Feature Based Retargeting of Parameterized Geometry

Karan Singh
Dept. of Computer Science,
University of Toronto
karan@dgp.toronto.edu

Hans K. Pedersen
Metris
hp@metris.com

Venkat Krishnamurthy
Metris
vk@metris.com

Abstract

This paper presents an approach for mapping layouts of parametric surface patches to a target 3D geometry. Its main contribution is to facilitate the feature based placement of an arbitrary network of patches, assuring that both boundaries and parametric flow conform to features of the target shape. The technique, referred to as dynamic templates, describes the algorithms and interface of a reverse engineering system, Paraform, that integrates techniques relying on a judicious choice of automation and user guided tools. Our approach is based on a novel use of constrained optimization for the fairing of structured surface grids, where grid points can be unconstrained in 3D or constrained to lie within the parameter space of curves, surfaces, or other geometry. We present our results as case studies in large industrial workflow problems, involving the reuse of geometric data.

1. Introduction

While the literature on surface fitting to 3D data is extensive, the problem of mapping a given surface patch layout between two similar data sets has received little attention. This is a common and tedious problem since numerous industrial workflows require repeated fitting of patches to models that geometrically resemble each other. Work and time is wasted because this coherence is ignored, leading to longer product cycles, production costs, and time-to-market overhead.

The approach described in this paper has been commercially used by hundreds of industrial clients since 2000. Our customer base is varied and range from automotive design to manufacturing and entertainment. Although the underlying workflows are very different, all areas have benefited from the reuse of patch layouts. For articulated models used in motion pictures, patch layouts can be reused from character to character. For a line of similar plastic toys, a mould layout can be mapped quickly from the outside to the inside of a shell and also from toy to toy. In automotive design, model changes from year to year are typically minor and incremental, and large savings are to be gained by reusing the CAD parts.

Although the broad and rapid adaptation of dynamic templates in reverse engineering leaves little doubt that the general paradigm is useful, we make no claim that our solution is the only one. Rather, the main contribution of this paper is to point to an important, hitherto overlooked problem in research literature with immediate and important applications.

Fundamentally, our problem statement is to design and reuse parameterizations that conform to feature based constraints, while leaving ultimate control in the hands of the user. Related work can be divided into 2 main categories: feature based parametric mapping techniques and surface fitting.

1.1 Feature based parametric mapping

In image morphing, the objective is to transform one image smoothly into a similar image such that corresponding features line up. Beier and Neely's solution [4] provided an intuitive UI for matching pairs of hand drawn points and line segments in the two images, whereupon the morphing was computed automatically. Similar feature based systems for 3D shape metamorphosis have been proposed [19] (see [9] for an excellent survey of both 2D and 3D morphing).

Closely related to image morphing is the problem of aligning textures with the geometric features of a surface in 3D. Litwinowicz and Miller demonstrated the first interactive feature based texture placement system for the case where a parameterization is provided for the target surface [25].

In general, however, the target surface may not have a suitable parameterization, and the problem of constructing or improving an existing uv-mapping has attracted increasing interest. Bennis et al. [5] presented a user guided optimization technique for placing seams across a parametric surface. A large body of subsequent work presented techniques for constructing 3D mesh parameterization and optimization techniques

for controlling distortion within them [8],[12],[15], [18],[26]. Gu et al. [11] and Levy et al. [23] independently developed algorithms for automatic generation of strategically placed seams, thus presenting the first automatic techniques for addressing the tradeoff between seams and distortion. More recently Alliez et al. [3] describe an approach to remesh geometry using a curvature field and Levy [21] describes an approach based on the global parameterization of a mesh that decouples the geometric properties of the mesh from the parametric structure of the mesh representing the shape.

While many of these techniques allow the user to specify feature based constraints very elegantly [18],[22],[23], they all focus on generating any suitable parameterization and thus offer no guarantees for the topological consistency of patch layout between different models. Recently, this problem of generating *consistent* parameterizations has been addressed in the context of mesh morphing [17] and mesh signal processing [27], the latter of which applied automatic feature detection on each of the two meshes in a preprocessing step. Using automatic derivation of a shared base mesh, it is guaranteed that the two atlases are topologically equivalent (these methods [10],[28] are thus refinements of the idea pioneered by Kent et al. in [14]).

In summary, a large body of varied literature spanning image morphing, texture mapping, and parameterization techniques share the unifying objective of generating a 2D function that optimally matches certain 3D features. It is interesting to observe that the evolution of 3D parameterization methods over the past decade have resulted in several recent approaches that conceptually are closely related to Beier and Neely's image morphing work [4]: given a pair of surfaces, identify their shared features and map one to the other automatically. While all of the above approaches are closely related, they are divided into two schools with respect to automation versus user interaction in the identification of features. This is a fundamental political decision that is critical for the choice of algorithms, data structures and interface. It is our belief that there is no right and wrong answer: no solution fits every application perfectly and it is pivotal to understand and analyze a wide variety of workflows in order to develop a broadly useful application.

To motivate the fundamental approach to be described in the remainder of this paper, we observe that a critical insight underlying the success of Beier and Neely's image morphing approach [4] was that morphing involves a degree of aesthetics that is difficult to quantify mathematically. In such cases, taking advantage of domain expertise from an experienced user is often preferable. To illustrate this point, although a plethora of vision and image processing algorithms have been proposed for extracting the types of features that Beier and Neely outlined with hand drawn line segments, practically all image morphing systems are still heavily based on user interaction.

Years of product research and collaboration with industrial partners led us to this conclusion: there exist situations where a fully automated mapping of the patch layout is desirable (for the parameterization of vast libraries of legacy mesh data, for example), but in the vast majority of observed workflows, domain expertise is critical for generating a useful patch layout. Just like the quality of an image morph is difficult to quantify, the quality of a surfaced model is most often a subjective measure, whether the shape is a movie character, a set of dental braces, a line of toys, or a car body. We thus strive towards a fully automated solution but acknowledge the importance of an expert user and therefore, integrate user control into the design of our proposed solution.

1.2 Surface Fitting

An additional objective of our system is that the parameterization must be fitted to an unstructured data set. This data set may often be incomplete and noisy.

The vast literature on surface fitting is not immediately applicable to our problem since existing techniques are not readily suited to the reuse of surface layouts relying on the domain knowledge from expert users.

Recently, Litke et al. [24] used quasi-interpolation to fit subdivision surfaces to 3D shapes. Although the goal of this work is very different, it is related in that it involves repositioning a surface layout to different target geometry. Our system uses NURBS for commercial reasons, but the ideas presented here could just as well be used with subdivision surfaces.

In other recent work, Weiss et al. [34][35][36] developed a variety of intelligent fitting techniques that makes clever use of geometry analysis to guide the fitting. For certain sub-problems, this work goes further than ours, although it differs philosophically in that automation is given higher priority than data reuse and interactive user guidance.

Perhaps most closely related is the approach for fitting textured meshes representing human faces to images by Blanz et al. [7]. This approach incorporated intelligent high level domain knowledge into an automatic fitting technique. It is thus a highly specialized formulation of our general problem statement; unfortunately it only works for faces and

only if color information is available. Thus, such an application would only be applicable by a small subset of our target user base.
Recently, Zwicker et al.[37] proposed fitting the uv parameters directly to the point cloud. Since points-to-polygons software already is widely used in reverse engineering industry, however, it is reasonable to assume that a mesh representation is already available. The mesh connectivity information allows efficient and easy fitting of a structured grid as proposed by Krishnamurthy et al. [16]. Their *spring* data structure was limited in that every sample in the 2D grid had to lie on the mesh. We lift this restriction with a generalized fitting technique that allows sample points to lie on the mesh, float freely in space, and transition freely between these two states as part of an optimization process.
Levin [18] solves an interesting complementary problem to ours where the target mesh (a subdivision surface) is altered to interpolate a network of curves that are not on the mesh. Levin assumes the existence of a mapping between the curves and vertex paths on the mesh, a problem that is addressed in this paper.

1.3 Overview

Section 2 presents dynamic templates, our approach to the feature based mapping of a parametric patch layout to target geometry. Section 3 illustrates the template approach with a range of real industrial applications, followed by conclusions drawn in section 4.

2 Dynamic templates

The terminology and data structures in this paper are an extension of those used by Krishnamurthy and Levoy [16]. *Face-points* are 3D points constrained to lie on the faces of an underlying mesh. A *face-point curve* is a piecewise linear curve connecting face-points, while a *spring-mesh* is a regular 2D grid of face-points. A spring-mesh is bounded by four boundary curves and its primary purpose in this paper is to capture the parametric structure of the network of curves defining a patch layout. A *patch layout* in this paper is simply a collection of topologically connected curves and springs. *Templates* are thus patch layouts that capture a representative parametric structure and seams for a class of objects. At a high level, dynamic templates can be seen as a combination of automated Beier-Neely style morphing and scattered data fitting. More specifically, patch layouts are mapped from one mesh to another in three steps:

1. The curves and springs are *detached* from the source mesh, creating a *template* of space objects.
2. The template is aligned with the target mesh.
3. The template is mapped to the target mesh.

Section 2.1and 2.2 will describe the first two steps, while section 2.3-2.5 will address the third.

2.1 Template creation (detachment)

Removing the point-on-surface constraint for one or more face-points of a patch layout on an underlying mesh turns it into a template. The unconstrained points are called *space-points*. Our system provides a detach operation, allowing an entire or part of a patch layout or individual face-points to be detached from an underlying mesh, thus creating a template.
Operations like mesh face deletion, that destroy local parameterization, also result in points of a fitted patch layout constrained to the deleted faces being turned into space-points. Parts of the patch layout defined entirely by these space-points become templates.
Further, when fitting a patch layout to meshes with noisy or missing data, it is often useful to detach a few points of a curve or spring where data is unreliable (see Figure 1, 7c,d). Curves and springs consisting of both space and face points are referred to as *hybrid* objects.

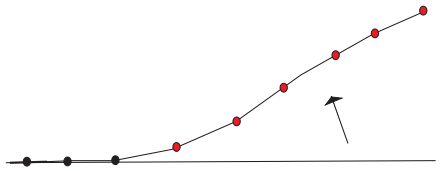

Figure 1. Detach transforms face-points (black) to space-points (grey). Templates may also be created by operations on meshes such as mesh face deletion.

2.2 Alignment

Before the template objects can be mapped to the target mesh, a rough alignment is usually necessary, since it affects the proximity based mapping constraints that we compute in section 2.4. We thus require a rough affine placement of the template object relative to the target mesh, after which we use standard iterative closest point based fitting techniques [6][31] to fine tune the alignment automatically. Models with a skeletal armature, such as those seen in Figure 6 can be automatically aligned by transforming the template skeleton to the target skeleton and deforming the template geometry using a skinning algorithm [1].

2.3 Template attachment

To map a template to the target mesh, our system provides a variety of tools.

Projections. If a low distortion mapping exists between the template and the target mesh, a projection can be a simple and computationally efficient solution: the space points within the template are simply projected onto the target mesh. Figure 2 shows a planar projection of a curve network. In this case, the result appears reasonable in areas where the projection does not introduce excessive distortion. The stretching seen near the corners, however, is a fundamental limitation of this approach. In general, we have found projection useful, primarily, as a local operation in regions where both the template and target mesh are very similar.

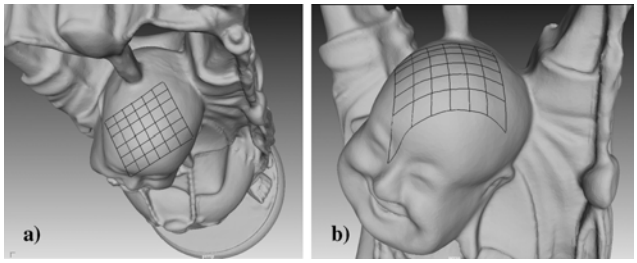

Figure 2.a) Curves before projection b) after projection.

Closest-point. For more complex shapes, such as a car body, many simple projections may be required. As a more practical alternative, a "*snap"* command, allowing a set of template objects (or individual control points within these) to be mapped to the closest point on the target mesh, is supported. In cases where the template can be reasonably well aligned with the target mesh, this method produces good results.

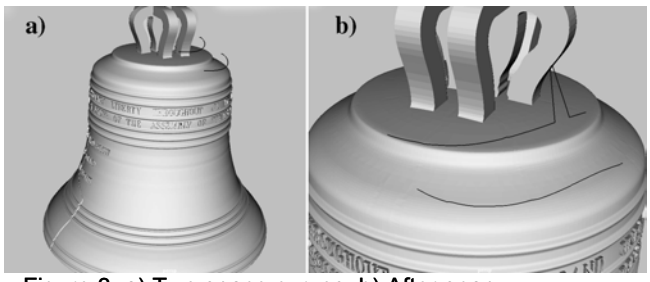

Figure 3. a) Two space curves. b) After snap.

Figure 3 shows an example use of snap. Note how the lower curve, which is closest to the surface in Figure 3a), is mapped smoothly to the mesh, while the other has an abrupt kink after the snap (see Figure 3b). The figure thus shows both the strength and a fundamental limitation of the heuristic. Given a space point, a k-d tree is employed to compute the closest point on the target mesh [29]. We have found a combination of vertex and face k-d trees to be useful. Having the trade-off between efficiency (vertex trees) and precision (face trees) is important when dealing with target meshes consisting of several million polygons.

Heuristic mapping techniques such as these can be helpful in certain cases, but they are generally labor intensive and do not represent a cohesive and general solution. We will now turn to the fitting technique that forms the foundation for our approach.

2.4 Feature-based automated fitting

Analogous to the line pairs used to demark features in [4], our system provides two types of primitives: *anchors* representing point-to-point correspondences, and *curve constraints* representing pairs of matching curves. Conceptually, the only difference in the user interface is that working in 3D allows us the luxury of displaying the source and target models in the same 3D window without much visual clutter. This makes it easier to follow the progress of the fitting process interactively and intervene, if desired, to edit feature constraints: For complex or large data sets, it is easy to forget a feature correspondence, and it would be frustrating for the user to have to start from scratch; thus we opt not to use a fully automatic batch fitting.

Anchors. An anchor is specified by clicking on points on the template objects and the target mesh. Alternatively, any number of anchors can be computed automatically (using the projection and closest-point operators described in section 2.3). Parameters control the spacing between anchors along curves or within springs. Once an anchor is created, its end points can be dragged interactively along the constraining entities (curves, springs, or meshes, in our case).

Curve constraints. A curve constraint is specified by positioning two anchors between the two pairs of curve end-points. We automatically detect when a topologically valid correspondence between a surface and a space curve exists and the resulting constraint is visualized as a pair of dashed lines between the curves (see Figure 4b). Thus, an automatic analysis of the patch layouts is running in the background while the user is editing anchors, and curve constraints pop up and disappear based on the current topology. In the implementation, care must be taken to discard curve constraints that would lead to non-manifold patch configurations or ambiguities of patch orientation.

As in [23], we utilize a "black box" feature detection module to automatically generate curves on the target mesh. More specifically, our application provides extensive functionality for extraction of various types of blends (constant or varying radius).

Figure 9 shows the base of a statue. The color coding in 6a marks feature regions (concave, convex, and flat) computed from the principal curvatures of the mesh. Using this information, positions, types, and parameters are inferred automatically (the details of this algorithm are beyond the scope of this paper). The near-circular curves at the top of the base (Figure 6b) were generated this way. These curves represent the *centerlines* of the blends. In Figure 6c, four anchors have been positioned, the two top ones forming a curve constraint with a segment of the feature curve. As the spring is mapped to the surface, the space curve is mapped exactly to the feature. Note that the resulting spring covers the hole in the mesh; how this is accomplished will be described in section 2.5. Note that although the curve in this example lies on the mesh, it is often convenient to operate with theoretical curves when working with blends, in particular the limit curve as the blend radius approaches 0. Dynamic templates are compatible with both constrained and space curves.

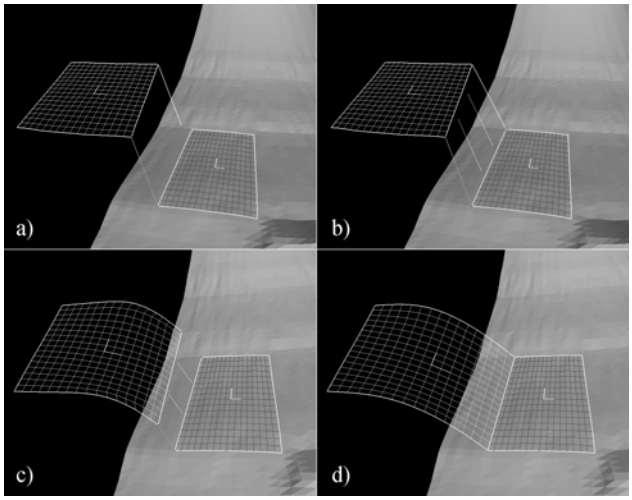

Figure 4. Curve constraints. a) No curve constraint. b) Curve constraint established. c) Partial wrap with curve constraint. d) After fusion of constrained curves.

We now describe the patch layout optimization or "wrap" algorithm. The algorithm combines anchors and curve constraints with the optimization of the 2D parameterization and 3D geometric attributes of the patch layout, into a single energy minimizing equation.

2.5 Patch layout optimization ("wrap")

Using constrained optimization to fit a uv-mapping to a mesh is a well studied problem. A number of proposed energy functions offer tradeoffs between stretching, shearing etc. Our algorithm is designed to allow any algorithm posed as an energy function, such as *conformal maps* [8], to be incorporated into the overall patch layout optimization. Unfortunately, however, existing techniques only allow a mapping to be fitted where the target mesh is defined. The reality of reverse engineering is that meshes obtained by 3D acquisition techniques almost always are imperfect. In particular, holes in under sampled areas, missing or "jaggy" data near boundaries, and localized noise are common artifacts. Existing methods rely on an intermediate mesh editing step (in particular hole filling and smoothing) before the fitting can take place. Such a workflow is inefficient and wasteful, and it offers no guarantees that the edited regions resemble the physical object (note that while this loss of accuracy may not be an issue for entertainment applications, it is critical for most industrial manufacturing and design workflows.) Furthermore, it ignores the fact that, for many manufacturing and design applications, exact CAD data is typically available in the very areas where the scanning process produces artifacts: boundaries and features of manufactured parts are typically constrained by 3D curves and 2D primitives; constraints expressed in terms of curves floating freely in space.

The "wrap" algorithm is thus an energy optimization method that fits a template patch layout to a target mesh with the following properties:

- Satisfy point anchor and curve constraints.
- Conform to regions of reliable target mesh data.
- Cap holes in mesh data and ignore noisy mesh data.

The algorithm implementation is an extension of the method described in [16]: we solve the non-linear optimization problem formulated above by reducing each shape or parameter constraint into a set of resulting forces on the patch layout points, which are moved subject to the forces iteratively. In addition to the parameter forces F_{fair} and F_{arc} described in [16] for face-points, we add shape and constraint based forces F_{tp} and F_{con} for space-points and a force F_{fold} to convert back and forth between face-points and space-points.

The resultant force $\boldsymbol{F_{result}}$ on any point is given by:

$$\boldsymbol{F_{result}} = \alpha * \boldsymbol{F_{fair}} + \beta * \boldsymbol{F_{arc}} + \gamma_{tp} * \boldsymbol{F_{tp}} + \gamma_{con} * \boldsymbol{F_{con}} + \gamma_{fold} * \boldsymbol{F_{fold}},$$

where $\alpha, \beta, \gamma_{tp}, \gamma_{con}, \gamma_{fold}$ as in [16] are scalar multipliers to control the relative magnitude of forces. The new forces F_{tp}, F_{con} and F_{fold} only apply to space-points.

Every iteration, a point is displaced by F_{result} in 3D for a space-point and projected on the mesh along F_{result} for a face-point as in [16].

$\boldsymbol{F_{tp}}$ is a shape-based 3D *thin-plate* force that ensures the unconstrained regions of the fitted surface conform to the curvature at the boundaries of the mesh, while F_{fair} and F_{arc} minimize parametric distortion in both constrained and unconstrained areas of the surface.

$\boldsymbol{F_{con}}$ is a constraint force $||P_c-P||$ that attracts a point P that is anchored or part of a curve constraint towards its constraint point P_c (P is attracted to a corresponding point P_c on a curve constraint based on arc-length).

$\boldsymbol{F_{fold}}$ is a folding force that attracts a space-point that is incident to at least one face-point to the target mesh. F_{fold} measures the local shape deformation between constrained and unconstrained regions of the patch layout. Figure 5 shows a hybrid curve with face-point, P_{seed}, incident to a space point, $P_{foldable}$. Marching along the mesh in the direction obtained by projecting the offset vector to the tangent plane yields a new face point, P_{folded}. F_{fold} represents the quaternion rotation of $(P_{seed}-P_{foldable})$ to $(P_{seed}-P_{folded})$ such that $\gamma_{fold}*F_{fold}$ is a fractional rotation of $P_{foldable}$ towards P_{folded}.

We now address how and when points change state from face-points to space-points and vice-versa.

Moving face-points to space. A face-point is updated by moving it from face to face along the mesh [16]. During this process, if the point moves to a boundary face, the mesh constraint is lifted.

Moving space-points to the target mesh. During the iteration process if the constraint distance $||P_c-P||$ falls below a threshold, the point P is moved to P_c and its state changed from a space-point to a face-point. Similarly, if the angle between vectors $(P_{seed}-P_{foldable})$ and $(P_{seed}-P_{folded})$ falls within a given tolerance, $P_{foldable}$ is moved to P_{folded} and its state changed from a space-point to a face-point. If a hole is encountered while marching along the surface, P_{folded} is obtained by moving in the direction of the tangent vector to the projected curve (ie. P_{folded} will float in space). Since only space-points incident to at least one face-point are folded, the anchor and curve constraint points of a patch layout change state to face-points first. These points then propagate other face-points by folding.

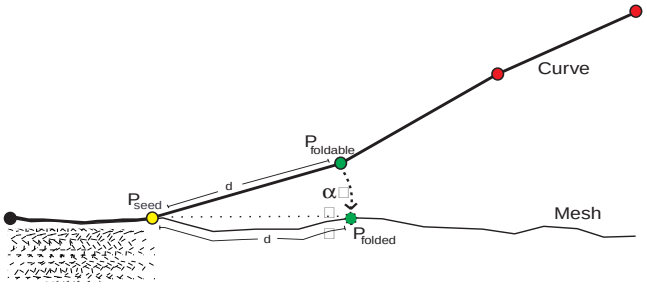

Figure 5. Space-point, face-point result after folding.

The angle tolerance parameter determines how conservative the folding proceeds: if the tolerance is high, many points will be folded during each iteration, likely resulting in excessive noise. If it is too low, no points will be folded. In our implementation, this consideration is hidden from the user using a *simulated annealing* technique, automatically raising or lowering the tolerance based on how many points are folded, in each iteration.The shape deformation heuristic could be implemented in other ways; the key idea is simply to augment the parameter and shape optimization step with a force that captures this optimization criteria.

3 Results

We have dealt with an increasingly diverse set of workflows and practical applications of dynamic templates such as patch layout reuse during industrial design iterations, re-meshing of objects to conform to a given mesh topology and morphing characters. To illustrate our results we pick two case studies, the first from industrial design and the second from film and entertainment, to emphasize the generality of our overall approach.

3.1 Automotive Design

The conceptual design phase in automotive design involves many transitions between the physical and digital manifestations of a model. Design iterations with incremental changes are common due to the high cost of dramatically altering a model.

Figure shows typical changes that would be part of a design iteration. Figure 10a shows a scanned mesh of the front quarter of a car with a set of feature curves and corresponding surfaces. Figure 10b shows the mesh after it has been remodeled. Four different types of changes are indicated. The first and largest is a precise shortening of the fender as a result of a scale operation (seen as the slight offset between template and mesh in insert Figure 10b-i. The second is a rounding of the depression in the fender resulting from smoothing a clay model (Figure 10b-ii). The third is the incremental addition of surface detail such as the tubular structure in the middle of the fender (Figure 10b-iii). Also note the hole on the headlight that could represent deleted surface detail or an artifact of the scanning process (Figure 10b-iv). Figure 10c) shows the result of a simple snap operation (using the k-d tree containing mesh faces (see section 0) for maximum accuracy). Note that the snap only alters the patches that change as a result of the mesh modification. Also note the satisfactory results of snapped templates in the altered regions around the fender and headlight, as long as the amplitude of the modifications to the mesh is small. Finally, Figure 10d shows how allowing points to migrate off the surface, causes points stuck around the hole in the headlight significantly improves the surface quality of the reused patch layout. The workflow shown here is representative of the iterations involved in an automotive design change – iterations that typically take weeks or months and are extremely costly.

Figure 11 shows a more advanced use of hybrid spring optimization. We emphasize that the mesh (shown in

Figure 11a) deliberately was chosen for its poor quality and numerous artifacts to demonstrate the robustness of hybrid springs. Specifically, it contains substantial noise, small holes and cracks, and larger holes present in the actual part. Figure 11b-c) shows a single hybrid spring fitted to the entire mesh. First, a space spring was created the top of the model. Using strategically placed anchors, part of the spring was wrapped to the upper part of the mesh. The boundary curves were then moved to their final position (see Figure 11b), causing the remaining parts of the spring to wrap onto the mesh like a rubber sheet. Notice how the spring covers all holes smoothly and how the parameterization is natural even in highly distorted areas. The final spring is shown in Figure 11c). Note that it was downsampled from an original 2,048x2,048 resolution to make the iso-parameter lines stand out clearly in the image. Although the entire session took nearly one hour using our current system, no existing automatic algorithm known to us would have yielded the same quality parameterization. Until such automation becomes feasible, hybrid springs presents a new way to approach highly challenging surfacing tasks.

3.2 Entertainment

We typically encounter organic forms in entertainment applications. Commonly used workflows involve the data reuse of a given patch layout for the geometric skin of an entire cast of characters. Figure 6a shows a 3D mesh acquired by a full body scanner. The objective of this case study is to map the surfaced model (Figure 6b) to the target (Figure 6d). The workflow presented here is analogous to that used for several 3D morphing sequences in the motion picture "X-Men" [13]. Another major reason for reuse of patch layout on characters is that topology specific skinning, setup and animation developed for one character can also be reused on others.

The first step is to detach the surface network, creating a template. The template corresponding to Figure 6b is seen in Figure 6c. Since this is a complex model with considerable detail in the face and feet regions, the template will be mapped in two stages: first, the detailed regions will be mapped with user guidance and second, the torso, arms, and legs will be automatically attached using curve constraints.

The face and feet are aligned manually with the target mesh (see Figure 7b). Note that the feet are deliberately scaled bigger than the feet of the target mesh. This is for visual clarity to make it easier to see and align the anchors shown in Figure 7c-d. The anchors are computed automatically. The user can then make fine adjustments, adding or deleting anchors interactively.

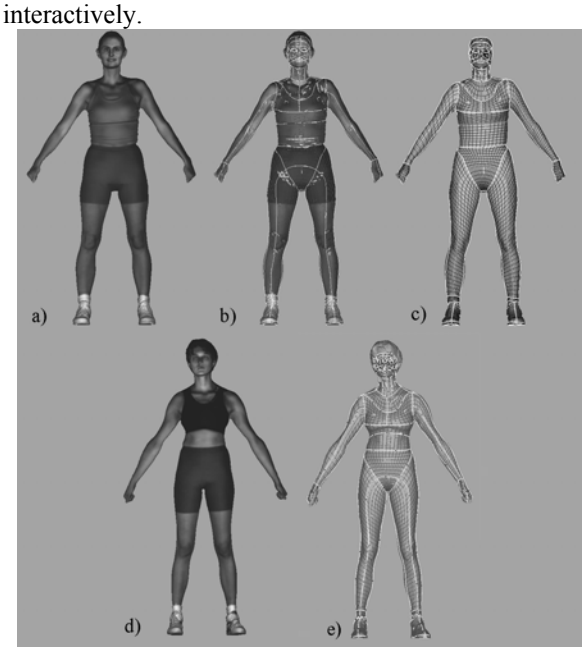

Figure 6. a) Original mesh. b) Original surfaced mesh. c) Template. d) Target mesh. e) Fitted template.

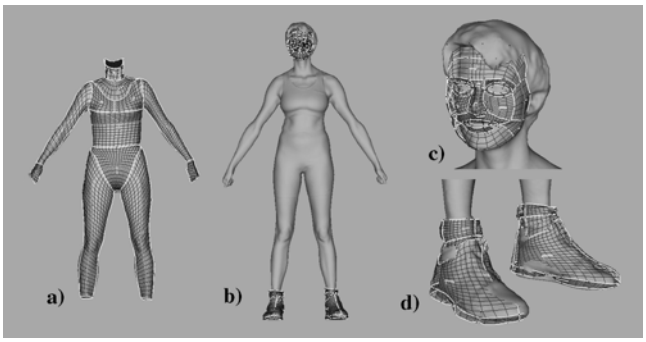

Figure 7 a) Template without face and shoes. b) Target mesh with face and shoe templates aligned. c) Close-up of face (with anchors). d) Close-up of shoes (with anchors).

Once the anchors are in place, the wrap algorithm automatically maps the template to the target mesh.

Figure 12 shows snapshots of the display taken while wrapping the right foot of the model.

Figure 12a shows the template, which has the shape of the original mesh (Figure 12a).

Figure 12b shows the state after a few iterations, where the anchors have started moving towards the mesh. In Figure 12c, the anchors have reached the mesh and the curves and springs have begun the wrapping process.

Figure 12d and

Figure 12e show further progress and a few springs (shown in blue) are now finished surface objects.

Figure 12f shows the result after wrapping. The total time for wrapping the face and feet was 20 seconds (2GHz Pentium IV CPU, nVIDIA GeForce 2 card).
With the most detailed regions in place, the next step is to map the large part of the template corresponding to the body. Figure 13a) shows the differences between the template and target geometries.
Articulated figures, possibly in different poses, are aligned well by deforming the template based on the affine transforms that map the skeleton of the template to that of the target mesh 13b). Curve constraints are used to line up the seams between the template and the already surfaced head and feet 13c+d).
The total time for the mapping between Figure 13b and Figure 13e was 35 minutes: 34 minutes on alignment operations and anchor editing on the face and feet and 1 minute for the wrap algorithm. A further 20 minutes was spent on post-editing to generate a patch layout of the quality that would take an experienced user up to a day to create from scratch.

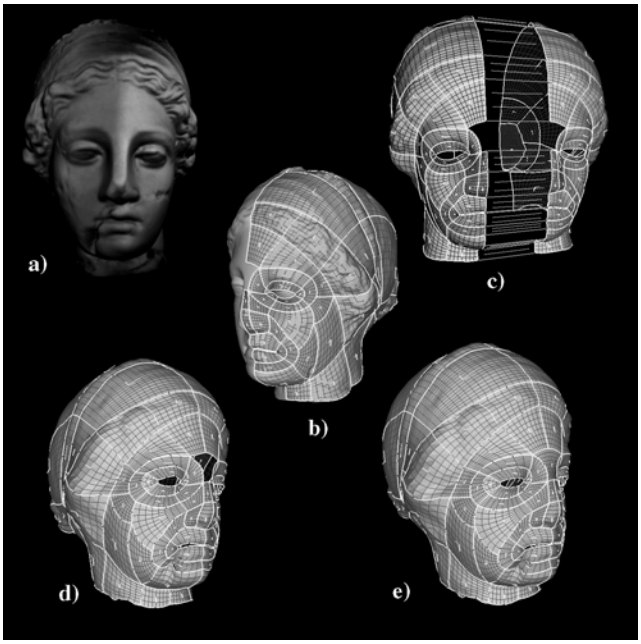

Figure 8. Mirroring example. a) Venus head model . b) One half surfaced. c) Mirrored template with curve constraints. d) Result of wrapping. e) Result after fixing one patch that the crosses plane of symmetry.

Also notable is the reuse of surfacing data on the same model, whereby only half of near symmetric models need be surfaced. The other half is created by mapping the surfaced half as a mirrored template, using the plane of symmetry as a curve constraint. An example of this is shown in Figure 8. This workflow is commonly in both entertainment and design applications. The patch layout also illustrates why automated techniques such as [24] need to be augmented by user expertise to capture the flow of geometry. Capturing geometric flow in a model is essential if the model is to be animated. We capture user expertise here by not only using the patch layout connectivity of the template as in [24] but the patch layout geometry of a manually surfaced model with similar geometric flow.

4 Conclusion

This paper presents an approach for mapping layouts of parametric surface patches to a target 3D geometry. The final aim of our work is to automate as far as possible the feature-based placement of a patch layout, assuring both boundary and internal parametric flow along features of the target geometry. At the same time we acknowledge an element of aesthetic and domain expertise that is best left under user control. We, therefore, provide simple user interactive control over various stages of the mapping process, allowing the user a choice of automation or manual intervention. Our algorithm has evolved over the past three years since its commercial deployment and its current presentation is a robust general technique that is applicable to various workflows and relatively insensitive to incomplete and noisy target geometry. We hope this paper will stimulate research interest in this new area with important industrial applications.
As stated in the introduction, many other solutions to the overall problem as well as its subproblems could be envisioned. Exploring such alternative approaches is an interesting area for future work. Especially interesting is more advanced use of feature extraction: by analyzing the target mesh and the patch layout, more correspondences could be computed automatically, which would further reduce the interactive workload. It is doubtful that a fully automatic approach could ever completely replace an experienced user, but the more intelligence and automation that can be integrated in the retargeting process, the better.

References

[1] Allen B., Curless B., Popovic Z. Articulated Body Deformation From Range Scan Data. Proceedings of SIGGRAPH 02. Computer Graphics Proceedings, Annual Conference Series, pp. 612-619.

[2] Alliez, Pierre and Mathieu Desbrun. Progressive Compression for Lossless Transmission of Triangle Meshes. Proceedings of SIGGRAPH 2001. Computer Graphics Proceedings, Annual Conference Series. pp. 195-202, 2001.

[3] Alliez, P. Cohen-Steiner D., Devillers O., Levy B. and Desbrun M. Anisotropic Polygonal Remeshing. Proceedings of SIGGRAPH 2003. Computer Graphics Proceedings, pp. 485-493, 2003.

[4] Beier, Thaddeus and Shawn Neely. Feature-based Image Metamorphosis. Computer Graphics (Proceedings of SIGGRAPH 92). *26 (2)*, pp. 35-42, 1992.

[5] Bennis, Chakib, Jean-Marc Vézien, Gérard Iglésias, André Gagalowicz. Piecewise surface flattening for non-distorted texture mapping. Proceedings of SIGGRAPH 91. Computer Graphics Proceedings. pp. 237-246, 1991.

[6] Besl, P. and N. McKay. A Method for Registration of 3-D Shapes. Trans. PAMI, Vol. 14, No. 2, 1992.

[7] Blanz, Volker and Thomas Vetter. A Morphable Model for the Synthesis of 3D Faces. Proceedings of SIGGRAPH 99. Computer Graphics Proceedings, Annual Conference Series. pp. 187-194, 1999.

[8] Matthias, E., DeRose T., Duchamp T. Hoppe H., Lounsbery M. and Stuetzle W. Multiresolution Analysis of Arbitrary Meshes. Proceedings of SIGGRAPH 95. Computer Graphics Proceedings, Annual Conference Series. pp. 173-182, 1995.

[9] Gomez, Jonas, Bruno Costa, Lucia Darsa and Luiz Velho. Warping & Morphing of Graphical Objects. Morgan Kaufmann Publishers; ISBN: 1558604642. January 1999.

[10] Gotsman, Craig, Gu Xianfeng and Sheffer Alla. Fundamentals of spherical parameterization of meshes. SIGGRAPH 03. Computer Graphics pp. 358-363, 2003.

[11] Gu, Xianfeng, Steven J. Gortler, Hugues Hoppe. Geometry Images. ACM Transactions on Graphics *26 (3)*. pp. 355-361. 2002.

[12] Guskov, Igor, Wim Sweldens, Peter Schroeder. Multiresolution Signal Processing for Meshes. Proceedings of SIGGRAPH 99. Computer Graphics Proceedings, pp. 325-334. 1999.

[13] Kaufman, Debra. "Simply Marvel-ous", Computer Graphics World, August 2000.

[14] Kent, James R., Wayne E. Carlson and Richard E. Parent. Shape transformation for polyhedral objects. Computer Graphics (SIGGRAPH 92). *26 (2)*, pp. 47-54, 1992

[15] Khodakovsky, Andrei, Peter Schröder and Wim Sweldens**.** Progressive Geometry Compression. Proceedings of SIGGRAPH 2000. Computer Graphics Proceedings, pp. 271-278, 2000.

[16] Krishnamurthy, Venkat and Marc Levoy. Fitting Smooth Surfaces to Dense Polygon Meshes. SIGGRAPH 96. Computer Graphics Proceedings, pp. 313-324, 1996.

[17] Lee, Aaron, David Dobkin, Wim Sweldens and Peter Schröder**.** Multiresolution Mesh Morphing. Proceedings of SIGGRAPH 99. Computer Graphics Proceedings, Annual Conference Series. pp. 343-350, 1999.

[18] Lee, Aaron W. F., Wim Sweldens, Peter Schröder, Lawrence Cowsar and David Dobkin. MAPS: Multiresolution Adaptive Parameterization of Surfaces. Proceedings of SIGGRAPH 98. Computer Graphics Proceedings, pp. 95-104, 1998.

[19] Lerios, Apostolos, Chase D. Garfinkle and Marc Levoy. Feature-Based Volume Metamorphosis. Proceedings of SIGGRAPH 95. Computer Graphics Proceedings, Annual Conference Series. pp. 449-456, 1995.

[20] Levin, Adi. Interpolating Nets of Curves by Smooth Subdivision Surfaces. Proceedings of SIGGRAPH 99. Computer Graphics Proceedings, pp. 57-64.

[21] Lévy, Bruno. Dual Domain Extrapolation. Proceedings of SIGGRAPH 2003. Computer Graphics Proceedings, Annual Conference Series. pp. 364-369, 2003

[22] Lévy, Bruno. Constrained Texture Mapping for Polygonal Meshes. Proceedings of SIGGRAPH 2001. Computer Graphics Proceedings, Annual Conference Series. pp. 417-424, 2001

[23] Lévy, Bruno, Sylvain Petitjean, Nicolas Ray, Jérome Maillot. Least Squares Conformal Maps for Automatic Texture Atlas Generation. ACM Transactions on Graphics. *21 (3)*. pp. 362-371. 2002.

[24] Litke, N., A. Levin, P. Schroeder. Fitting Subdivision Surfaces. IEEE Visualization 2001, pp 319-324.

[25] Litwinowicz, Peter and Gavin Miller. Efficient Techniques for Interactive Image Placement. Proceedings of SIGGRAPH 94. Computer Graphics Proceedings, pp. 119-122, 1994.

[26] Maillot, Jérôme, Hussein Yahia, Anne Verroust. Interactive Texture Mapping. Proceedings of SIGGRAPH 93. Computer Graphics Proceedings, Annual Conference Series. pp. 27-34, 1993

[27] Praun, Emil, Wim Sweldens and Peter Schröder Consistent Mesh Parameterizations. Proceedings of SIGGRAPH 2001. Computer Graphics Proceedings, Annual Conference Series. pp. 179-184, 2001.

[28] Praun, Emil and Hughes Hoppe. Spherical Parameterization and Remeshing. Proceedings of SIGGRAPH 2003. Computer Graphics Proceedings, Annual Conference Series. pp. 340-349, 2003.

[29] Preparata, Franco P. and Michael Ian Shamos. Computational Geometry: An Introduction. Springer Verlag, ISBN: 0387961313, January 1991.

[30] RapidForm. INUS Technology Inc., 2001.

[31] Rusinkiewicz, Szymon and Marc Levoy. Efficient Variants of the ICP Algorithm. 3rd International Conf. on 3D Digital Imaging and Modeling (3DIM), 2001.

[32] Sander, Pedro V., John Snyder, Steven J. Gortler and Hugues Hoppe. Texture Mapping Progressive Meshes. Proceedings of SIGGRAPH 2001. Computer Graphics Proceedings, pp. 409-416, 2001.

[33] Sederberg, Thomas W. and Eugene Greenwood A physically based approach to 2D shape blending. Computer Graphics (Proceedings of SIGGRAPH 92). *26 (2)*, pp. 25-34, 1992.

[34] Varady, T., P. Benko, G. Kos, G. Renner, V. Weiss. Segmentation and Surface Fitting in Reverse Engineering. In Machining Impossible Shapes. Eds. G. Ollling, B. K. Choi, R. B. Jerard, IFIP TC5 WG5.3, Kluwer Academic, 1999, pp. 167-172.

[35] Weiss, V. Reverse Engineering Free-Form Shapes. PhD Dissertation. Geometric Modeling Studies, GML 2000/2, Computer and Automation Research Institute, Budapest, 2000.

[36] Weiss, V., L. Andor, G. Rennder, T. Varady. Advanced Surface Fitting Techniques. Computer Aided Geometric Design. 2002.

[37] Zwicker, Matthias, Mark Pauly, Oliver Knoll and Markus Gross. Pointshop 3D: An Interactive System for Point-Based Surface Editing. ACM Transactions on Graphics *26 (3)*. pp. 322-329. 2002.

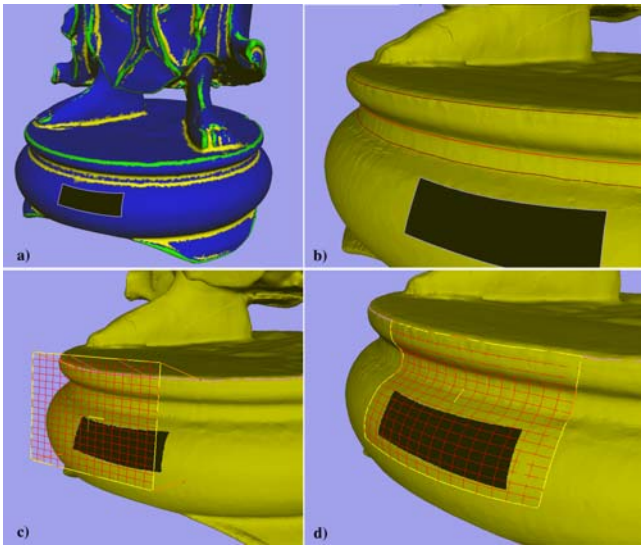

Figure 9. Wrap: a) Mesh with feature map. b) Automatically extracted blend centerlines. c) Space spring with anchors before mapping. d) Spring fully wrapped onto the mesh.

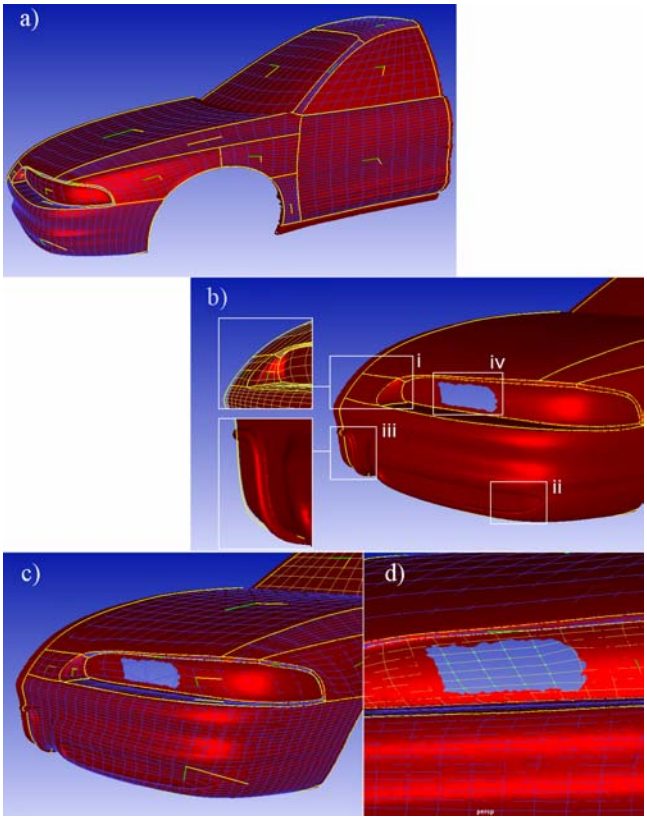

Figure 10 a) Car model with patch layout. b) After edits to the mesh. c) After snap. d) After detach and relax of spring samples near the hole.

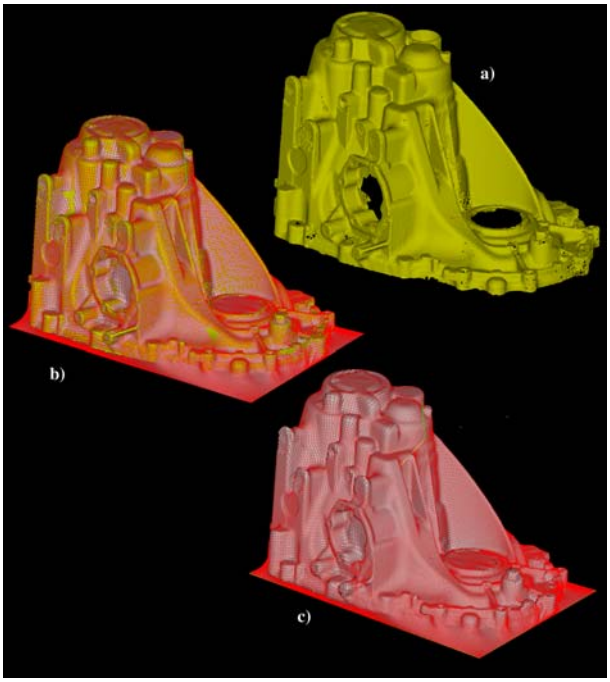

Figure 11 a) Mesh. b) Mesh + Spring. c) Hybrid spring.

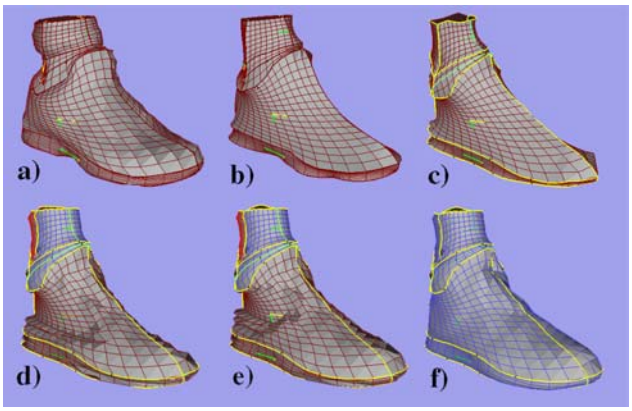

Figure 12.The wrap algorithm: intermediate steps.

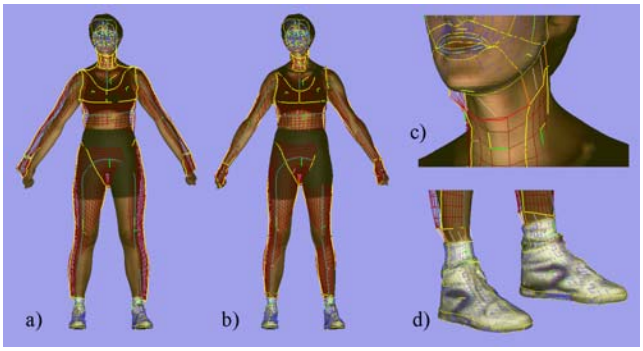

Figure 13. a) Partially surfaced target mesh with the remaining template superimposed. b) After local orientation of articulations. c) Curve constraints between neck and face components. d) Curve constraints between legs and feet.