Multiresolution Surface Blending for Detail Reconstruction

Hono Salvala, Andy Keanea and David Toalb

aComputational Engineering and Design group, University of Southampton, Rolls Royce University Technology Centre for Computational Engineering, Boldrewood Innovation Campus, Burgess Rd, Southampton, SO16 7QF, Hampshire, United Kingdom

ARTICLE INFO

Keywords:
surface reconstruction
assembly based modelling
shape blending
reverse engineering

ABSTRACT

While performing mechanical reverse engineering, 3D reconstruction processes often encounter difficulties capturing small, highly localised surface information. This can be the case if a physical part is 3D scanned for life-cycle management or robust design purposes, with interest in corroded areas or scratched coatings. The limitation partly is due to insufficient automated frameworks for handling -localised - surface information during the reverse engineering pipeline. We have developed a tool for blending surface patches with arbitrary irregularities, into a base body that can resemble a CAD design. The resulting routine preserves the shape of the transferred features and relies on the user only to set some positional references and parameter adjustments for partitioning the surface features.

1. Introduction

New computer technologies for processing and capturing geometric data have emerged in the previous decade. Several methods in engineering deal with digital models, which are altered in real-time to mirror properties of real-world objects, one of these methods being the digital twin [1, 2, 3], used to create a real-time updated digital representation of a device in service. Aside from technical applications, visuals for animation and video games have shifted their focus to incorporating captured geometry from real-world objects rather than creating it artificially. With various applications for simulation and manufacturing, we believe that incorporating outsourced data into engineering design frameworks is more approachable now than it was previously. Simulating the effects of surface flaws, for example, is not a simple operation; it necessitates either sculpting the design’s surface or transferring irregularities from an external source while maintaining key shape attributes.

While building models from diverse combined shapes is typical in the videogame business, the engineering design process is generally confined to a distinct framework with its own methodologies and geometric formats. The necessity of employing efficient processes in large projects, such as when design, simulation, and product management occur collaboratively across a complete team, reinforces this tendency. The rigidity of NURBS geometry (Non-Uniform Rational Bézier Spline), also known as solid geometry and often used in engineering design frameworks, is another incentive to stay in a single framework. Meshes, on the other hand, are easier to modify and manipulate locally, whereas solids typically require a reconfiguration step to accommodate additional entities, and any local modification necessitates the addition of new NURBS patches.

Figure 1: Top: a surface patch placed near a blade's trailing edge. The patch contains typical scratches found in turbine coatings. Bottom: blended scratches added into the blade's surface.

The combined use of solid and mesh format is often found when reconstructing 3D scans, exporting geometry for simulation or preparing the model for 3D printing, but they rarely coexist during the design process. Nevertheless, some popular packages offer tools for handling meshes along with solids. Take Siemens PLM as an example; it provides a geometric object called “convergent geometry” that handles the so-called free-form formats for tasks like sculpting, filling holes, or shape deformation. These tools

Hono Sal. Val., Andy Keane, David Toal: Preprint submitted to Elsevier
are fundamentally manual and rely on the user to blend and stitch shapes together, leaving an isolated framework that does not fully integrate into the engineering design workflow. This deficit is commented on by Bodgan et al. [5] when performing the reconstruction of a Francis turbine scan, during which localized defect information is lost because of the lack of a framework for managing it.

Our algorithm handles such localized irregularities by extracting surface regions and transferring them to another model's surface. Referring to figure 1, the routine should be capable of deforming the extracted region to match the model's curvature at the blending area, and it should avoid distortions on the relevant detail. As scratches in the figure resemble a particular material displacement defect, any alteration of the relative proportions might affect later simulation results. A tool that provides good and automated detail transfers also makes it possible to build a repository of geometric features (e.g., scanned defects on physical objects, textures and other resources for simulation). The routine is achieved by separating the patch into a feature layer and a thin-plate layer, which supports the features. Then the thin-plate is mapped onto the design, being folded to wrap on the surface. To finish the transfer, features are reconstructed with new positioning and orientation onto the fitted thin-plate (see figure 2).

![Figure 2: A surface patch with bumps is blended into another shape, which overall curvature doesn't match the patch. a) The fitting region (black/yellow square UV map) is projected onto the base. b) Surface features are then reconstructed on the fitted thin-plate.](image)

2. Related work.

The task of blending shapes has been discussed many times: building models from a precomputed database of parts [6] is an early approach that ensures the outcome by restricting the choices to a given part set. More flexibility has been gained since then, by connecting arbitrary boundaries with biharmonic surfaces [7], to optimizing and deforming the boundaries to blend [8]. A recent approach is to provide hierarchical reasoning that helps decompose geometry into a certain set of features, and later reassembling new shapes upon those features. Sometimes this hierarchy is hard-coded into the routine using predefined relationships [9, 10, 11, 12], but with the increasing popularity of machine learning techniques much attention has shifted to self-learning algorithms, trained with hundreds of examples, including assembled and disassembled shapes [13, 14, 15, 16, 17, 18].

The utility of these tools on product design affects various processes: they can suggest design aspects to the user [19, 20, 21, 22], facilitate the recovery of shapes from images [23, 24, 25, 26, 27] and scan data [26, 28, 29]. Even further, well developed assembly algorithms enhance design search for purposes such as generative and robust modelling, where certain constraints are imposed and an algorithm tries different designs in a search for the one matching the imposed conditions [14, 30, 31, 32, 33]. Algorithms for performing design exploration are of the most interest for us, along with other practices such as surface inspection. As previously commented, our routine aims to transfer surface features between non-shape-matching surfaces; it merges both geometries while preserving the relative proportions of each individual feature. Straight applications of such a tool are: shape reconstruction for surface defect inspection (such as coating scratches, abrasion) [34, 35], simulation of surface irregularities in robust design exploration [36], and real-time diagnosis of devices through sensor data processing [4].

Building the proposed tool require a high-level perspective of industrial design requirements and the different subprocesses involved in an automated blending algorithm. Most CAD software shares several core aspects; the design philosophy is to build models by introducing new elements and constraints to the geometry. Also, the usual way to encode shapes is by assembling solid primitives (as planes, spheres, cages) and surface patches defined by scalar functions. The typical CAD design is composed of well defined geometric parameters such as lengths or angles, and even more abstract characteristics can be introduced such as rotational symmetries. This practice enhances compatibility with manufacturing processes and simplifies the design process for the user through a programmatic framework. When bringing arbitrary mesh geometry to an engineering design framework, it is often processed to isolate and classify the different parts of its shape through segmentation, so the same engineering, programmatic-like framework can be applied for the imported mesh.

**Segmentation of surfaces.** Performing a segmentation in a surface mesh means isolating and -often- classifying regions from a model. It is often utilized in processes such as topological optimization, surface reconstruction and computational object recognition. A common situation is wanting a triangular mesh to be reconstructed as a NURBS surface, and there is plenty variety in publications regarding this topic [37, 38, 39, 40]. In particular, Laplacian and Bi-Laplacian based segmentation are pretty flexible in terms of recognizing features of any nature. They are applicable on acute mechanical shapes or organic bodies, yielding in

---

1 A programmatic framework allows to introduce new aspects of the shape procedurally.
both cases an intuitive separation of the different parts. Publications such as Mejia’s et al. combine seeding and Laplacian techniques to accelerate the feature isolation [41, 42], referring to this approach as spectral-based mesh segmentation. In the standard terminology of digital geometry, spectral mesh processing refers to the manipulation and analysis of shapes by exploiting the properties of operators like the graph Laplacian applied on a surface mesh. Laplacian methods capture geometric features in different scales, often also described as decoupling the curvature in different frequencies of varying magnitude.

There are still limitations, particularly for recognizing high abstraction features as components of an assembly; these can be addressed by novel routines based on hardcoded hierarchical reasoning [12] and deep learning optimizers [43, 44]. Another capability lacking in segmentation algorithms to date is handling localized detail surface information. As scanning technology reaches greater accuracy, there is a growing interest in more localized surface details. These can be introduced into simulation pipelines for application life-cycle management (ALM) [4], or as part of a reverse engineering process. While scanning a heavily eroded Francis Turbine, it is noted by Sedai et al. [45] that there is no tool for handling erosions and defects during the reverse engineering process, which discards those in favour of larger scale features. Apart from disregarding small detail during the segmentation process, encoding this kind of shape is another obstacle, as complex irregularities require a large number of NURBS patches to be defined. For that reason, another technique is often applied to handle mesh deformations, usually referred to as free-form deformation (FFD) by industrial side publications, and more commonly named skinning in audiovisual, creative and machine learning circles.

**Skinning and deformation.** Probably one of the best definitions for presenting this concept is provided by Jacobson, Deng, et al. “Skinning is the process of controlling deformations of a given object using a set of deformation primitives” [46]. Skinning can be applied in many different forms, regarding the chosen handlers: e.g. skeleton, bones [46], point, lines and cages [47], or point lattices [44, 48]. And also regarding how the geometry deforms when moving the handlers: e.g. biharmonic deformation fields [47, 49] or spline interpolation [44, 48]. In particular, for handling surface patches, it is common practice to generate a thin-plate version of the surface where any irregularity is removed and use it to guide detail to new positions and orientations (see figure 3). Generating the thin-plate can be done by minimizing acute curvature along the surface, leading to a softening effect. As an example, the Bi-Laplacian operator $\Delta^2$ can be discretized for a triangular mesh and serves to smooth away high-frequency features of a given shape.

Solving the following equation, with fixed boundaries $v_{i \in bc} = v_{bc}$, leads to a smoothed version of the patch:

$$\Delta^2v_i = 0 \quad \text{subject to} \quad v_{bc} = v_{bc}. \quad \text{} \quad (1)$$

The resulting surface would have the minimal curvature imposed by the boundary conditions, leaving a biharmonic surface. After performing any transformation on the thin-plate, details need to be reconstructed with the new shape. For this process, there are various propositions, one of the simplest ones being based on Laplacian processing. It can also be applied on deformation fields to remove abrupt deformations out of a given deformation field [50]. To perform what is called a biharmonic deformation, we solve equation 1, this time applied on a deformation field $\{d_i\}$ over the mesh vertices $\{v_i\}$:

$$\Delta^2d_i = 0 \quad \text{subject to} \quad d_{bc} = d_{bc}. \quad \text{} \quad (2)$$

The displacements $d_{bc}$ are imposed at selected indices $i \in b$, the solution $\{d_i\}$ is a soft interpolation of the deformation field passing through the imposed values (see figure 4).

**Figure 3:** a) Surface patch with bump irregularities. b) Thin-plate version of the patch.

**Figure 4:** Biharmonic deformation of a patch, where a central vertex has been displaced upwards. The result is an interpolating surface that smoothly extends the deformation along the surface.

Biharmonic deformation has limitations, though, being a linear method, it fails to calculate local rotations after deformation correctly, and the reconstructed features appear tilted towards their original orientation. This behaviour is often referred to as a non-rotation-invariant deformation. To avoid this problem and accomplish shape preservation, non-linear techniques come into play, such as rigid deformation algorithms. As-rigid-as-possible (ARAP) deformation
minimizes the stretching and bending energy of the mesh, introduced by the imposed transformation. Roughly describing the mathematical details, rigid deformation methods aim to find, for given boundary conditions, the new positions \( \{v'_i\} \) and the local rotations \( \{R_i\} \) satisfying the following difference:

\[
(v'_i - v'_j) = R_i(v_i - v_j),
\]

where \( v_i \) are the original positions of the mesh vertices, \( p' \) the deformed positions and \( R_i \) the local rotation matrix for each vertex. A variational approach to solve this equation was proposed to decouple the search for \( R_i \) and \( v'_i \) [51]. This approach provides more physically intuitive deformations (see figure 5), but often at the cost of stability and processing power. Not directly involved in our work but worth of mention is another method offering rotation-invariant results, which aims to force the deformation gradient of a thin-plate to be inherited by the original geometry [52].

\[ \text{Figure 5: } \text{a) Deformed patch through biharmonic deformation. b) Deformed patch through rigid deformation. The bumps in the biharmonic approach get tilted towards their original orientation, while for the rigid deformation, they correctly reorient to match the new disposition.} \]

This collection of techniques (segmentation, skinning, deformation) have been combined together for various surface blending solutions [53, 20], highlighting the ones that include Laplacian techniques on their approach [49, 54, 55, 56, 8].

3. Multiresolution Surface Blending.

To understand how the term Multiresolution fits into the description of this routine, consider the surface patch as a combination of two geometric entities: one is a soft surface that depicts the overall curvature, and the other defines surface irregularities that are distributed along the soft surface. This results in a two-level description of the shape, which is known as a multiresolution scheme. This view is useful since each of these levels must be addressed individually when fitting the patch onto the base surface. The supporting level should inherit the base’s curvature while keeping its surface area and shape, while the detail level should calculate the supporting level’s inherited deformation. The schematic of figure 6 presents an exploded view of the steps.

This section outlines the proposed method for transferring imperfections from a triangle edge manifold mesh (the patch) to another surface (the base). It will be illustrated using a toy model of a turbine blade as the base model, as well as a transfer patch with bumps. Consider a triangular mesh patch of vertex positions \( p_i \) stacked on the matrix \( P \); the mesh connectivity is described by a face matrix \( F_p \), which remains unaltered through the process and therefore

\[ \text{Figure 6: Flow diagram of the blending routine. 1. Thinplate generation: High frequency features are removed from the patch} \ P \ \text{to yield a smooth supporting thin-plate} \ P'. 2. Define placement: Placement references are provided by the user or inferred by some heuristics. 3. Planarization: The thin-plate is planarized by unfolding while keeping proportions and distances. 4. Cut region of blend: From the base model, a region where the blending should happen is isolated and cut leaving an edge manifold surface; which is also mapped to the plane in a shape preserving manner. 5. Alignment: Both planar meshes are aligned, according to the positional references. 6. Barycentric mapping: The planar thin-plate vertices are mapped to positions in the planar base surface. Such map serves to obtain the thin-plate fitted on the original base model. 7. Reconstruction: Details are reconstructed on their updated disposition, over the thin-plate wrapped over the model.} \]
As an overview, \( \mathcal{P} \) denotes the smoothed (thin-plate) version of \( \mathcal{P} \). The complete blending process performs a transformation \( \mathcal{P} \rightarrow \mathcal{P}' \) so that the shell version of blended patch \( \mathcal{P} \) matches the base surface.

Regarding the involved software and tools, most of the routine is built with the Libigl library [57] in Python, while the samples were prepared with Blender [58] and MeshLab [59].

3.1. Thin-plate generation.
As briefly described in the literature section, it is often helpful to look at surface details as features lying on a smooth curved shell, called the thin-plate. This two-level structure is essential for this approach, as transferring the features onto a new model can be seen as deforming the thin-plate to fit the model’s surface and then reconstructing the detail elements in the new dispositions. The thin-plate \( \mathcal{P} = \{p_1, p_2, ..., p_N\} \) we seek is a softened version of \( \mathcal{P} \), and it should match the original surface on those vertices which aren’t part of irregular areas while interpolating the feature-occupied regions. There is no alteration of mesh topology:

\[
\mathbf{T}_{\text{thin}} : \mathbb{R}^{3 \times N} \times \mathbb{N}^* \rightarrow \mathbb{R}^{3 \times N} \\
(\mathcal{P}, I_{\text{smooth}}) \mapsto \mathcal{P} = \{p_i\}_N
\]  

As an overview, \( \mathbf{T}_{\text{thin}} \) is a surface smoothing transformation which alters only those vertices contained in high-frequency shape regions \( i \in I_{\text{feature}} \).

**Segmentation of the patch.** It is required to differentiate the vertices contained in high-frequency regions from the rest, and the Laplacian serves as metric to differentiate smooth from detailed regions. The implementation applies a single step Laplacian smoothing [50] and evaluates the suffered displacements and normal rotation. A single step Laplacian softening

\[
\mathbf{T}_{\text{smooth}} : \mathbb{R}^{3 \times N} \rightarrow \mathbb{R}^{3 \times N} \\
P \mapsto (P - \mathbf{E}),
\]

were \( \mathbf{E} = \{e_i \in \mathbb{R}^3\} \) are vertex-wise displacements after smoothing, which is obtained by solving:

\[
(M + L)(P - \mathbf{E}) = LP,
\]

which leads to

\[
(M + L)E = MP,
\]

where \( M \) is the mass matrix of the mesh, \( L \) is the cotangent Laplacian. We then evaluate per-vertex displacement and rotation, although choosing per-face computations would be an option too:

\[
R(i) = \hat{F}_d(i) + \hat{F}_n(i)
\]

The rotational term \( F_n(i) \) can be calculated in several ways, it is a measure of angle shift for each vertex. Doing a polar decomposition for each triangle deformation is an option, comparing the normal map before and after the smoothing would be a valid approach too. The hats in \( \hat{F}_d \) and \( \hat{F}_n \) indicate that the corresponding fields are normalized to the unit. \( R(i) \) is a height map over the vertices, which highlights high inflexion areas. By imposing a threshold value on \( R(i) \), we can isolate the vertices in detail containing regions (see figure 7), stored as an index set

\[
I_{\text{feature}} = \{i | R(i) > \text{thresh}\},
\]

It is also useful to denote the index set of vertices excluded from high-frequency regions,

\[
I_{\text{smooth}} = \{i | R(i) < \text{thresh}\}.
\]

**Removing features from the patch.** The thin-plate \( \mathcal{P} \) is generated by minimising the biharmonic energy (eq. 1) in \( I_{\text{feature}} \) while holding the soft regions \( I_{\text{smooth}} \) in their original disposition. Fixing the position of the non-feature-containing regions guarantees that the resulting softened surface accurately portrays the patch’s underlying curvature:

\[
\Delta^2 p_i = 0 \quad \text{subject to} \quad p_i = p_j \quad \text{if} \quad i \in I_{\text{smooth}}.
\]

**Figure 7:** Height field \( R(i) \) of position and normal displacement; and segmentation map obtained from imposing a threshold on the height-map. Light areas resemble the vertices in \( I_{\text{feature}} \), the rest (darker areas) are considered non-feature containing regions \( I_{\text{smooth}} \).

**Figure 8:** From left to right: the original mesh has their irregularities (high-frequency features) removed by softening until only a soft surface is left (low-frequency feature).
This softened version of the patch can be thought of as a supporting plate for the features, which will be utilised to compute the mapping to the base surface (see figure 8). The details are then recreated in the new configuration.

3.2. Fitting the thin-plate onto the base.

The next step is to place the thin-plate onto the base’s surface, following criteria of shape preservation. The placement is decided by providing positional references between both surfaces (figure 9). Those can be defined by mapping selected vertices of the thin-plate $\mathcal{P}$ (with indices stored in $I_{\text{map}}$) to vertices of $\mathcal{B}$:

$$m : I_{\text{map}} \rightarrow \mathbb{N}$$

(11)

The method through which those references are chosen varies depending on the intention behind blending the surfaces. For a precise placement, the user could provide piecewise correspondence between a few vertices. For the test presented here, the twentieth closest points between the patch’s and base’s surface are selected (see figure 9).

Figure 9: Visual representation of the placement reference map $m(i)$. a) In our experiments, references are provided as the closer vertex pairs between base and patch. b) The green dots represent $\{b_{uv}\}$ and the red dots are their corresponding vertices in the patch.

Unfolding meshes to the planar space. An intuitive way to compare surface locations is mapping both meshes $\mathcal{P}$, $\mathcal{B}$ to the two-dimensional plane while preserving geodesic distances. The planarized -aka unfolded- meshes are later aligned by minimising distances between each pair $\{p_i, b_{uv(i)}\}$. To simplify computations, the surface region of the base that concerns the blending process is isolated from the rest of the model (see figure 10). For that, the vertices $\{b_{uv(i)}\}$ are selected, and the adjacent ones are iteratively added, up to a certain number of iterations. Another approach to this region isolation step would be to measure the maximum geodesic distance $d_{\text{max}}$ from a vertex $p_i$ in $I_{\text{map}}$ to the edge of the thin-plate, and then from $b_{uv(i)}$ measure its geodesic distance to the rest of the base’s vertices, gathering the ones within $d_{\text{max}}$. That alternative could be much more computationally expensive in dense meshes.

The thin-plate and isolated base region are then planarized using the Libigl library’s rigid deformation method (ARAP) [51, 60, 61] (see figure 11). We chose this method as it maintains the relative distance between vertices after they unfold. The vertex positions of the planar surfaces are stored in matrices $\mathcal{P}_{uv}$ and $\mathcal{B}_{uv}$.

$$T_{\text{unfold}} : \mathbb{R}^{3\times N} \rightarrow \mathbb{R}^{2\times N}, \quad \mathcal{P} \mapsto \mathcal{P}_{uv}$$

(12)

Alignment of meshes. Aligning both unfolded surfaces is a matter of translating and rotating the patch until the references $\{p_i, b_{uv(i)}\}$ match as much as possible (see figure 13a). The alignment process is formulated as a Procrustes problem: we seek a rigid transformation matrix to align the reference vertices, stored in a matrix $A = [p_{uv}]_{I_{\text{map}}}$ to their corresponding vertices of the base, stored in a matrix $C = [b_{uv(i)}]$. The derived transformation $R$ is later applied to all vertices of $\mathcal{P}_{uv}$:

$$\mathcal{P}'_{uv} = R \mathcal{P}_{uv},$$

$$R = \arg\min_{R} || R A - C ||^2 \text{ subject to } R^T R = I.$$  

Reverting the planarization. With the meshes aligned in the planar space, it is only required to undo the planarization of the base surface while carrying the thin-plate vertex positions along with it. The vertices $p'_{uv,i,j}$ are reinterpreted as barycentric coordinates respect to the triangles in $\mathcal{B}_{uv}$. Each vertex $p'_{uv,i,j}$ lies inside one base’s mesh triangle $t_i = \{a, b, c\}$, the correlation can be represented as an $|\mathcal{P}_{uv}| \times |\mathcal{B}_{uv}|$ adjacency matrix $A$ so that

$$A_{i,j} = \begin{cases} 1 & \text{if } j \in t_i \\ 0 & \text{otherwise} \end{cases}$$

(14)
The barycentric coordinates \( w_i = w_{ij} \) are first derived within the planar space, and later used to calculate the thin-plate vertices along the original (non-planar) base’s surface:

\[
W = \mathcal{P}_{uv}^{-1} (A B_{uv}), \quad (15)
\]

\[
\mathcal{P}' = WAB \quad (16)
\]

being \( \mathcal{P}' \) the vertex positions of the, now fitted onto the base, thin-plate mesh \( \mathcal{P}' \) (see figure 12).

**Figure 12:** a) Planar base \( B_{uv} \) (wireframe) and aligned thin-plate \( \mathcal{P}_{uv}' \) (square texture). b) Fitted thin-plate \( \mathcal{P}' \) onto the base surface.

### 3.3. Feature reconstruction.

The reconstruction parts from the thin-plate are fitted to the base surface (see figure 13b). The original patch \( \mathcal{P} \) is now deformed to match \( \mathcal{P}' \) at smooth regions, while the features deform to follow the new curvature and position. To minimize distortions, this step is driven by a rigid deformation algorithm, although later, biharmonic methods are also discussed.

**Figure 13:** a) Original patch \( \mathcal{P} \) and fitted thin-plate \( \mathcal{P}' \). The segmentation map is represented over both meshes to illustrate the new disposition of the features.

The ARAP routine utilized is provided by the Libigl library. This transformation takes as input the original mesh \( \mathcal{P} \), the fitted thin-plate \( \mathcal{P}' \) and the index set of vertices excluded from feature regions \( I_{\text{smooth}} \):

\[
T_{\text{rec}}: \mathbb{R}^{3 \times N} \times \mathbb{R}^{3 \times N} \times I_{\text{smooth}} \rightarrow \mathbb{R}^{3 \times N} \quad (\mathcal{P}, \mathcal{P}', I_{\text{smooth}}) \mapsto \mathcal{P}' \quad (17)
\]

Better described by the specific implementation, the reconstruction comprises a rigid deformation \( \mathcal{P} \rightarrow \mathcal{P}' \) for which the fitted thin-plate \( \mathcal{P}' \) provides the boundary condition \( \mathcal{P}_i = \mathcal{P}'_i \) \( \forall i \in I_{\text{smooth}} \). After the reconstruction process, the blended patch \( \mathcal{P}' \) matches the base shape, and features inherit the new positions, orientations and underlying curvature with minimal bending, keeping their shape appearance as much as possible (see figure 13b). Compared to other reconstruction methods such as a biharmonic deformation, ARAP algorithms do a much better work inferring the local surface rotation and, therefore, the new orientations. Biharmonic deformation methods are affected by the original disposition, yielding tilt and distortion on the features (see figure 14).

**Figure 14:** (a): a particular bump on the original, undeformed patch. (b): same bump after a rigid reconstruction of detail. (c): same bump after a biharmonic reconstruction. The bump reconstructed by biharmonic means is tilted because of a substantial change of orientation from the original.

**Stitching vs Morphing meshes together.** At this point, the patch is already wrapped around the blade’s surface, but features are yet stored in separated meshes. Getting a single unique mesh from the fit can be solved through different methods. This process hasn’t been explored in depth during this research because it varies depending on the application. For aesthetic purposes, re-meshing might be allowed as mesh quality is not such an essential constraint, while for simulation, altering the connectivity of the mesh grid might be a concern, as it affects simulation results. It also must take into account mismatches between mesh topology and resolution. For the sake of discussion, we explored a couple of implementations, but an actual evaluation of their effect on mesh quality is kept for further work. These mesh merging methods comprise stitching the meshes together and morphing the base mesh to mimic features after the wrapping of the patch.

Both approaches have pros and cons; stitching the patch’s edge to the base mesh is an option, and it could be performed right after the planar alignment. The triangle quality at the stitching region will be deficient without a post-processing optimization of the meshes to get and smooth transition (figure 15). Most mesh optimization packages could solve this issue (e.g. Pygms, trimesh, GDAL...). Also, note that the original mesh connectivity is lost. Morphing the original mesh is an option that would preserve the original connectivity. The resolution and topology of either mesh might not coincide with the outcome of obtaining a creased, low-affinity representation of the irregularities (figure 16). The issue could be solved by refining the base mesh resolution and applying post-processing optimizations (e.g. smoothing).
Multiresolution Surface Blending for Detail Reconstruction

Figure 15: Planar patch and and based stitched at the edge of the patch, through a Delaunay triangulation. Low-quality meshes are used in this example to stress test the routine.

Figure 16: Wireframe: wrapped patch $P'$. The base model's mesh is morphed to mimic the features by calculating its barycentric coordinates with respect to the patch's triangles. The resolution of the base model is lower than the patch's, and therefore, its representation of the features is limited, adding crease edges.

4. Discussion and example cases.

Some experiments are shown here to demonstrate the routine’s capabilities and limitations. Fitting curved patches over surfaces with acute bending appears to be possible with this routine, as shown in figure 17. Note that the final distribution of the features depends on how the thin-plate is mapped to the base. The particular mapping method described before strives to preserve the relative distances between every feature but, if reallocating individual features was required, the thin-plate could be allowed to stretch to match the imposed allocations.

Scratch patterns can also be transferred from non curved patches, getting the scratches to bend around the leading edge of a blade and then extending through the airfoil. The resolution of the base model is a limiting factor to represent the inherited details, if no refining step is undertaken (see figure 18). To workaround this problem we propose to keep the patch as a separate entity, until a final output geometry is required by the user, with an specified process that might vary depending on the application.

Another interesting situation emerges when dealing with features defined by sharp edges, an extreme case being extruded cube cages over the patch (see figure 19). For these features, the segmentation algorithm still performs, and so do the mapping and reconstruction steps, but light distortions result from the curvature change (see figure 20).

Figure 17: Bump patch transferred to the leading edge of a blade model. a) Patch and base in disposition for blending. b) Thin-plate mapped to the base surface, the change of curvature forces certain bending of the patch to preserve its proportions. This requirement of proportion conservation was bypassed if required by the user. c) Bump features inherited by the base model.

Figure 18: Scratch pattern patch, blended onto the edge of a turbine blade. a) Patch and base model in disposition for blending. b) Thin-plate mapped onto the model’s surface. c) Patch features reconstructed with their new disposition, shown as a wireframe for mesh quality demonstration.

Figure 19: Patch with applied segmentation map. The cages are correctly isolated from the rest of the surface.

These particular bends appear as a result of imposing rigid deformation constraints, because the inherited underlying curvature spreads from the feature boundary and
distributes along their shape. Distortions are hard to be measured in terms of quality, as they depend on the intentions when transferring features, different methods behave differently in terms of preserving the appearance of the original details. Biharmonic methods might avoid bending distortions at the price of being non rotation-invariant and reconstruct the geometry merely as an interpolation process.

Further study is aimed towards extensive testing of the limitations, a more generalized blending algorithm that deals with patches of arbitrary topology and, at some point, further integration into reverse engineering pipelines might be proposed.

5. Conclusions

An algorithm for performing surface feature transfer has been described and demonstrated with triangular mesh models and patches. The tool relies on the user to set reference locations between both shapes and for adjusting a threshold to partition the mesh into soft and irregular regions. More control could be added to allow the patches to be stretched and meet more specific positioning requirements. The resulting algorithm provides reliable results even if dramatic deformations occur, performing exceptionally well in the case of isolated features embedded on a soft supporting surface. In situations where those features fill the entire patch area (figure 21), the reconstructed details get softened if the initial partitioning isn’t generated with careful control. If the features reach the patch boundary, or in other words, if the patch boundary contains important features, it is necessary adding extra computation to blend them with the base. Stability is another issue, as non-linear methods, like rigid deformation, are sensitive to low-quality meshes and may flip polygons during the optimization process or never get to converge. The solution to this problem was to introduce a coarser version of the thin-plate mesh that serves as skinning lattice to simplify deformation steps.

References

Multiresolution Surface Blending for Detail Reconstruction


