davis, CA 95616-8562, USA

ARTICLE INFO

Article history:
Received 30 August 2021
Received in revised form 24 January 2022
Accepted 1 February 2022

Keywords:
Digital twin
Mesh morphing
Sheet metal simulation

ABSTRACT

We present a finite element (FE) approach that deforms a given meshed CAD-based simulation model to a shape represented by a triangulation. We use an FE solver to calculate a smooth deformation field that we apply to the simulation mesh. The FE load case derives displacement boundaries from computed distance estimates between source and target meshes. We reduce mesh distortions via an iterative approach until a specified required mesh quality threshold is achieved. Our specific application is concerned with meshes used for sheet metal simulations arising in automotive applications where one wants to construct digital twins of measured sheet metal parts or entire assemblies. The approach is validated for parts and assemblies, considering simulation as well as experimental data. Our computational experiments produce errors below ± 0.05 mm, which is on the order of measurement uncertainty of common optical measurement devices. We provide test results for fully and partially measured parts to document the robustness of our implementation.

1. Introduction

Today's production processes use computer simulations concerning multiple manufacturing levels for optimization or prediction purposes. In contrast to physical testing, simulations are cheaper and more economical for process optimization. Due to the affordability of powerful hardware, simulation models have become sufficiently suitable to deal with real-world complexity, i.e., they are now capable of computing high-fidelity simulation models or supporting processes in real-time [1]. To make simulations more realistic sensor data from a real-world process environment can be integrated. This kind of simulation model is commonly called "digital twin", because the simulation model is improved by using sensor data from the actual process. The concept of a digital twin has recently become very popular and can be applied to various problems; a review can be found in [2]. For instance, in the automotive industry assembly processes of sheet metals are optimized using realistic finite-element (FE) simulation models. The challenge for proper processes modeling is the fact that the desired geometrical tolerances of the final assembly are on a smaller scale than the geometric deviations of single assembly components. Observed deviations of single components are generally caused by production uncertainties of previous production steps. To achieve acceptable assembly quality, every assembly needs to be optimized individually, based on the distortions of the components being assembled. To make high-fidelity predictions for process parameters, or to derive countermeasures, a digital twin preserving the real geometry of assembly components can be used, see [3,4].

To capture the free-form surfaces of assembly components with high resolution, optical measurement devices are necessary to acquire a point cloud representation. From the acquired point clouds, geometric information is obtained to derive a digital twin that is used to predict optimal assembly parameters. Converting the geometric information captured via a point cloud to the format required by a numerical simulation is a challenging and crucial process. This process is often done manually by using reverse engineering solutions, or the model is drastically abstracted, i.e., only offsets to mechanical boundaries are applied. Several methods can be used to create a simulation mesh based on a given point cloud, which, unfortunately, is often incomplete or noisy, making manual corrections inevitable. We introduce an automated FE-based mesh morphing algorithm that can handle measurement noise and incomplete measurements. A high-level description of the workflow is shown in Fig. 1. A tessellated point cloud and an FE model, established via a CAD definition of the "nominal geometry", are the inputs. The "nominal geometry" describes the desired geometry that is defined by a CAD model and technical drawings. The FE model is composed of FE meshes, the definition of element types, the definition of material properties, the interconnection between single meshed components, and the definition of load case. The tessellation of the point cloud is the representation of an actual geometry that differs from the "nominal geometry". Our method transforms the FE mesh to match the measurement data and produces a morphed FE model as output.

* Corresponding author.
E-mail address: claus@cs.uni-kl.de (F. Claus).

https://doi.org/10.1016/j.cad.2022.103232
0010-4485/© 2022 Elsevier Ltd. All rights reserved.

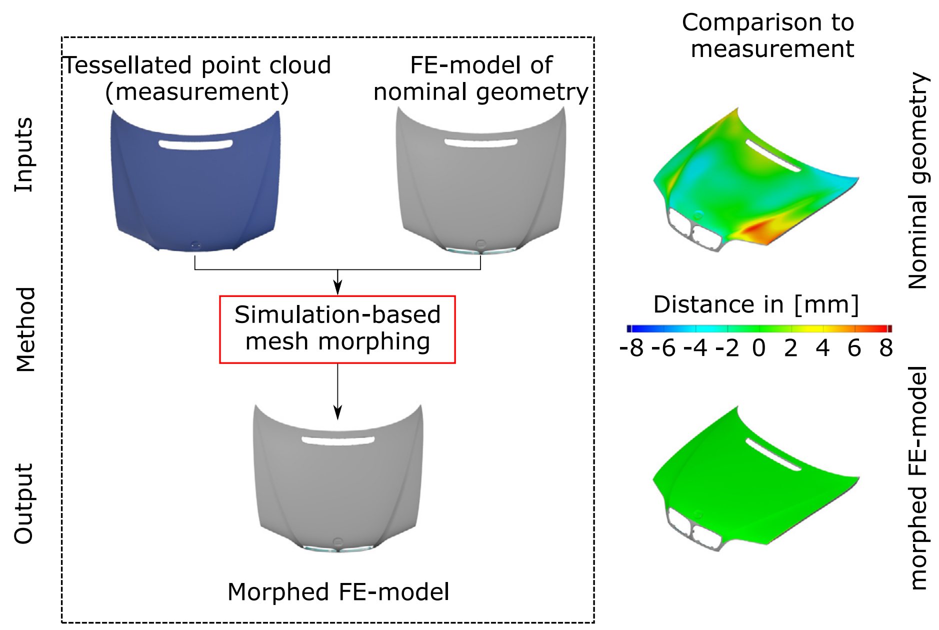

Inputs
Method
Output
Tessellated point cloud (measurement)
FE-model of nominal geometry
Simulation-based mesh morphing
Morphed FE-model
Comparison to measurement
Nominal geometry
Distance in [mm]
-8 -6 -4 -2 0 2 4 6 8
morphed FE-model

Fig. 1. Workflow for BMW engine hood. A tessellated point cloud and an FE model generated for the "nominal CAD geometry" serve as input. The output is a morphed FE model. Right: distances between FE model and measurement before and after morphing. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The color-coded pictures on the right of Fig. 1 show a comparison between measurement and FE model, before and after morphing. Our goal is to approximate a target geometry within ±0.05 mm which is the measurement uncertainty for commonly used 3D scanning systems.

2. Background

This section summarizes topics that are related to our method or motivate the need for solution approaches. The main challenge faced by methods covered in the literature is the need to handle real geometries acquired by optical measurement devices for establishing precise prediction models. We categorize existing approaches into those based on (1) reverse Engineering, (2) direct meshing, or (3) morphing.

2.1. Reverse engineering

An intuitive approach for generating a model of a measured shape utilizes reverse engineering (RE) methods to obtain a CAD representation of the acquired point cloud. Based on the CAD model, a simulation mesh can be generated that preserves measured geometry. By modeling the simulation model, based on the CAD representation obtained with RE, a geometric digital twin can be established. RE is used for many different application areas, e.g., manufacturing [5,6], medicine [7,8], civil engineering [9,10], and design processes [11]. A review is provided in [12]. RE can be time-consuming and costly. To speed up this process, dynamic approaches, e.g., the one described in [13], focus on optimizing data acquisition and reconstruction at the same time. In [14] multi-sensor data fusion is utilized to increase speed and precision. Although RE methods are well-established for generating virtual representations of complex free-form surfaces, additional model generation steps are often necessary to obtain the desired quality of a digital twin. Unfortunately, this model generation approach requires manual involvement. Thus, the pipeline from acquiring point cloud data to digital twin generation becomes costly and error-prone.

2.2. Direct mesh generation

To eliminate the need for RE, direct mesh generation methods produce FE meshes directly from a point cloud. This can be a challenging task, see [15]. The method discussed in [16] addresses this problem by coarsening an acquired point cloud using bubble packing [17]. A coarsened point cloud is triangulated to define the required FE shell elements. The resulting FE model is used to predict post-assembly shapes. A voxel-based approach for direct mesh generation is described in [18], capable of generating high-quality quadrilateral (quad) surface meshes from the acquired data. Although generating meshes directly from measured point clouds is beneficial, it does not directly address the problem of handling geometry that has not been captured entirely.

2.3. Morphing

To change the shape of digitized geometry in a smooth and controlled manner, morphing can be used, see [19]. Morphing is relevant for our approach as it can be used for transforming geometry in such a way that it matches a target shape. Morphing methods perform (1) CAD data-based morphing or (2) mesh-based morphing. CAD data-based morphing algorithms apply a transformation to CAD surfaces. The topology and number of faces of the CAD model remain the same during morphing. As mesh and modeling rules for automated model generation workflows are commonly defined together with a CAD definition, the update of the simulation model can be performed efficiently and robustly. This principle can be used for different applications, i.e. see, [20,21]. CAD data-based morphing has also been used to generate a digital twin from measured geometries. In [22], a parametric model for compressor blades is described where the model is obtained by fitting scanned data. The model was used to evaluate the impact of manufacturing variability on the performance of a multi-stage high-pressure compressor [23].

In contrast, a mesh-based morphing method calculates a deformation field that is applied directly to the vertex positions of a simulation mesh. This method does not require any re-meshing or re-modeling steps. Typically, mesh-based morphing is used to make rapid changes in a simulation mesh in the design phase, see [24]. In [25] mesh morphing is used to generate simulation models with shape deviations based on CMM measurements. A summary of mesh morphing algorithms is given in [26] and in [27]. A widely used branch of methods for mesh-based morphing approaches is called "spring analogy model". While the spring analogy model uses mathematical formulations for calculating a deformation field, FE-based methods solve the linear elasticity equations for calculating the vertex displacements [28]. Our approach is closely related to this branch of research as it classifies as an FE-based mesh morphing method.

When approximating a free-form surface discretely with a set of linear elements, deviations occur in areas with high curvature (beaded edges). To improve representation in these areas, adaptive mesh sizes are used. This can lead to an excessive amount of elements resulting in large computational costs for solving FE problems on the generated mesh. To reduce the number of elements while still preserving a highly accurate representation of continuous geometries, higher order-meshing can be employed. During the generation of curvilinear meshes, morphing is used to shape a piece-wise linear approximation to "resemble" more and more a higher-order representation, see [29]. In the following, we only use linear mesh elements. However, approximation errors caused by linear elements are discussed, and curvilinear meshes could be used to improve achieved results.

Based on the reviewed literature, we identified the following technological gap. Although the generation of CAD surfaces or meshes from acquired point clouds can be considered as a solved problem – solved satisfactorily by RE or direct mesh generation methods – it still requires manual interventions. Especially, when parts cannot entirely be captured by scanning, significant manual work is necessary. Finalizing a simulation model from an automatically generated CAD representation or mesh minimally requires manual quality control of the model. CAD-based morphing approaches are more robust in terms of automated model generation, as meshing and modeling can be updated efficiently and reliably. However, these approaches are limited in terms of precision and flexibility, especially when complex geometries are involved. While mesh-based morphing provides the necessary flexibility, available methods either cannot be used to match measurement data or have problems maintaining mesh quality when morphing complex geometries. We address this technology gap via an iterative simulation-based mesh morphing algorithm. Our algorithm is designed for surface meshes and applied to use cases from the automotive industry. Our contributions are the following:

- An iterative FE simulation-based mesh morphing algorithm that handles complex geometries, measurement errors, and only partially measured geometries.
- Real-world validations with simulation and experimental data, showing approximation errors below ± 0.05 mm.

The novelty of this work is the high degree of automation of our approach. In contrast to available approaches where substantial manual work is necessary to achieve a geometrical digital twin derived from 3D scans, our approach only needs to be set up once per part.

3. Method

Our approach falls in the class of FE-based mesh morphing methods. As kernel for the displacement field calculation, we use an FE simulation. The definition of the simulation problem is crucial for describing the method. First, a high-level description of the approach is given. An overview of the processing steps is presented by discussing a pseudo code, see Algorithm 1. We also discuss the mathematical details of the various functions involved.

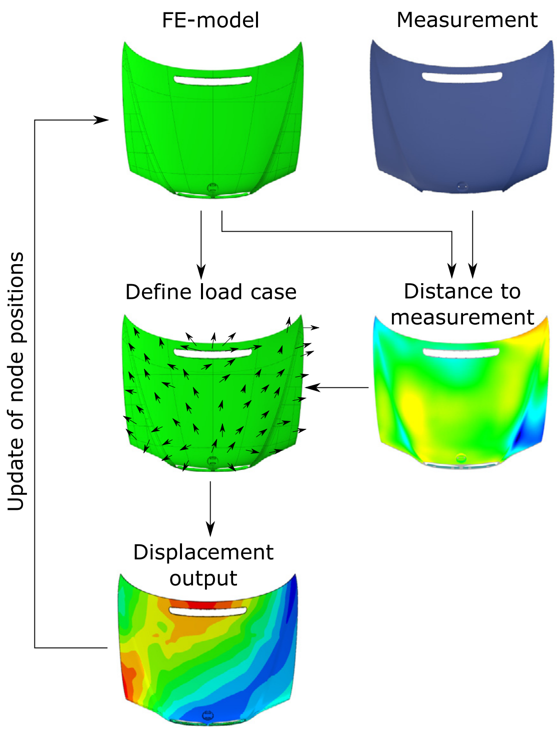


Fig. 2. Computational progression of the proposed algorithm. The input of the algorithm consists of an FE-mesh based on the nominal geometry and tessellated measurement data of the actual part. A distance field is computed establishing a load case, subsequently applied to the FE model. The output updates the FE-model. The algorithm terminates when FE-model and measurement geometries match.

3.1. *High level description*

Fig. 2 provides a high-level overview of the computational scheme. The basic idea of the FE-based morphing approach is to calculate distances between FE mesh nodes and a measured target geometry; these distances define an FE-load case. The displacement of the simulation output is used to adjust node positions of the FE-model. This process is repeated until the geometries of FE-model and measurement match. Most crucial is the calculation of the distance field and the placement of FE-boundary conditions. Fig. 3 shows how the placement of FE-boundaries is carried out, based on a calculated distance field. Following common meshing terminology, the phrases "source" and "target" are introduced to differentiate between meshes, finite elements, and vertices belonging to different inputs. For the use case, the source mesh is related to the FE mesh representation of the "nominal geometry", while the target mesh is a tessellation of the point cloud of the measured, actual geometry. Based on the calculated distances, FE boundary conditions (displacements) are associated with the FE mesh. We divide the boundary conditions into two groups: (a) restrictive boundaries with prescribed movement conditions in all three dimensions, and (b) boundaries considered in node normal direction only, allowing in-plane movement. In Fig. 3, restrictive boundaries (blue) are used for nodes belonging to mesh perimeters. The zone marked in gray is a row of elements next to the mesh perimeter. To prevent collision of restrictive perimeter boundaries and boundaries applied only in normal direction, this zone is introduced. For mesh nodes belonging to this zone, the placement of boundary conditions is restricted to perimeter boundaries only.

When the simulation problem is set up, an FE solver can calculate nodal displacements, bringing the FE mesh closer to

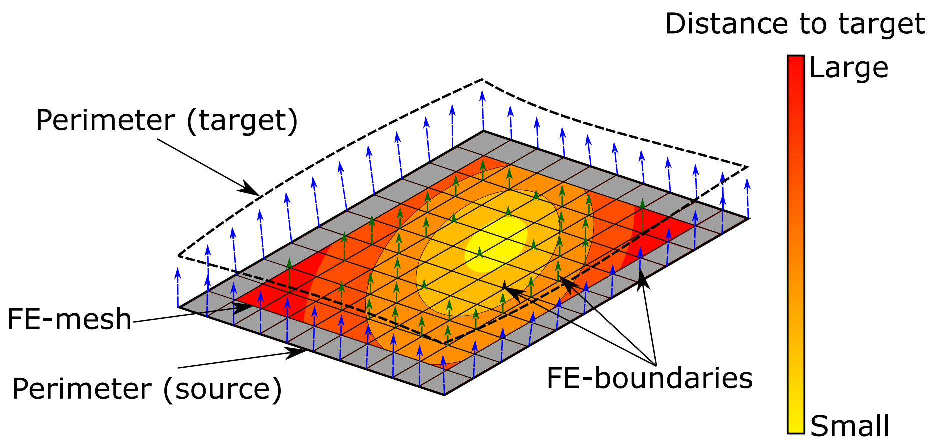


Fig. 3. High-level illustration showing different types of boundaries applied to the FE-mesh which is used for morphing. Types of boundaries differentiated by (blue) perimeter vector boundaries and (green) displacement boundaries applied only in surface normal direction. The magnitude of displacements is calculated based on the local distance to target geometry. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the measurement data, thereby reducing distance. This step is repeated until the distance value satisfies a specified minimum threshold value.

3.2. Algorithm

Algorithm 1 Outline of the morphing algorithm. Input: measured point cloud and simulation mesh; Output: morphed simulation mesh.

```
 1: Data loading
 2: $X \leftarrow$ Target mesh          ▷ Tessellated point Cloud as trimesh-obj
 3: $Y \leftarrow$ Source mesh          ▷ Tessellated FE mesh as trimesh-obj
 4: Parameter
 5: $\epsilon \leftarrow$ Termination criterion
 6: max-angle $\leftarrow$ Maximal normal angle difference
 7: weight_function(a) $\leftarrow$ Magnitude weight function
 8: c_params $\leftarrow$ Coarsening parameters
 9: Pre-processing
10: perims=extract_and_match_perimeters($X$ , $Y$)
11: disp_perims=register(perims)          ▷ perimeter displacement vectors
12: disp=disp*weight_function(mag(disp)) for disp in disp_perims
13: Main loop
14: while $\epsilon < error$ do :
15:     magnitudes=[ ]
16:     for all $vert$ in $Y$.vertices do:
17:         $X_{tri}$, distance_vector = find closest triangle(vert , X)
18:         angle = angle(vert.normal , $X_{tri}$.normal)
19:         if angle > max-angle then skip          ▷ Filter out bad matches
20:         $b$ = $\langle$ distance_vector , vert.normal $\rangle$
21:         magnitudes.append(weight_function($b$))
22:     end for
23:     magnitudes = coarsening(magnitudes , c_params)
24:     write_solver_deck($Y$ , magnitudes , disp_perims)
25:     run_FE_solver()
26:     update_verticies($Y$)
27:     error=RMS($X$ , $Y$)
28: end while
29: Output: write_out_morphed_model($Y$)
```

In the following, the algorithm implementation is described using the pseudocode Algorithm 1.

As input, two Trimesh [30] objects must be created — lines (2) and (3). While 3D scans usually are handled as tessellated data and can be directly loaded via Trimesh, the simulation mesh of the "nominal geometry", needs to be converted. Therefore, nodes and elements are parsed from the solver deck from which a Trimesh object can be created. In lines (5)–(8), the parameters that can be defined by the user are listed. These are: *(5) termination criterion (ϵ)* – floating-point value that is compared each iteration with the Root-Mean-Square(RMS) error value – see Eq. (1), *(6) max-angle* — floating-point value that represents the maximum acceptable angle between normal of source mesh node and matched triangle of target mesh during distance computation, *(7) weight function* — a continuous function that returns a weighted value for a given displacement magnitude, *(8) coarsening parameters* — a set of parameters for coarsening a list of displacements. The first calculation steps are performed during pre-processing. In line (10), the perimeters of both input meshes are extracted and matched. A perimeter is considered a topological boundary of the geometry. Next, the extracted perimeters are parameterized and up-sampled, if necessary, and the point set registration problem is solved in line (11), resulting in a target displacement vector for each perimeter node on the source mesh. The target displacement vectors are scaled by the *weight_function* defined in line (7). Matching perimeters and calculating target displacements are optional and can only be applied if the perimeter curves are represented accurately in the target mesh.

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} d_i^2}{n}} \tag{1}$$

Eq. (1): Calculating RMS error. Distance d_i is the distance error per node and n the number of nodes.

The main loop of the algorithm iterates until the RMS error is below ϵ – line (14) – which is calculated at the end of each iteration — line (27). In the body of the main loop the following steps are performed: For each vertex of the source mesh, the nearest triangle of the target mesh is searched — line (16). This function in native, provided by the library Trimesh, returns the closest point, distance as a scalar, and closest triangle-ID. Next, the vertex normal of the source mesh is compared to the identified triangle normal. If the angle between those normals exceeds the user-defined value *max-angle* (6) the pair will be considered a mismatch and will not be considered further in the current loop — line (18). If the difference in normal direction is within the allowable tolerance a, the displacement magnitude for FE computations will be calculated in two steps. First, the scalar product between the distance vector and normal vector of the source mesh vertex is computed in line (20). Computing the scalar product is done to obtain only the portion of displacement in normal direction. Second, the obtained scalar value b is weighted by the weight function defined in line (7). The weighted scalar value is stored in a list — line (21). Before performing the FE computation, the list of magnitudes is coarsened – line (23) – to ensure that one does not apply boundary conditions to every single vertex. The coarsening function thresholds the list and selects every nth element of remaining entries. The coarsened list of displacements, perimeter displacements, and current mesh node positions are used to generate a solver deck. We note that the values held by the list *magnitudes* are scalar values while *disp_perims* are displacement vectors. Both are used for generating displacement boundary conditions. The difference is that the scalar values from *magnitudes* are used to displace the corresponding mesh node in normal direction while the in-plane degrees of freedom (DOFs) remain unrestricted so the mesh can move in-plane. In contrast, the vectors of *disp_perims* are used to move corresponding nodes along a pre-defined vector by defining all three translational DOFs. When boundaries and vertex positions are written to the new solver deck, the FE computation can be performed. The FE problem is defined as a mechanical linear elastic problem, see Eq. (2), which is solved by the simulation software. The resulting displacement field is used to update the vertex positions of Y, used to evaluate an RMS error by comparing with X. At this point, one iteration has been performed. The steps line (15)–(27) are repeated until the RMS error is below ϵ. The final mesh is written to an output file.

$$0 = K\vec{u} + \vec{f} \tag{2}$$

Eq. (2): Equilibrium formulation of linear elastic problem. The matrix K is the stiffness matrix of the mesh, $\vec{u}$ are nodal displacements, and $\vec{f}$ are nodal forces.

3.3. Mathematical description of calculation steps

Important mathematical functions used in the pseudo code Algorithm 1 are explained in more depth. In particular, these are: (a) perimeter matching and point set registration — line (10)/(11); (b) distance computations and filtering — line (17)–(21); (c) user-defined weight function — line (7); (d) coarsening the calculated distances with user defined parameters — line (8) and (23); and (e) handling edge cases — not mentioned in the pseudo code.

Matching perimeters and point set registration — In analogy to mass–spring models, the morphing approach matches the mesh perimeters and applies strict displacement conditions. The nodes on perimeters can be identified easily and matched robustly. This property is used to prevent overlapping of the mesh perimeters in the final result. First, all target and source mesh element edges that belong only to one element are extracted from both meshes. Next, resulting edges are grouped by connectivity. Each group of connected edges is a mesh representation of a perimeter. To eliminate mesh influence each perimeter is re-sampled. Re-sampling is carried out by parametrizing each edge-list of perimeters by scaling accumulated edge lengths to one. Equidistant sampling is performed, generating new nodes by linear interpolation of corresponding edges. For each re-sampled perimeter, the characteristics length (l) and center of mass (c) are calculated, see Fig. 4. Based on these calculated characteristics, perimeters of source and target mesh are matched. After the matching of perimeters is done, a point set registration is performed to find target displacement vectors for each vertex on the perimeters of the source mesh. The used registration method is called "coherent point drift" (CPD) [31], implemented in the library "Probreg" [32]. CPD is a state-of-the-art non-rigid point set registration method. CPD solves the point set registration problem by viewing it as a probability density estimation problem to which a Gaussian mixture model (GMM) is applied. The GMM centroids of the source point cloud are fitted to the target point cloud while the movement of GMM centroids is restricted to be coherent to maintain topological structure. The original paper provides more details. CPD can handle outliers and noisy data very well, which is important for our problem. Fig. 5 shows a simple example for registering two matched perimeters that are discretized by vertices. By applying the transformation calculated with CPD to the source point cloud, the transformed source vertices are obtained. While the source point cloud preserves the original vertices from its mesh, the target point cloud is a dense equidistant re-sampling of the parameterized perimeter curve. The re-sampling of the target perimeter is done to achieve the best possible representation of the target curve, as the information of edges connecting the points or order of points is not used by the CPD algorithm. In contrast, the source point cloud is not re-sampled since we are interested in a target displacement vector for exact these vertices. If necessary, additional nodes on the perimeter edges of the source mesh could be interpolated for the registration step. However, this was not necessary for the use cases covered in this paper, and we did not implement this step. The advantage of using a point cloud representation for registering the nodes of matched perimeters is that orientation, the starting point of the curve, or distribution of vertices along the curve do not matter. The output of CPD provides the desired transformation of the source points. Based on this transformation, the desired displacement vectors are computed and are used to define nodal displacement boundary conditions on the perimeter vertices of the FE mesh (source).

Distance computation and filtering — The distance computation for the vertices not contained in the perimeters is carried out by using a proximity function implemented in Trimesh. This implementation also does resolve ambiguous distances internally. As input, this function needs a mesh object (target) and a list of vertices (source). For each source vertex, the closest triangle of target mesh, the point on this triangle, and distance as scalar are returned. As we are interested in distances between the FE mesh vertices and the measurement, we use the tessellated measurement as target input mesh and the vertices of the FE mesh as source input points for distance calculations. On the left side in Fig. 6, the components of the distance computation are visualized. Just finding the triangle closest to a vertex might result in mismatches for various cases. For example, incomplete meshes or large in-plane shifts between meshes lead to wrong matches that cause errors when deriving boundaries for the subsequent FE computations. To handle mismatches, two filters are used, see right side of Fig. 6. First, the normal vector of the source vertex is compared to the normal of the matched triangle. If the angle between the two normal vectors is above the user-defined threshold – line (6) – the vertex will not be considered for applying displacement boundary. To deal with large in-plane shifts, a second filter is applied. This second filter calculates the scalar product between the distance vector and source vertex normal vector. As every normal vector is automatically normalized by Trimesh, this scalar product can be geometrically interpreted as the component of the distance vector in vertex normal direction. The reason for only considering distances in vertex normal direction is this: The displacement boundaries for the FE problem are only applied perpendicularly to the mesh surface. All other DOFs are unrestricted. Thus, the mesh can slide during FE analysis, automatically resolving in-plane shifts.

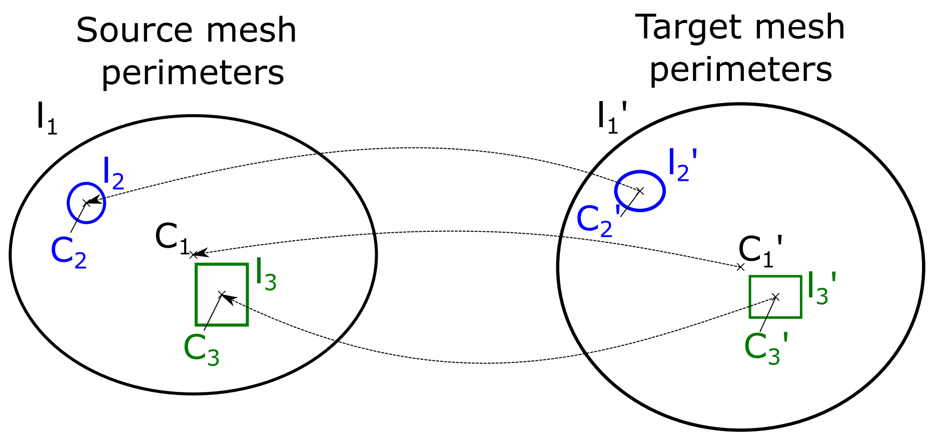


Fig. 4. Matching perimeters (1–3) of source and target meshes. Perimeters with different shapes, lengths, and locations are matched using as characteristics center of mass (C) and length (l) for comparison.

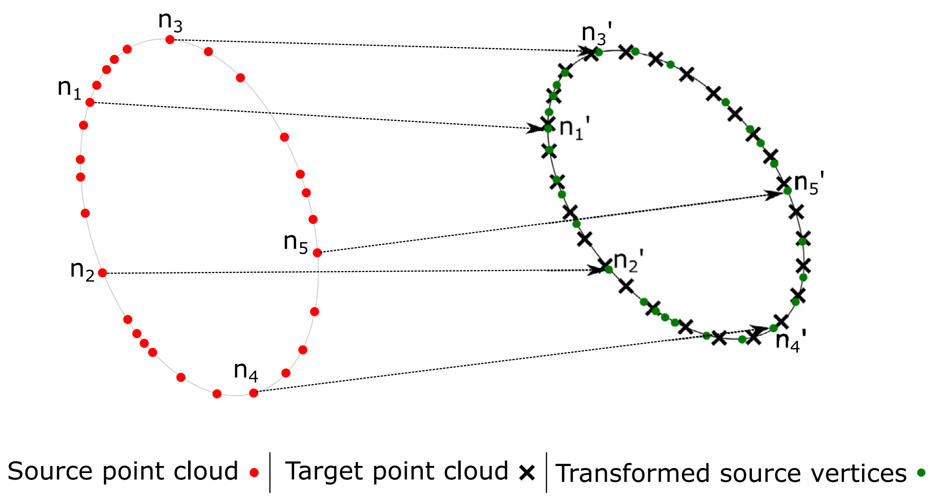


Fig. 5. Point set registration using non-rigid CPD algorithm. This algorithm does not consider the order of vertices or orientation of the perimeter curve.

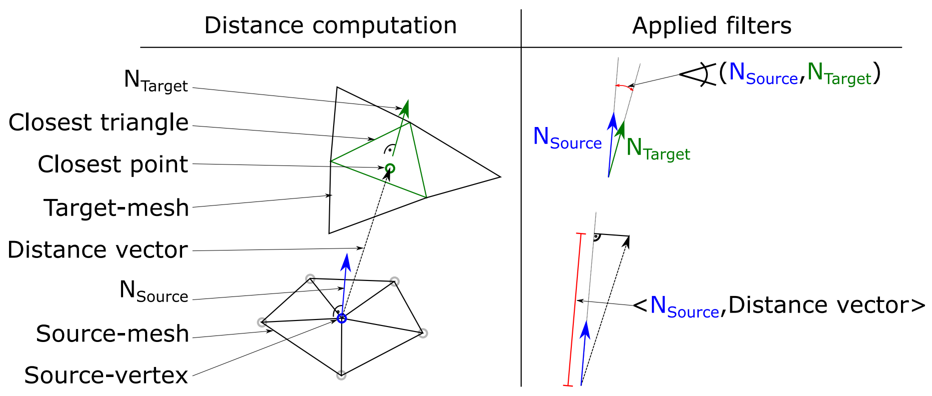


Fig. 6. Left: distance computation used for finding distances for each source mesh vertex. Right: filters used to remove mismatches (normal-filter), and handle in-plane shifts (scalar product).

User-defined weight function — To prevent mesh distortions caused by largely displaced mesh nodes during the FE calculations, a weight function is applied to the magnitude of calculated distances. This approach generates a smooth displacement field with smaller displacement magnitudes. Between iterations, the distance to target geometry is re-evaluated and updated. The function used in our method is plotted in Fig. 7. For small distance magnitudes, the function weights magnitude of the corresponding displacement boundary with a factor near 1. For large distances, the magnitude gets weighted down, but never towards zero as this would pin down single mesh nodes resulting in slow convergence of shape optimization problems. For different applications, the weight function needs to be scaled according to element size. The weight function could also be used to accelerate convergence as this function has a direct impact on

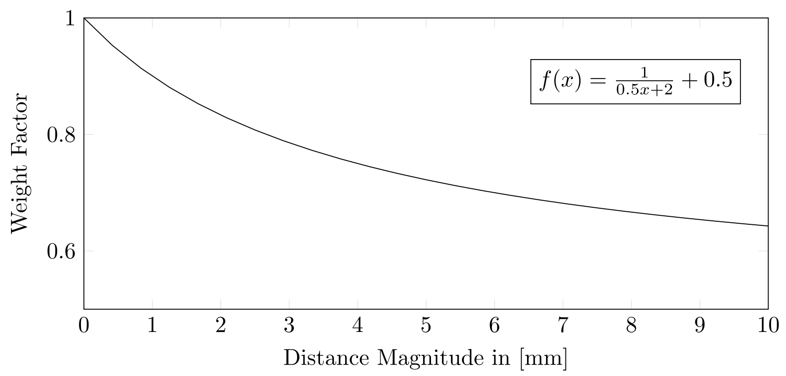


Fig. 7. Weight function $f(x)$ for scaling matched distance values. Small distances have weights close to 1, while large distances have weights of approximately 0.5.

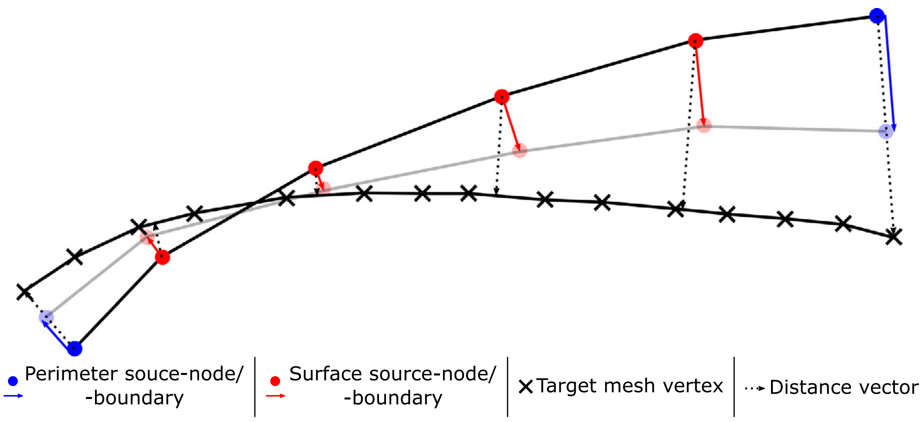

Perimeter souce-node/ -boundary
Surface source-node/ -boundary
Target mesh vertex
Distance vector

Fig. 8. One iteration step with weighted boundaries. Blue: displacement vectors of perimeter mesh vertices. Red: displacement applied to mesh nodes in normal direction. Transparent: updated mesh node positions after FE simulation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the choice of either more or less aggressive displacement magnitudes. The effect of applying such a weight function to the calculated displacement magnitudes is depicted in Fig. 8. In the figure, the source mesh is considered as FE simulation mesh, to which boundary conditions are applied, while the target mesh corresponds to a tessellated representation of the actual measured geometry. The length of each displacement boundary is scaled by the weighting function. The boundaries placed on the perimeters are shown in blue. These boundaries have a predescribed vector calculated by the point set registration. The red boundaries are displacements for the source mesh nodes that do not belong to perimeters. These boundaries only displace the corresponding mesh node in normal direction during FE calculations. This is achieved via the second applied filter, see Fig. 6, and is done so the node can slide perpendicular to the surface normal. This sliding has the effect, that the mesh can relax to avoid distortions. After each iteration, the mesh node positions are updated, shown via reduced opacity. The new shape is closer to the target mesh. Due to the mesh update, vertex normal vectors change as well. This leads to fewer mismatches when filtering distances in the next iteration. As matches and normal directions do change between iterations, the weighting function does define how aggressively the FE computation changes the mesh.

Coarsening matched distances — Before defining boundary conditions from the calculated and filtered distances, the identified matches are thresholded and coarsened. For thresholding and coarsening, different user-defined values can be set, considered as a set of coarsening parameters in line (8) of Algorithm 1. First, all distance values are thresholded by a certain value. This is done to only consider mesh nodes that are in a defined range of the target surface. Thus, mesh nodes far away from the target mesh are not considered until they move close enough due to previous iterations. Next, a margin can be defined by a number of rows. This number defines the number of element rows next to perimeters that must not have any boundary conditions, compare Fig. 3. As the perimeters have restrictive displacement conditions, additional boundaries near perimeters can cause mesh distortions, especially during the first iterations. This value can be set to 0 towards the last iterations. All remaining values that are not classified as a mismatch by the distance computations or removed due to proximity to perimeter mesh nodes are clustered using three classes (a,b,c). For each class, different density parameters for boundary placements are defined. The purpose of this classification and varying densities for boundary placement is illustrated in Fig. 9: Class (a), nodes that are already inside the target range. These nodes are kept in position (only normal direction restricted) — displacement value is set to zero; Class (b), nodes that are used mainly to pull the source mesh towards the measurement during the current iteration; Class (c) remaining nodes below the threshold, but outside class (a) and (b). For each class, distance value ranges and a coarsening number n must be defined by the user. Coarsening for each class is performed by selecting every nth node for placing a boundary condition. For class (a) and (c), a higher value, e.g. $n = 5$, can be selected. The rationale is that in class (a) almost no deformations occur. Thus, only a few boundary conditions must hold for the source nodes near the target surface. In class (b), most of the deformations take place. Therefore, dense sampling of boundaries is chosen, i.e., $n = 2$. Class (c) is used to guide the area adjacent to the deformation zone. As distance computations might be less robust and discontinuous in this class, only a few boundaries are placed. The reason

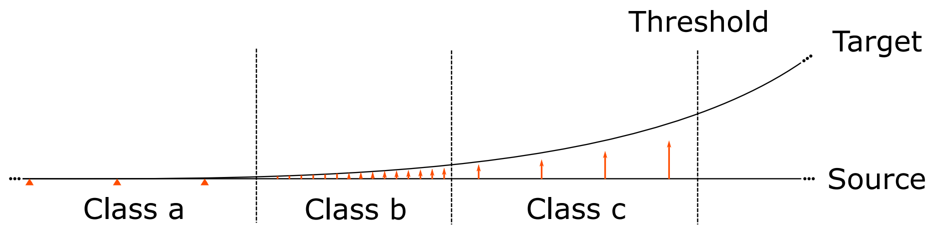


Fig. 9. Placement of weighted displacement boundaries, classified by distances into three classes: (a) close to target, held in position; (b) in deformation zone and dense placement of boundaries; (c) coarse placement to guide source mesh towards target.

for selecting only every *n*th node for placing a boundary is this: The simulation result becomes "smoothed" as the part's stiffness acts similarly to a cubic spline between restricted nodes. With each iteration, the classes shift. Class (a) permanently grows until the whole part is covered.

Edge cases — Although the proposed algorithm is reliable and robust, some edge cases should be considered to improve results. The phrase "edge case" is meant in the context of computer programming. One edge case occurs in areas with high surface curvature. In these areas, the comparison between node normal and mesh element normal leads to errors. These errors are caused by the node normal not matching an adjacent element normal. This leads to a "mismatch" classification when the difference in normal angle exceeds the specified threshold for filtering. Specifically, when calculating the scalar product, see Fig. 6, the resulting displacement vector is generally quite small. To compensate for such errors the calculation of the scalar product is skipped when the computed distance value is below local mesh resolution (maximum edge length of surrounding elements). The second edge case occurs when the target mesh does not cover the whole surface of the source mesh (incomplete measurements). In this case, perimeters might not be captured. Further, measurement values are often noisier near perimeters. To handle such problems, we remove rows near perimeters from the candidates for boundary placement. This applies to elements that are between class (c) and thresholded nodes, see Fig. 9. As all not-measured areas have large distances to the next triangle on the measurement, there exists a high degree of change of distance near the outline of the measurement. This property can be used to shrink the nodes where boundaries can be placed. The user can define how many rows of elements are discarded to adjust the algorithm to measurement quality.

3.4. Parameters used to adjust algorithm

The proposed algorithm is based on various parameters that can be used to improve morphing quality for individual use cases. This section introduces parameters that need to be defined by the user. Their impact on the morphing algorithm is discussed as well.

Search distance — This parameter defines the maximally allowed distance magnitude, calculated by trimesh, see, "Distance Computation and Filtering". All distance values above the search distance are not considered for placing boundary conditions. This value only thresholds distances for nodes that do not belong to a perimeter of the geometry. Varying this value has a significant impact on the number of boundary conditions being placed.

Normal angle — This parameter is used for the comparison of node normal (source) and triangle normal (target), determined with the distance computation. If the angle between node normal and triangle normal exceeds the parameter value, the match is classified as a mismatch and discarded, see Fig. 6. This parameter impacts the number of node candidates that are kept for boundary placement greatly.

Class (a–c) — This set of parameters controls the density of boundary placement for each class introduced in "Edge Cases". For each class, a range of distances as well as a number *n* must be defined. The distance range is used to generate a list that is a subset of all mesh nodes, where the calculated distance values are within the defined range. The number *n* is used to coarsen the list by selecting every *n*th element. By selecting different ranges and sampling densities, the main deformation zone can be controlled, compare Fig. 9.

Remove rows — This parameter is an integer that defines how many rows of elements next to perimeters must not contain boundary conditions, except perimeter boundaries. In Fig. 3, elements colored in gray show where this parameter is set to one. By increasing the number of element rows being excluded from boundary placement, the distance between the different kinds of boundary conditions is increased. In this zone (compare Fig. 3) no boundary conditions except perimeter boundaries are placed. Subsequently, nodal displacements only result from deformation calculated by the FE simulation.

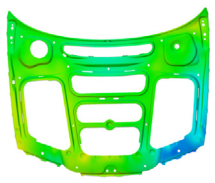
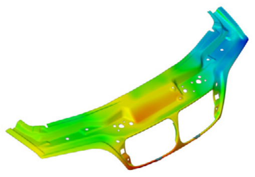
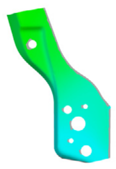
Relaxing parameters — This condition determines when an intermediate morphing step is close enough to the target geometry in order to relax parameters, which were listed before. For example, this condition could be an RMS error value, maximum error value, or fixed iteration steps. The idea is to use moderate parameter sets in the beginning and switch to more aggressive parameters when getting close to the final step. For instance, by increasing the "normal angle" criterion, lowering "remove rows" and increasing sampling density *n* for all classes, faster convergence, and higher geometric precision result. When applying such conditions too early, the FE mesh can become distorted, resulting in low-quality results. The choice of this condition is problem-dependent. However, to achieve the desired behavior we apply them as early as possible, i.e., when perimeter boundaries have reached their target position. In general, it would be also doable to introduce multiple relaxing criteria that allow a successive change from moderate parameter sets towards restrictive ones.

4. Case study

For verification and validation, three different case studies are presented. First, a verification with simulated input data is provided. Single sheet metal parts with different features and sizes are morphed with an artificially created geometry as target. The second case study investigates an assembly of an engine hood that is morphed, using only a target geometry for the outer skin. This study is also performed with simulated input data to be able to evaluate the morphing of the underlying structure. The third case study presents a validation where the method is applied to real, imperfect measurement data, generated via 3D scanning. Additional verification investigating surface defects and lateral displacements is provided in Appendix.

4.1. Case 1 — Verification: Morphing single sheet metal parts

Fig. 10 shows three different sheet metal parts. All parts belong to the sub-structure of an engine hood of a BMW E46. These

Part Feature	Part 1	Part 2	Part 3
# Mesh Elements	120381	40851	4039
Material	Steel	Steel	Steel
Thickness [mm]	0.7	0.7	2
# of Perimeters	51	27	7
Geometry & distance to measurement			

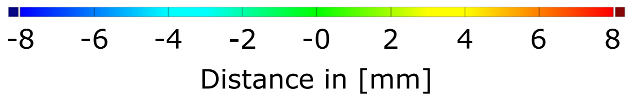


Fig. 10. Different input geometries for validation. Shown are deviations of up to 8 mm compared to target geometry. These parts were chosen to verify proper morphing of different geometrical features. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

parts were selected because they have different characteristics like size, number of perimeters, and geometric features. The parts are represented by an FE mesh with first-order shell elements. The material model was chosen as linear elastic with Young's modulus of 210 GPa and a Poisson's ratio of 0.3. The simulation software used is ABAQUS from Dassault Systèmes. In the figure, the distance between the FE mesh and the target geometry is shown, encoded in color in the pictures. The target geometry was created by performing an FE simulation deforming the original mesh. The simulation result was extracted and converted to a triangulated mesh acting as a pseudo-measurement. These FE-generated pseudo-measurements have ideal properties, like smoothness and completeness, compared to real measurement data. The verification is performed to determine whether the proposed method works with ideal input data for different kinds of geometries.

Execution parameter — The user-defined parameter values for execution are listed in Table 1. In addition to the computation steps discussed in Algorithm 1, adaptive parameters ("Relaxing Parameters") are employed. Per iteration, the maximum distance is determined and parameters are switched to more aggressive states when an already good approximation is achieved. This is done to increase convergence speed and handle unwanted blind spots that are filtered out when using moderate parameters. Finding suitable parameter values for a given geometry and measurement quality can be done by performing a parameter study. Performing a parameter study is time-consuming but is not necessary for every part. For most geometries, a well-chosen standard set of parameters is sufficient (e.g. parameter values for Part 2). Whether it is worth performing a parameter study can be determined by the complexity of the part and quality of measurement.

4.1.1. Results

The results of the verification are shown in Fig. 11. The first three iterations and the last iteration are shown. For this verification, the number of iterations was fixed to analyze convergence behavior and robustness of the method. After three iterations, no deviations above 0.2 mm can be observed. The ninth iteration is analyzed on the scale of ±0.05 mm, which is at the scale of measurement uncertainty of commonly used measurement devices for this application. Even at this small scale, only single elements show a noticeable difference which is explained by the mesh's resolution.

Table 1
Parameter sets for validation.

Parameter	Part 1	Part 2	Part 3
n Class a [0 mm, 0.05 mm]	3	3	3
n Class b]0.05 mm, 1 mm]	2	2	2
n Class c 1 mm<	2	2	2
Search distance	10 mm	10 mm	10 mm
Remove rows	2	2	1
Normal angle	10°	30°	30°
	Relaxing parameter		
Relax if *max dist* <	0.5 mm	2 mm	2 mm
Remove rows	0	0	0
Normal angle	30°	40°	40°

4.2. Case 2 — Verification: Morphing a partially scanned assembly

To determine whether the proposed method can handle partially scanned parts or even assemblies, the simulation model of the whole engine hood is morphed, only using the information of the outer skin that can be captured in an assembled situation. The morphing result of the inner structure is compared with the deformed target model. Again, the target geometry is created using an FE simulation deforming the "nominal model". We do not expect a high-precision reconstruction of the hidden areas but at least a plausible representation of the parts not captured in a 3D scan. Fig. 12 shows an exploded view of the assembly. The engine hood consists of seven individual sheet metal components with different thicknesses. Besides the visible outer skin, four reinforcement and two inner parts are components of the assembly. All parts are made from steel sheet metal and are modeled with 365 152 first-order shell elements. The components are joined together by modeling spot welds, acoustic and structural adhesives.

4.2.1. Results

The results of morphing the assembly only using geometrical information of the outer skin are shown in Fig. 13. The left side of the figure shows the mesh with "nominal geometry" compared to the deformed mesh created with an FE simulation. The deformed mesh is triangulated and exported, serving as a pseudo-measurement. Only the triangulated outer skin is used as input of the algorithm as target geometry. After morphing (right side), the outer skin is well-approximated, within ±0.05 mm. The underlying sub-structure is only deformed by interacting with

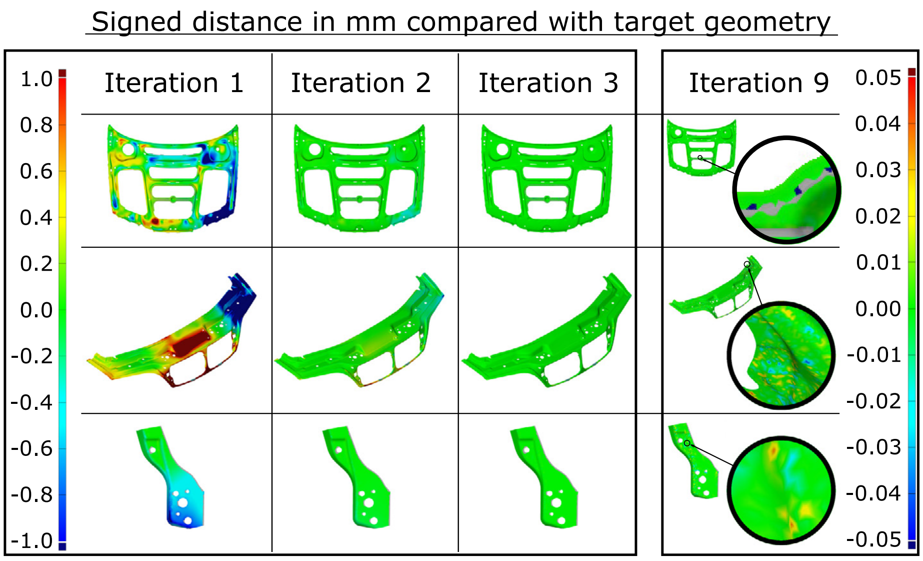


Fig. 11. Series of iterations for three parts. Color represents signed distance to the target geometry. Iterations 1–3 are measured at a scale of ±1 mm, and iteration 9 at a scale of ±0.05 mm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

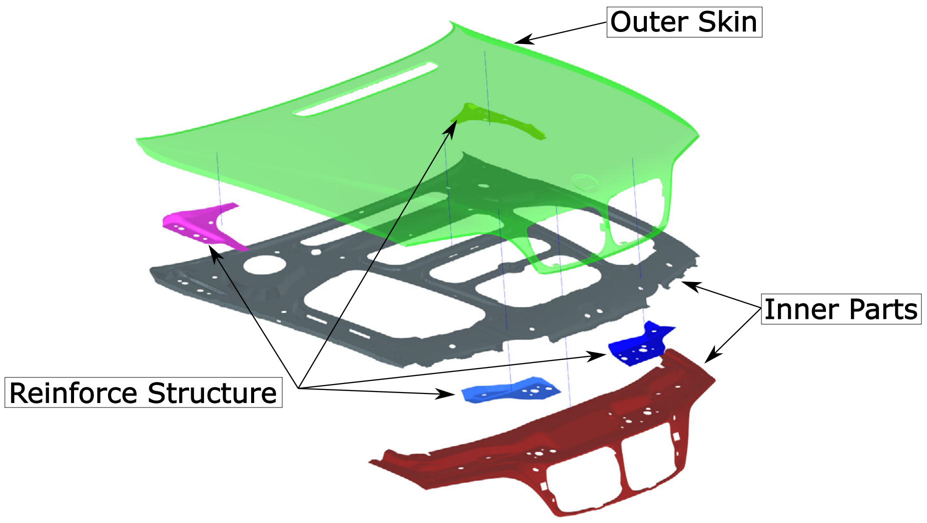


Fig. 12. (This figure was published in [33], licensed under Attribution 4.0 International (CC BY 4.0), see https://creativecommons.org/licenses/by/4.0/; changes: figure shows only one sub-figure.)

the outer skin during the FE simulation. The results for the substructure have errors of up to ±0.5 mm compared to the target geometry. A much better approximation is obtained compared to the situation before morphing.

4.3. Case 3 — Validation: Morphing with experimental data

To validate the performance of our method for real-world use cases, 3D scan data is used as input. We consider this validation as crucially important: When working with real sensor data, additional challenges are expected that do not arise when working with artificial test data. For acquiring the point cloud the experimental setup shown in Fig. 14 is used. The left side of the figure shows the part mounted on an aluminum profile frame using the original hinges and locks. To be able to adjust attachment positions, one hinge is mounted onto a compound slide. A dial gauge is used to tune the positions of locks and buffers. The frame is attached with markers used for aligning point clouds. To reduce reflection effects a light gray spray paint is applied to the engine hood. The right side of Fig. 14 shows the setup of the 3D scanner. An HP pro S3 structured light scanner is used to capture the whole part in one picture. To handle such a large scan window the scanner is set up using custom calibration panels and the hardware arrangement shown. The reason for capturing the whole part in one scan is the desire to avoid errors caused by point cloud registration. The downside of this scan setup is the fact that the acquired point clouds become noisy on an order of around ±0.1 mm. We consider a noisy scan as a challenge for the algorithm and argue that a higher-resolution measurement is more easily processed, which makes the chosen setup sufficient for validation purposes. The acquired point cloud consists of 4.9 million points, generated by merging multiple scans using different exposure times. Points with a high degree of uncertainty are automatically removed by the software. The point cloud is tessellated and smoothed, and unnecessary points are removed, resulting in a triangle mesh with 700,000 vertices. This number is about four times the density of the simulation mesh. The smoothing, coarsening, and tessellation steps applied to the scan data were performed with the commercial software

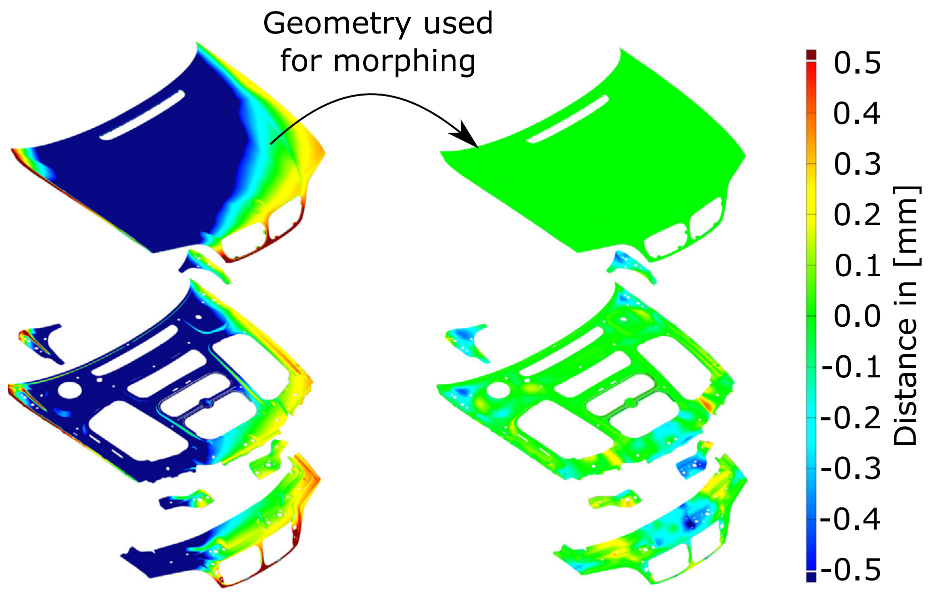


Fig. 13. Morphing full assembly only using target geometry for outer skin. Inner parts are morphed indirectly by interacting with the outer skin via FE simulation.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

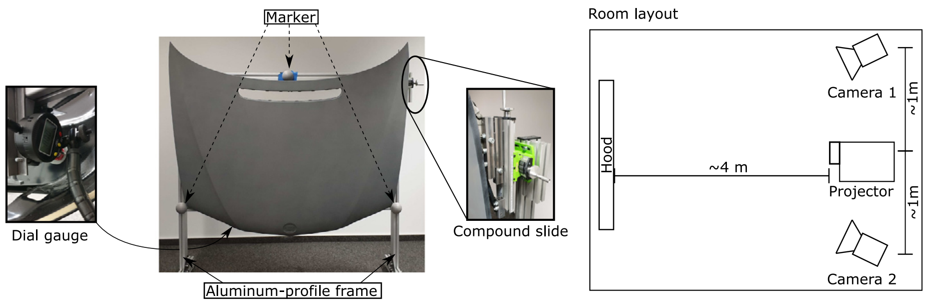


Fig. 14. Experimental setup. Left: fixture and specimen. Right: room layout for capturing the whole part in one picture. (This figure was published in [33], licensed under Attribution 4.0 International (CC BY 4.0), see https://creativecommons.org/licenses/by/4.0/, changes: spelling.)

tool "GOM Inspect", which is commonly used in the industry for post-processing scan data.

Before using the generated mesh as morphing target, alignment between the FE model and measurement must be performed. Different methods can be used to perform this step. We decided to use a best-fit approach with specified tolerances, performed with the commercial software tool "GOM Inspect". The reason for using a best-fit approach for alignment is the fact that the average distance between "nominal geometry" and actual geometry is minimized; the maximally possible number of boundary conditions for the FE model can be achieved for the first iteration of the morphing algorithm, see distance computations in Fig. 6. However, the best-fit alignment of the measurement is performed without a specific reference frame. Conventional 3-2-1 alignment must be performed after morphing to embed the resulting model in the reference frame used for vehicle definition.

The parameter set used for the morphing algorithm is listed in Table 2. The relaxing conditions were chosen dependent on the iteration number and changed two times. This was chosen such that the sampling density was increased and the number of distances discarded by the normal filter was decreased with rising iteration number.

4.3.1. Results

In Fig. 15 the results of the morphing with actual measurement data are shown. The left side of the figure shows the difference between the simulation mesh based on "nominal geometry" and the actual measured part; differences of up to ±7 mm can be observed. The right side shows the result of the morphing algorithm. For consistency, nine iterations are performed. The results show that only areas near perimeters of the measurement exhibit error values of over ±0.05 mm. The average error is 0.0002 mm and RMS is 0.0017 mm (Areas near mesh perimeters not considered). The morphing process uses the simulation model of the whole assembly. However, the sub-structure cannot be evaluated as it is not captured in the available measurement data. Nevertheless, the sub-structure is morphed to a plausible shape by manually assessing the geometry.

4.4. Run times

To benchmark our method in terms of computational cost, we provide execution times. All times are based on the computing environment summarized in Table 3. Fig. 16 provides an overview of all measured computing times. Each donut diagram represents run times for nine iterations per part/assembly. The times consider times for sub-tasks: (a) initial preparation, (b) FE solver pre-processing, (c) FE simulation, and (d) other computations. The initial preparation time concerns data-loading and extracting and matching mesh perimeters. As these operations have to be done one time only, the preparation of the first iteration is listed separately. The FE solver pre-processing time relates to preparations and I/O operations performed by the used simulation software, while FE simulation time considers the actual run time of the

Table 2
Parameter set used for morphing scan data.

Parameter	Iteration 1–3	Iteration 4–6	Iteration 7	Iteration 8-9
n Class a [0 mm, 0.05 mm]	5	3	3	1
n Class b]0.05 mm, 1 mm]	3	3	3	3
n Class c 1 mm<	3	3	3	3
Search distance	5 mm	5 mm	5 mm	5 mm
Remove rows	3	1	1	1
Normal angle	15°	30°	50°	60°

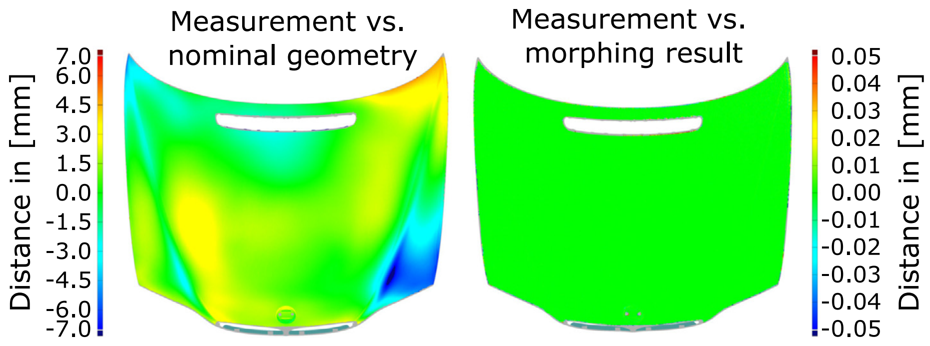


Fig. 15. Left: distance between measurement and nominal geometry; right: morphed simulation mesh compared to measurement.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

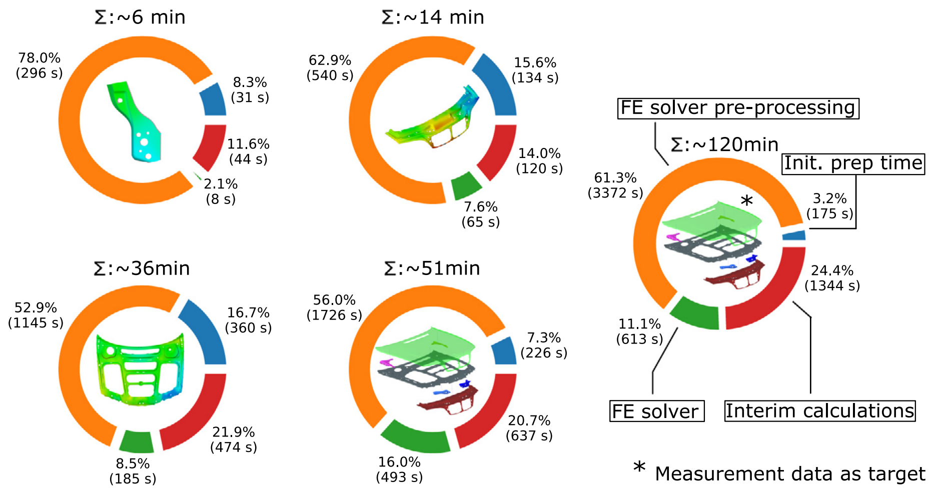


Fig. 16. Measured run times. Total times consider times for (a) initial preparation, (b) FE solver pre-processing, (c) FE simulation and (d) other computations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

FE solver. The other computations cover the run times of the proposed method, in particular the main loop of Algorithm 1. The figure shows the composition of run times for different levels of model complexity. For the single sheet metal parts discussed, run times are between six and 36 min The numbers of mesh elements used are drastically different, see Fig. 10. Morphing the full assembly requires about 51 min for the experiment, with simulated data. The run time of the morphing using real sensor data took with 120 min way longer.

4.5. Loss of element quality during iterations and convergence behavior

To assess the convergence behavior of our method and impact on element quality of the transformation, different quantities are evaluated from the experiments. For the convergence behavior, the RMS error between the current state and target geometry is evaluated after each iteration. The change of element quality is measured by counting the number of elements that fail the quality checks defined in Table 4. The values in the table for the quality checks are ABAQUS standard values for elements not ideal but still acceptable for simulation. The choices of displacement magnitude and placement have a substantial impact on element quality. In our verification and validation, the minimal edge length of small elements is 0.048 mm, and the maximal displacement value of 3.6 mm is about two orders of magnitude larger. To prevent elements from distorting during the calculation, the method takes advantage of element stiffness properties. Most importantly, one must avoid the placement of inconsistent boundary conditions close to each other. This is prevented by applying the filter, coarsening, and weighting steps, described in Section 3.3.

Our evaluation results for element quality and RMS are summarized in Fig. 17. The upper graphs show that the amount of elements considered badly shaped does not change significantly and remains almost constant. The lower graphs show convergence behavior. For all cases, a similar convergence behavior is observed. The approximation changes most substantially in the first three iterations. In the following iterations, the RMS

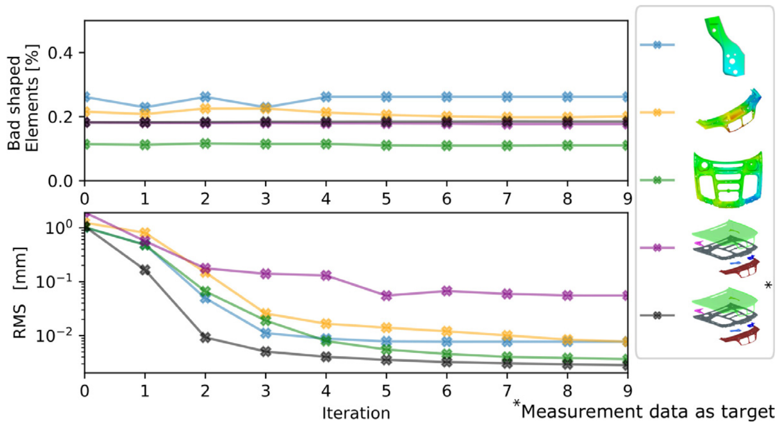


Fig. 17. Evaluation of element quality and convergence behavior for all experiments. Top: number of badly shaped elements in % across all iterations; bottom: convergence behavior of proposed algorithm using RMS error per iteration. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Specification of computing environment used to measure run times.

Processor	Intel i7-5820k@3.6 GHz
Memory	64 Gb DRR4@2100 MHz
Storage	M.2 SSD read; write 3200 MB/s;1400 MB/s
Graphics card	2x Nvidia GTX 1070 8 GB GDDR5
Operating system	Microsoft Windows 10 Professional
Finite element solver	ABAQUS - Dassault Systèmes

Table 4
Standard element criteria used by ABAQUS for evaluating mesh quality.

Quality criterion	Value
Shape factor (triangles) less than	0.01
Triangle corner angle less than	5°
Triangle corner angle greater than	170°
Quadrilateral corner angle less than	10°
Quadrilateral corner angle greater than	160°
Aspect ratio greater than	10
Edge length shorter than	0.01 mm

value still decreases towards zero but with smaller changes per iteration.

5. Discussion

We discuss the following aspects: (a) parameters of Algorithm 1, (b) the results of different validations, (c) computational times, (d) convergence behavior, (e) simulation parameters, and (f) drawbacks and limitations of the approach.

Parameter of Algorithm 1 — A common problem with flexible algorithms is the fact that for every new case a suitable set of parameters must be found. This is time-consuming and usually carried out with a trial-and-error method. Although our algorithm has several parameters that must be defined, it is not difficult to find suitable values, as parameters are related to understandable geometric measures, compare Section 3.4. Selecting not-quite-ideal parameter values mostly results in bad convergence behavior and longer run times, but not in a failure of the method.

Verification results — The presented verification results document the applicability of our approach to real-world cases. The simplest use case morphs a single sheet metal part to a noise-free target mesh that fully covers the geometry. The results are shown in Fig. 11. The final result shows, for all morphed part shapes, error values below ±0.05 mm, which is the measurement uncertainty for 3D scanning devices commonly used in industrial applications. Single spots with minor deviations below ±0.05 mm occur due to mesh resolution. The mesh nodes are close to the target surface, but the element edge deviated from the target mesh and shows a difference in the surface-to-surface distance comparison. An example for this problem is given in Fig. A.18. This is a minor weak point of the method that would be worth improving, but for most applications, the sub-mesh resolution approximation is not relevant. We conclude that the precision target defined in Section 1 has been achieved.

In the second verification, presented in Section 4.2.1, a full assembly was morphed only using a target mesh for the outer skin. This experiment was chosen to show how the proposed method handles not-measured sub-structures that often cannot be scanned, especially in assembled situations. The goal was to obtain at least a realistic approximation of the underlying structure. The established goal of ±0.05 mm cannot be kept for not-measured areas, as information for performing high-fidelity morphing is not contained in the measurement. This verification demonstrates that, during the deformation of the underlying structure, no effects like buckling occur. We can accept error values up to ±1 mm. The results of this verification are shown in Fig. 13. The comparison between target and morphed assembly parts does not show any deviations above ±0.5 mm. All parts of the sub-structure are of a plausible shape. Although the digital twin is not accurate in terms of geometry, we argue that the results are better compared to not updating the sub-structure.

Validation results — The validation presented in Section 4.3 was performed to show the applicability to real sensor data. The results are shown in Fig. 15. In this case, the measurement covers only the outer skin, but not entirely. Especially, perimeters and areas not visible for both cameras of the scanner are not captured. For morphing, the perimeters were not considered and only boundary conditions perpendicular to the surface were applied. The results have errors that are below 0.05 mm, which meets the established precision target. Only small areas towards mesh perimeters of the measurement show higher approximation errors. Closer examination has revealed that observed differences in these areas are caused by measurement errors. At these locations, the tessellation of the point cloud has outliers that are not removed by previous smoothing or cleaning steps. However, since it is forbidden to place FE-boundaries near perimeters by the selected parameters for the mesh morphing, the result remains smooth, causing the observed differences in the comparison. This leads to a higher RMS error value the result is converging to — see Fig. 17.

Computational times — The computing times of our method could be further improved, and there exists a great potential for

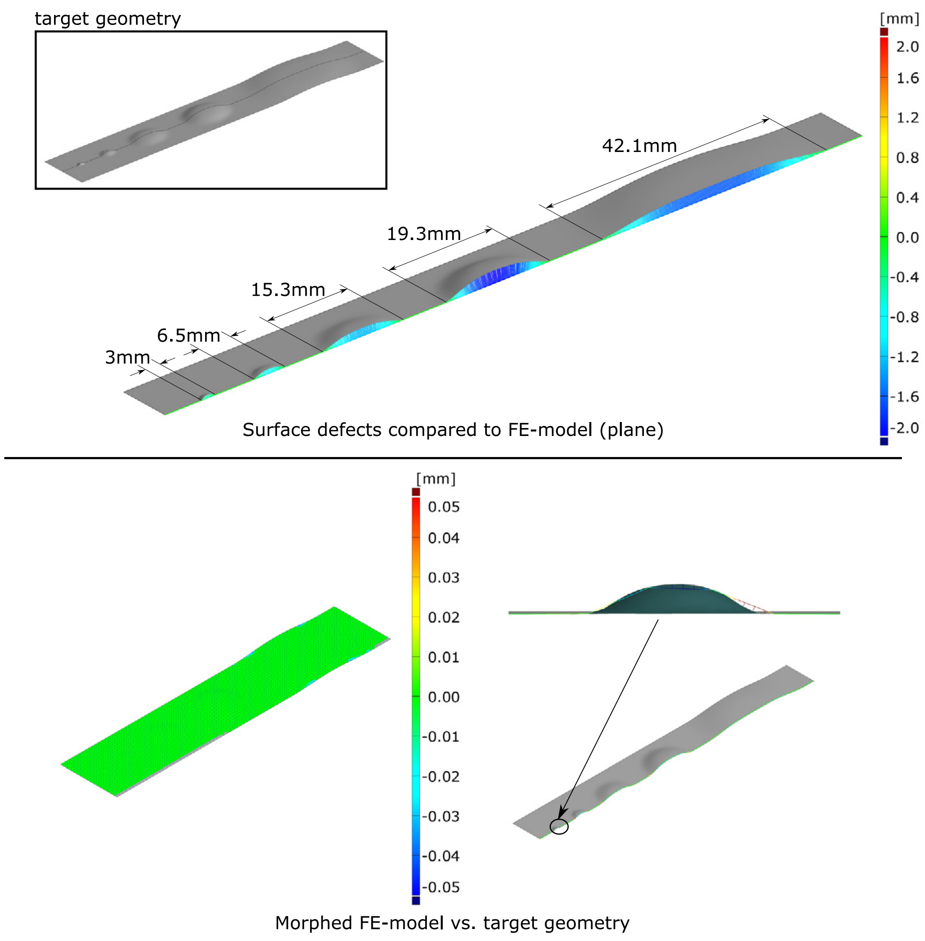


Fig. A.18. Artificial example of morphing surface defects of different sizes. Top: target geometry compared to FE mesh (plane). The cross section shows radii and heights of dents. Bottom: morphed results compared to target geometry. The cross section shows approximation errors for small dents below mesh resolution. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

improvements in this regard. The run times are summarized in Fig. 16. With the rising complexity of the FE problem, run times are increasing. It is striking that most of the overall computation time is spent in the FE pre-processing step. This step is performed by the simulation software and could only be improved by turning off checks performed by the pre-processor, or by switching to another simulation software. We consider this part of run time behavior as a high potential for improvements. We want to highlight the difference in run times between the full assembly with simulated and measured target geometry. Although model size is exactly the same, run times are different. This can be explained by the different densities of the target meshes, having an impact on the other necessary computations. However, the major computing time was spent on the last iterations where the density of boundary placement was raised to achieve good local approximation results. For these iterations, almost three times more boundary conditions were used which resulted in very large pre-processing times while solver times stayed on the same scale.

Convergence behavior — The convergence behavior of the algorithm is shown in Fig. 17. We measure convergence as a function of RMS depending on iteration, which should converge towards zero. The convergence behaviors for all experiments are similar to each other. The largest improvements are achieved during the first two iterations. From iteration three, RMS values only change in a small way, converging to zero stably. It is important to have stable and fast-converging behavior of the algorithm; otherwise, an optimal number of iterations would have to be defined instead of a minimum error value.

Concerning convergence of the FE problem, we discuss factors affecting numerical stability. Most crucial for the stability of the simulation problem is the choice of displacement boundary conditions. If boundaries that are close to each other are inconsistent or even conflicting, then mesh distortions will occur and the FE problem will not converge. To prevent such conflicts, different precautions were made:

- Controlling density of boundary placement by coarsening matches
- Application of weight function to prevent excessive displacements
- Thresholds applied to displacements

In addition, a single linear-order element formulation and linear elastic material model are used, and non-linear effects are neglected to improve the robustness of the FE calculations further.

Simulation parameter — Different parameters for the FE computations can influence the final shape of the morphed mesh and the computation times. Especially the material model defines the behavior of the part/assembly and thus influences the output. In our validations, we used a linear elastic model and did not consider geometric non-linearity for the FE computations. A cubic interpolation between mesh nodes with boundaries can be achieved, as the flexible behavior of the simulation mesh minimizes bending energy. As the real behavior of the part is not of interest during morphing, different simulation parameters can be used to change the behavior of the mesh and control element

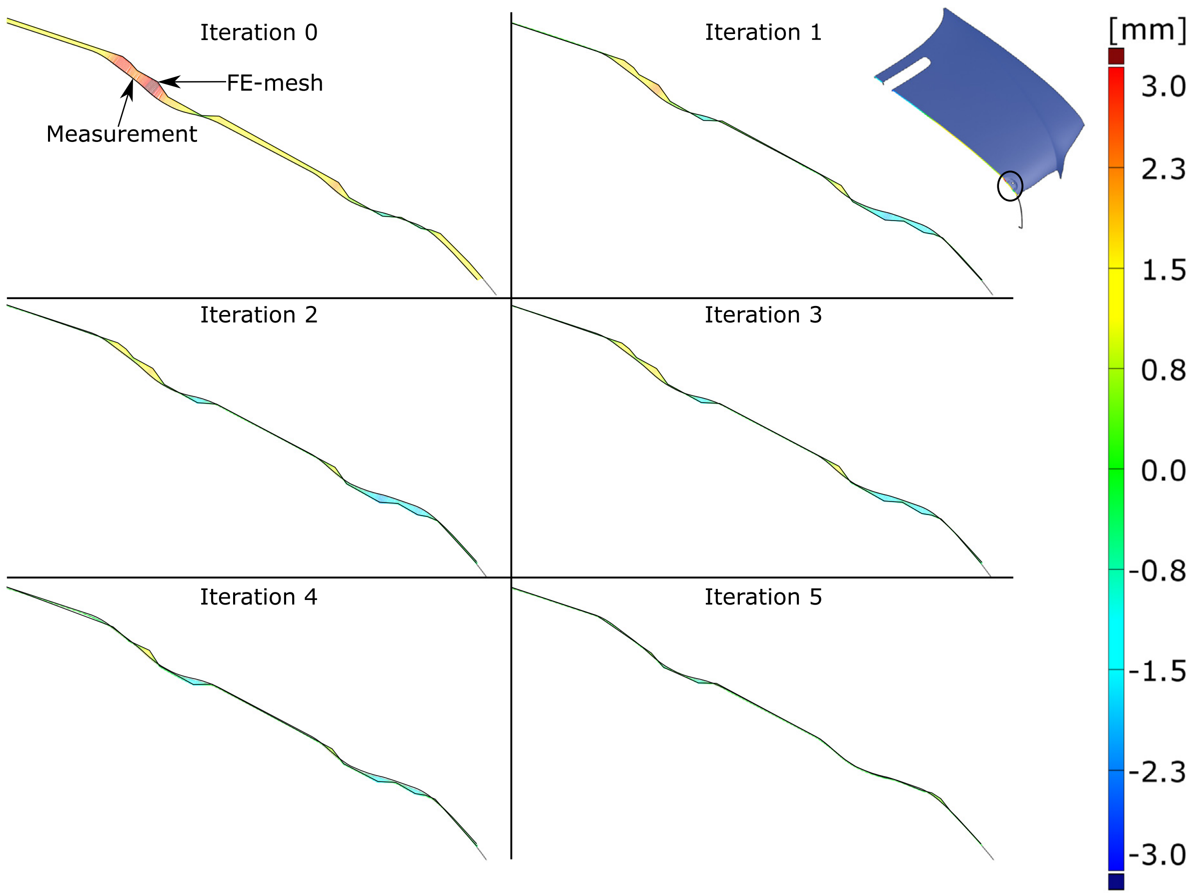


Fig. A.19. Evaluation of cross section at emblem position during iteration steps 0–5. "Iteration 0" represents the initial situation and "Iteration 5" is an intermediate result close to a morphed situation. It can be observed that the emblem is offset at "Iteration 0". This offset is removed during morphing. This evaluation results is based on scan data from Section 4.3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

distortions. We did not perform investigations concerning this issue and consider it as potential future research.

Drawbacks and limitations of the approach — The presented morphing approach provides a high-fidelity output and hardly requires any manual interventions. Nevertheless, some drawbacks and limitations should be pointed out. Further improvements are desirable concerning computational cost and scalability. Computational times could already be reduced as discussed; unfortunately, each iteration involves the execution of the FE-solver to simulate displacements for the full simulation model. Executing the FE-solver is essential and cannot be avoided or drastically accelerated. Another issue is the fact that no mesh quality controls are implemented. This can lead to errors when mesh elements are poorly shaped. The current implementation of the proposed algorithm does not address the issue of poorly shaped elements. The proposed method was designed and only validated for surface meshes. Using the method for volume meshes was not assessed, and modifications and extensions would be necessary to extend the method to volume meshes.

6. Conclusions

We have described an FE-based mesh morphing algorithm that uses a simulation mesh and a measured part geometry as input and computes a morphed simulation model. Our method only needs access to node positions of the FE model and nodal simulation results. It can be implemented straightforwardly with commercial software tools. We have implemented a prototype in python, using ABAQUS from Dassault Systèmes as FE solver. Our prototype can be used to validate our method for industrial applications with simulated and experimental data, and we have summarized and discussed our results. The results of the simulated experiments meet the pre-defined quality goal of error values below ± 0.05 mm for ideal conditions. Morphing results of the experimental data show only slightly higher RMS error values, which can be explained by measurement noise, especially towards the perimeters of the mesh. We have presented run times and have described the convergence behavior of our implementation.

Concerning potential future research, it would be worthwhile exploring reasons for the high preparation times required by the FE solver and considering taking advantage of different material models for the FE model. Theoretically, an extension to volumetric FE models is possible. Further, a local re-meshing approach could be integrated into the implementation to improve mesh quality in areas where elements are distorted significantly by the morphing process.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 252408385 – IRTG 2057. We thank Q-DAS GmbH for providing CAD models.

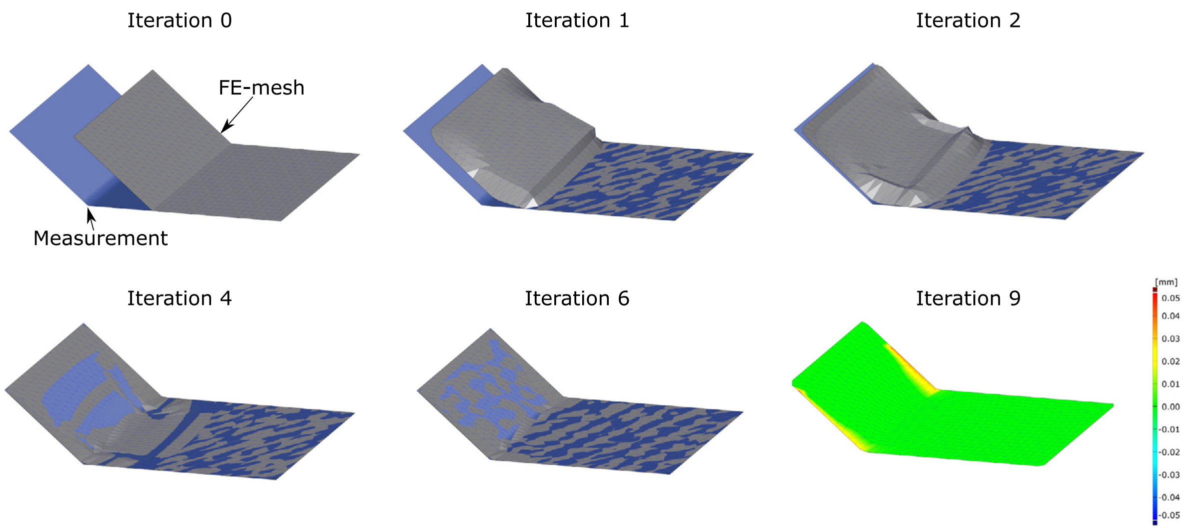


Fig. A.20. Artificial example with large lateral displacements. First, perimeters are closely aligned to each other; next, nodes inside the surface can be considered and are morphed. Mesh elements are stretched but not distorted. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Appendix. Additional test cases

See Figs. A.18–A.20.

References

[1] Alizadeh R, Allen JK, Mistree F. Managing computational complexity using surrogate models: a critical review. Res Eng Des 2020;31(3):275–98. http://dx.doi.org/10.1007/s00163-020-00336-7.

[2] Jones D, Snider C, Nassehi A, Yon J, Hicks B. Characterising the digital twin: A systematic literature review. CIRP J Manuf Sci Technol 2020;29:36–52. http://dx.doi.org/10.1016/j.cirpj.2020.02.002, URL https://www.sciencedirect.com/science/article/pii/S1755581720300110.

[3] Söderberg R, Wärmefjord K, Carlson JS, Lindkvist L. Toward a digital twin for real-time geometry assurance in individualized production. CIRP Ann 2017;66(1):137–40.

[4] Claus F, Rupprecht F-A, Hagen H. Online simulation considering production uncertainties to improve assembly quality. In: Nafems world congress 2019 - summary of proceedings. Québec City; 2019, p. 51, URL https://www.nafems.org/publications/resource_center/nwc_19_356/.

[5] Bagci E. Reverse engineering applications for recovery of broken or worn parts and re-manufacturing: Three case studies. Adv Eng Softw 2009;40(6):407–18. http://dx.doi.org/10.1016/j.advengsoft.2008.07.003, URL https://www.sciencedirect.com/science/article/pii/S0965997808001312.

[6] Anwer N, Mathieu L. From reverse engineering to shape engineering in mechanical design. CIRP Ann 2016;65(1):165–8. http://dx.doi.org/10.1016/j.cirp.2016.04.052, URL https://www.sciencedirect.com/science/article/pii/S000785061630052X.

[7] Racasan R, Popescu D, Neamtu C, Dragomir M. Integrating the concept of reverse engineering in medical applications. In: 2010 ieee international conference on automation, quality and testing, robotics (aqtr), Vol. 2. 2010, p. 1–5. http://dx.doi.org/10.1109/AQTR.2010.5520710.

[8] Ameddah H, Assas M. Bio-CAD reverse engineering of free-form surfaces by planar contours. Comput-Aided Des Appl 2010;7(sup1):1–7.

[9] Yang IT, Acharya TD, Lee DH, Shin MS. Reverse engineering of a bobsleigh structure using lidar. Int J Appl Eng Res 2017;12(6):976–80.

[10] Cabaleiro M, Riveiro B, Arias P, no JC, Vilán J. Automatic 3D modelling of metal frame connections from LiDAR data for structural engineering purposes. ISPRS J Photogramm Remote Sens 2014;96:47–56. http://dx.doi.org/10.1016/j.isprsjprs.2014.07.006, URL https://www.sciencedirect.com/science/article/pii/S0924271614001804.

[11] Singh K. Industrial motivation for interactive shape modeling: A case study in conceptual automotive design. In: Acm siggraph 2006 courses. Siggraph '06, New York, NY, USA: Association for Computing Machinery; 2006, p. 3–9. http://dx.doi.org/10.1145/1185657.1185671.

[12] Varady T, Martin RR, Cox J. Reverse engineering of geometric models—an introduction. Comput Aided Des 1997;29(4):255–68.

[13] Huang Y, Qian X. Dynamic B-spline surface reconstruction: Closing the sensing-and-modeling loop in 3D digitization. Comput Aided Des 2007;39(11):987–1002. http://dx.doi.org/10.1016/j.cad.2007.06.008, URL https://www.sciencedirect.com/science/article/pii/S0010448507001601.

[14] Yu Z, Wang T, Wang P, Tian Y, Li H. Rapid and precise reverse engineering of complex geometry through multi-sensor data fusion. IEEE Access 2019;7:165793–813. http://dx.doi.org/10.1109/ACCESS.2019.2948124.

[15] Remondino F. From point cloud to surface: the modeling and visualization problem, http://dx.doi.org/10.3929/ethz-a-004655782.

[16] Gentilini I, Shimada K. Predicting and evaluating the post-assembly shape of thin-walled components via 3D laser digitization and FEA simulation of the assembly process. Comput Aided Des 2011;43(3):316–28. http://dx.doi.org/10.1016/j.cad.2010.11.004.

[17] Shimada K. Physically-based mesh generation: automated triangulation of surfaces and volumes via bubble packing (Ph.D. thesis), Massachusetts Institute of Technology; 1993.

[18] Guan B, Lin S, Wang R, Zhou F, Luo X, Zheng Y. Voxel-based quadrilateral mesh generation from point cloud. Multimedia Tools Appl 2020;79(29):20561–78. http://dx.doi.org/10.1007/s11042-020-08923-5.

[19] Sederberg TW, Parry SR. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on computer graphics and interactive techniques*, 1986, p. 151–60.

[20] Gaun L, Bestle D, Huppertz A. Hot-to-cold CAD geometry transformation of aero engine parts based on B-spline morphing. Turbo expo: power for land, sea, and air, Vol. Volume 2B: Turbomachinery, 2014, V02BT45A020. http://dx.doi.org/10.1115/GT2014-26683, arXiv:https://asmedigitalcollection.asme.org/GT/proceedings-pdf/GT2014/45615/V02BT45A020/4228981/v02bt45a020-gt2014-26683.pdf.

[21] Gagliardi F, Giannakoglou KC. RBF-based morphing of B-rep models for use in aerodynamic shape optimization. Adv Eng Softw 2019;138:102724.

[22] Lange A, Vogeler K, Gümmer V, Schrapp H, Clemen C. Introduction of a parameter based compressor blade model for considering measured geometry uncertainties in numerical simulation. Turbo expo: power for land, sea, and air, Vol. Volume 6: Structures and Dynamics, Parts A and B, 2009, p. 1113–23. http://dx.doi.org/10.1115/GT2009-59937, arXiv:https://asmedigitalcollection.asme.org/GT/proceedings-pdf/GT2009/48876/1113/2765201/1113_1.pdf.

[23] Lange A, Voigt M, Vogeler K, Schrapp H, Johann E, Gümmer V. Impact of manufacturing variability on multistage high-pressure compressor performance. J Eng Gas Turbines Power 2012;134(11):112601. http://dx.doi.org/10.1115/1.4007167, arXiv:https://asmedigitalcollection.asme.org/gasturbinespower/article-pdf/134/11/112601/5892954/112601_1.pdf.

[24] Van der Auweraer H, Van Langenhove T, Brughmans M, Bosmans I, Masri N, Donders S. Application of mesh morphing technology in the concept phase of vehicle development. Int J Veh Des 2007;43(1–4):281–305.

[25] Franciosa P, Gerbino S, Patalano S. Simulation of variational compliant assemblies with shape errors based on morphing mesh approach. Int J Adv Manuf Technol 2011;53(1):47–61. http://dx.doi.org/10.1007/s00170-010-2839-4.

[26] Alexa M. Recent advances in mesh morphing. Comput Graph Forum 2002;21(2):173–98. http://dx.doi.org/10.1111/1467-8659.00575, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00575 URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00575.

[27] Selim M, Koomullil R. Mesh deformation approaches – a survey. J Phys Math 2016;7(2):1–9. http://dx.doi.org/10.4172/2090-0902.1000181.

[28] Kim J, Miller BJ, Shontz SM. A hybrid mesh deformation algorithm using anisotropic PDEs and multiobjective mesh optimization. Comput Math Appl 2015;70(8):1830–51. http://dx.doi.org/10.1016/j.camwa.2015.08.008, URL https://www.sciencedirect.com/science/article/pii/S0898122115003764.

[29] Geuzaine C, Johnen A, Lambrechts J, Remacle J-F, Toulorge T. The generation of valid curvilinear meshes. In: Idihom: industrialization of high-order methods-a top-down approach. Springer; 2015, p. 15–39.

[30] Dawson-Haggerty, et al. trimesh, URL https://trimsh.org/.

[31] Myronenko A, Song X. Point set registration: Coherent point drift. IEEE Trans Pattern Anal Mach Intell 2010;32(12):2262–75.

[32] Kenta-Tanaka, et al. probreg, URL https://probreg.readthedocs.io/en/latest/.

[33] Claus F, Hagen H, Leonhardt V, Leitte H, Hamann B. Interactive quality inspection of measured deviations in sheet metal assemblies. In: Garth C, Aurich JC, Linke B, Müller R, Ravani B, Weber GH, Kirsch B, editors. 2nd international conference of the dfg international research training group 2057 – physical modeling for virtual manufacturing (ipmvm 2020). Open access series in informatics (oasics), Vol. 89, Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik; 2021, p. 6:1–6:18. http://dx.doi.org/10.4230/OASIcs.iPMVM.2020.6, URL https://drops.dagstuhl.de/opus/volltexte/2021/13755.