Morphing for Computer Aided Design

Theodoros Athanasiadis, Ioannis Fudos

Department of Computer Science, University of Ioannina, TR-2011-12, July 6, 2011

Abstract

In this paper, we survey techniques related to morphing with applications to Computer Aided Design. Moreover, we examine several challenges and explore directions for further research in this field.

1. Background and tools

1.1. Mesh Parameterization

Given any two surfaces with similar topology it is possible to compute a one-to-one mapping between them. If one of these surfaces is represented by a triangular mesh, the problem of computing such a mapping is referred to as mesh parameterization. The surface that the mesh is mapped to is typically refered to as the parameter domain. The purpose of mesh parameterization is to obtain a piecewise linear map, associating each triangle of the original mesh with a triangle of a domain. An important goal of parameterization is to obtain *bijective (invertible)* maps, where each point on the domain corresponds to exactly one point of the mesh. The geometric shape of the domain triangles will typically be different than the shape of the original triangles, resulting in angle and area distortion. The distortion is an important factor of the parameterization and applications typically try to minimize the distortion for the whole mesh. Maps that minimize the angular distortion are called *conformal* and maps that minimize area distortion are called *authalic*. Mesh parameterizations have numerous applications in computer graphics such as in Morphing, Mesh Completion, Remeshing, Surface fitting and Texture Mapping.

1.2. Non Linear Optimization

In mathematics, nonlinear optimization is the process of solving a system of equalities and inequalities, collectively termed constraints, over a set of unknown real variables, along with an objective function to be maximized or minimized, where some of the constraints are nonlinear. More specifically in the context of this work we will deal with continuous nonlinear problems of the following form:

$$\min_{x} f(x) \tag{1}$$

Email address: {thathana,fudos}@cs.uoi.gr (Theodoros Athanasiadis, Ioannis Fudos)

$$\text{s.t. } g^L \leq g(x) \leq g^U \tag{2}$$

$$x^L \leq x \leq x^U \tag{3}$$

where the $x \in R^n$ are the optimization variables (possibly with lower and upper bounds, $x^L \in (R \cup -\infty)^n$ and $x^U \in (R \cup +\infty)^n$). The function $f : R^n \rightarrow R$ is called the *objective function.* A vector x satisfying all the constraints of 2 is called a *feasible solution* to the problem. The collection of all such points forms the *feasible region.* The Non Linear Problem (NLP) is to find a feasible point x^* such that $f(x) \geq f(x^*)$ for each feasible point x.

For solving optimization problems like the above there are two categories of algorithms, *global* and *local* optimization methods. Local optimization means that the method attempts to find a *local* minimum, and there is no guarantee that you will get the global minimum for the problem, while global optimization methods try to find the *global* minimum of the *objective function.* In some cases the *local* minimum found is in fact the *global* minimum (convex problems).

Local optimization algorithms generally depend on derivatives of the *objective function* and constraints (gradients and hessians) to aid in the search. For this reason it is useful the function to be real-valued and twice continuously differentiable. There are ways to tackle this strict requirement, but then there is no guarantee that the solver will find a solution. Local optimization also depends on the initial values of the variables, the better the initial values are the faster the solver will converge to a solution. Examples of *local* optimization methods are the *Sequential Quadratic Programming* (SQP) method which is a generalization of Newtons's method for unconstrained optimization, the *augmented Lagrangian* method and the *Interior Point* method.

Global optimization algorithms try to find the best set of parameters to minimize the *objective function.* In general, there can be many solutions that are locally optimal but not globally optimal. Consequently, global optimization problems are typically quite difficult to solve exactly and most methods incorporate probabilistic (random) elements in the algorithms (through random parameter values, etc). More modern methods employ strategies aiming to search the search space in a more intelligent way (*Genetic algorithms*). For *global* optimization there are several algorithms, some known types are *Simulated Annealing*, *Tabu Search*, *Genetic Algorithms* and *Branch and Bound Algorithms.*

1.3. Laplacian Smoothing

In many applications, like the finite element method mesh quality affects numerical stability as well solution accuracy. Therefore, the quality of the mesh triangles is an important factor. There are various existing mesh improvement methods that can roughly be classified into two categories, methods that use topological modifications performed by inserting or removing nodes as well as local reconnection of nodes, and smoothing methods based on relocating existing nodes.

Amongst the second category, the Laplacian smoothing method, has gained popularity due to its simplicity and efficiency. Laplacian smoothing is the most commonly used and straightforward method for mesh smoothing. It simply moves each node to the centroid of the polygon formed by its adjacent nodes. It is a local smoothing algorithm because, in each step, the movement of a node is calculated by using the locations of its adjacent nodes only. However, Laplacian smoothing sometimes can lead to low quality or invalid elements as well as deformation and shrinkage in the case of surface meshes.

In Laplacian smoothing, we can consider a mesh as a spring system. Each edge connecting the central node with its neighboring node can be seen as a linear-spring. Let v_i be the vector from the central node v (x,y) to the i th neighboring node: $v_i = (x_i - x, y_i - y)$. The sum of the spring forces acting on the central node is: $F = K \sum_{i=1}^{k} v_i$ where K is the spring constant, and k is the number of neighboring nodes. When the central node is located exactly at the geometric center of the polygon, the spring forces are balanced out and the spring system is in equilibrium. Therefore Laplacian smoothing can be considered as an iterative way to find this force-balancing state. Below we can summarize the advantages and disadvantages of Laplacian smoothing:

Advantages:

- Computational efficiency
- Easy implementation

Disadvantages:

- Does not always move the node to the optimal position to get the best element quality
- Generation of inverted elements
- Tends to lose element size uniformity if iterated many times
- Tends to yield lower quality elements if iterated more than a few times

Therefore, modified methods have been proposed to circumvent these problems [34].

2. Introduction

2.1. Product Design Challenges

In product design, 3D models are often created in the early stage of product development, because such models are very effective for preliminary design evaluation by the development team. The evaluation of design concepts in the early stage helps products to meet requirements for manufacturing, cost, safety, quality, maintenance, and so on. These geometric models are used as an intermediate object shared between the different groups involved in the design process. It is therefore crucial to use for the design, tools that enable fast and intuitive definitions and modifications of the product geometry.

To this end in recent years, feature-based editing approaches have become popular for the design of products in CAD systems. In Feature-based design the designer does not manipulate directly the surface but the features themselves. Form features with geometric meaning, such as holes, ribs, and slots can be manipulated and parameterized by numerical parameters. Thus, the product definition can rapidly change according to the modifications performed. In addition, form features from existing models can be used to assemble new engineering parts. However, form features do not support all the needs in terms of shape definition during the design process since analytic surfaces are unable to represent free form shapes that are widely used in engineering designs.

Another major challenge in product design is the optimization of design variables in order to meet specific requirements. The optimization is usually performed by specialized optimization software that usually takes as an input a *Finite Element Method* (FEM) mesh model derived from the original CAD design [37]. Nevertheless, every required mesh modification even the slightest one produces a cycle in the design process since it requires a return to the CAD system. This procedure is not only tedious, but sometimes not even possible since the various stages of the design process may be handled by different groups or even by different companies. Therefore, automated shape variation procedures are important to avoid cycles in the product design. One way to achieve slight mesh modifications is through morphing techniques. Consequently, morphing as a tool for small modifications in the optimization process has started to find its way in simple forms in commercial software [7].

It is therefore essential to establish a new editing paradigm for the design of models beyond traditional CAD editing of mechanical parts that will provide robust and efficient 3D mesh deformation and feature pasting that respects design constraints. To this end, our goal is to devise a technique based on morphing that can be used to deform certain parts of a free-form model with respect to design constraints imposed by the product specifications.

Mesh morphing can enable the creation of different variants of mechanical components or assemblies and the ability to rapidly obtain an improved solution without returning in the CAD system for slight changes. Common operations that need to be supported include local modifications on mesh parts, moving features along the contour of a surface, and snapping features to new target geometry. The optimization task can then be performed based on design variables, while following certain design constraints to achieve the final design objective. This way the model can be invoked in an optimization cycle where a multitude of valid models can be created for optimization. The optimization algorithm calculates new values for the design variables and the process is repeated until the optimal solution is found. The constraints to be satisfied may be surfaces that are required to precisely preserve their type and general shape or design parameters such as relative distances between surface areas.

One area of research closely related to our goal is geometric feature cut-and-paste. However, existing methods for free-form surfaces lack the ability to preserve features and constraints, which are required in engineering applications and are more oriented towards visual pleasing results. Existing research for cut-and-paste editing techniques will be reviewed in the next chapter.

2.2. Features in Product Design

Features in a product model encapsulate the engineering meaning or significance of the geometry of portions of the product. By *features* we mean the generic shapes of a product which engineers can associate certain attributes and knowledge useful for reasoning about that product. These high-level modeling entities can be used to link the design rationale with the model. They can also be used to associate geometric and other constraints with the model in terms of high-level characteristics of the part modeled, and to organize constraint propagation after a design change. Hence *features* can be thought of as building blocks for the product definition and make the design process more efficient. It is therefore essential for engineering applications to preserve these *features* after every operation or deformation performed on the product model during the design process.

There are various types of features that are used in product design, some examples are the following:

- *Form features.* They describe portions of a part's nominal geometry.
- *Tolerance features.* They describe geometry variation from the nominal form.
- *Assembly features.* They describe relationships between parts in a mechanical assembly.
- *Functional features.* Non geometric parameters related to function, performance etc.
- *Material features.* Material composition, treatment, condition etc.

In general, the design features can be classified according to the application in which they are used. Form features, tolerance features, and assembly features are closely related to the geometry parts and are called *geometric features.* Current features-based CAD systems mainly address geometric features, in particular form features and some kinds of assembly features and hence our work is mainly focused on form features.

A form feature is defined as a partial shape that has an engineering meaning, such as a round hole. A form feature contains both shape and parametric information. Shape information that describe the general form of the feature can be a set of curves and parametric surfaces. These curves and surfaces are often required to keep their original types, such as circles and cylinders for manufacturing and assembly reasons. Therefore, it is essential to maintain the curvature and the general shape of the form feature by defining a set of appropriate constraints to be satisfied during any deformation. Since in a mesh model a form feature consists of a subset of vertices of the original model, these constraints refer to constraints over the vertices during a deformation. In general, these constraints can be categorized into two groups: *soft constraints* that are approximately satisfied in the least squares sense and *hard constraints* that are precisely satisfied.

Some possible constraints can be:

- positional constraints, where vertices must be fixed at a certain constant position.
- curvature constraints, where the curvature normal is constrained.
- rigidity constraints, where the relative positions of pairs of vertices is constrained.

One important consideration when defining the form feature deformation behavior is how to avoid conflicting or redundant constraints. In this case, the system formed from the constraints can be very hard or even impossible to solve.

2.3. Feature selection and extraction

In order to modify a feature of a model, the feature first needs to be defined. Although, a feature can be manually defined by the user, the procedure becomes tedious and error-prone in complex models. There are different approaches for the segmentation problem that can be categorized into two broad classes *local search* approaches and *global* approaches. In *local search* methods, a region is typically separated into several small regions by exploiting the tangency and the curvature of adjacent faces. However,

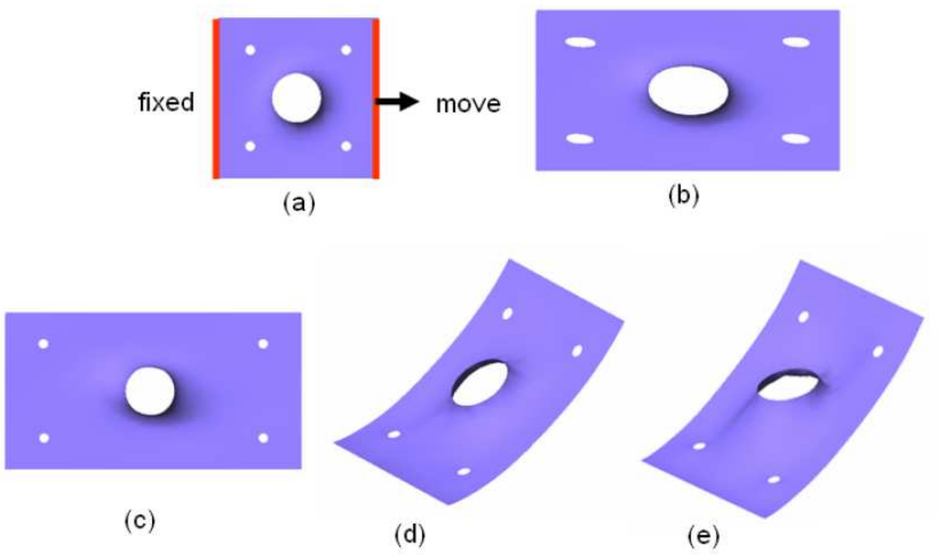

Figure 1: Constrained deformation of a sheet metal panel. (a) Original shape; (b) deformed shape with no form feature constraints; (c) stretched while preserving the shapes of circles; (d) deformed shape with five rotated circles; (e) deformed while preserving the direction of the center circle. Masuda et al [25]

local searches are not always robust for noisy data, such as scanned models. In *global search* methods the segmentation problem is reduced into a global optimization problem. Nevertheless, global methods are often very time-consuming for large models.

2.4. Cut-and-Paste Editing

Cutting and pasting are among the most common operations implemented by image drawing and manipulation software. These operations are a natural way to build complex images from individual components from various sources. In the context of mesh editing, Cut-and-paste editing extracts a characteristic feature form a source model and copies it to a target model. The user usually selects a surface region which is separated into the base surface and the detail surface, only the detail surface is used as a feature to be pasted. The detail surface can be stored either as a height-field or a parametric volume map [10]. The drawback of the height-field representation is that usually general features may be thick or have overhangs and can not be properly represented. To paste the detail surface to a target model, the corresponding vertices of the target model are moved based on the detail map. For 3D models, we should take into account that the smooth attachment of boundaries between a pasted model and its base is sometimes necessary. One possible way to resolve this issue is to perform a union operation between the two models and then apply a blending function along the boundaries of the features [26]. However blending functions for arbitrary meshes is a difficult problem to tackle.

Another approach [36],[35] is the modification of differential coordinates instead of directly changing spatial coordinates. The mesh geometry is then implicitly modified after reconstructing the surface from the differential coordinates. This method has the advantage of reducing deformation artifacts that may appear after the feature pasting.

Existing Cut-and-paste editing methods can be roughly categorized into two broad groups. The first approach uses mesh fusion to blend the source surface and target surface directly [15],[26]. The second one first extracts a base surface as a medium between the source surface and the target surface, and then transfers the details to the target surface via the base surface [10],[6]. The former pays more attention to the smoothness of the boundaries at the joint of the source and target surfaces. The latter focuses on the global deformation of the source surface according to the target surface. Base surface extraction is the most important step of the latter algorithms. On the one hand, it determines the geometry to be pasted, because the details to be pasted are defined as the difference between the feature surface and the base surface. On the other hand, it decides, to some extent, the type of surfaces that can be handled by the algorithm. This is an important factor since one fundamental problem with most of the aforementioned approaches is their inability to deal with non-zero genus features.

2.5. Morphing

Image morphing is a popular technique for creating a smooth transition between two images. It is a special effect used primarily in motion pictures and animation that transforms (morphs) one image into another through a seamless morph sequence. The simplest method for changing an image into another is to cross-fade between the two images. Other approaches use physics based analogs such as the 2D particle system where pixels of one image are mapped onto pixels of the other [29]. There is a class of methods for image morphing that involve image warping around regular (e.g. a sphere of a cylinder) or non regular (e.g. a free-form surface) shapes [27, 29]. More advanced methods include morphing based upon fields of influence surrounding two-dimensional control primitives often called features [5].

Shape morphing is a technique that aims to generate a smooth sequence that transforms a source shape into a target shape. Although we have some quite efficient and effective methods for 2D morphing, the 3D case remains an open problem both in terms of feasibility and efficiency.

Existing methods for 3D morphing can be categorized into two broad classes: *volume based* or *voxel based* [22] and *mesh based* or *structural* [18] approaches. The volume-based approach represents a 3D object as a set of voxels usually leading in computationally intensive computations. The mesh-based approach exhibits better results in terms of boundary smoothness and rendering since the intermediate morphs are represented as volumes and techniques such as marching cube [24] are employed to acquire the final polygonal representation used for rendering. Furthermore, most applications in graphics use mesh-based representations, thus making mesh-based modeling more broadly applicable. However, volume-based methods surpass the mesh based ones in that they can handle the morphing of very different topologies more easily, since volume to volume morphing is a lot similar to image morphing by means of treating voxels instead of pixels.

Although mesh morphing is more efficient as compared to volume-based morphing, it requires a considerable preprocessing of the two considered objects. Mesh morphing involves two steps. The first step establishes a mapping between the source and the target object (correspondence problem), which requires that both models are meshed isomorphically with a one-to-one correspondence. The second step involves finding suitable paths for each vertex connecting the initial position to the final position in the merged mesh (interpolation problem). For performing structural morphing, we can use *boundary representation (Brep)* or *surface representation* in which we represent each object by its surface description, or *volumetric or solid meshes*, for instance tetrahedral representations. In volumetric mesh morphing, it is much easier to maintain robustness and avoid the folding phenomenon. However, volumetric mesh morphing is computationally expensive as compared to surface mesh morphing since the number of elements in the former case is much larger in comparison to the latter case.

3. Related work

3.1. Morphing

Most surface-based mesh morphing techniques employ a merging strategy to obtain the correspondence between the vertices of the input model. The merging strategy may be either automatic or user specified. Kent et al. [18] proposed an algorithm for the morphing of two objects topologically equivalent to the sphere. The algorithm works in two steps, first the two objects are mapped to a sphere and then the two projected topologies are merged. A common topology suitable for interpolation is created. The mapping presented is accomplished by a mere projection to the sphere and thus is applicable solely to star shaped objects.

The main problem with 3D parameterization techniques like [18] is how to find an appropriate mapping over the unit sphere for each of the morphed objects. Several techniques have been proposed to overcome this limitation inspired by physics. In [14] the authors use a spring system to model the mesh and gradually force the mesh to expand or shrink on the unit sphere by applying a force field. Methods using springs do not always produce acceptable mappings especially when handling complex non convex objects.

[1, 2, 38] use a spring-like relaxation process. The relaxation solution may collapse to a point, or experience foldovers, depending on the initial state. Several heuristics achieving convergence to a valid solution are used.

[33, 30, 13] describe methods to generate a provably bijective parameterization of a closed genus-0 mesh to the unit sphere. The projection involves the solution of a large system of non-linear equations. A set of constraints on the spherical angles is maintained to achieve a valid spherical triangulation.

[32] uses a polyhedron realization algorithm that can transform any general polyhedron into a convex one which is isomorphic to the original. The realization consists of two phases, simplification and re-attachment. During the simplification phase, low valence vertices are detached from the vertex-neighborhood graph of the polyhedron one by one, and the corresponding graph is re-triangulated. This step is repeated until a 4-clique results. The second phase starts by first creating a tetrahedron and then the vertices are re-attached to the polyhedron, in the reverse order of their detachment, while maintaining the polyhedrons convexity.

[28] presents a similar method that first simplifies the surface mesh to a tetrahedron while creating a progressive mesh favoring triangles with good aspect ratio and then in similar way reattaches the vertices and simultaneously optimizes positions of the embedded vertices. The positions of the vertices are optimized to minimize a stretch metric.

[31] presents a method that directly create and optimize a continuous map between the meshes instead of using a simpler intermediate domain to compose parametrizations. Progressive refinement is used to robustly create and optimize the inter-surface map. The refinement minimizes a distortion metric on both meshes.

[19] also presents a method that relies on mesh refinement to establish a mapping between the models. First a mapping between patches over base mesh domains is computed and then mesh refinement is used to find a bijective parameterization. One advantage of this approach is that it naturally supports feature correspondence, since feature vertices are required as user input for the initial patch mapping.

[21] uses reeb-graphs and boolean operations to extend spherical parameterization for handling models of arbitrary genus. Each genus-n model is represented as a genus-0 positive mesh and n genus-0 negative meshes. Therefore $n+1$ spheres are required to parameterize these $n+1$ meshes independently, and thus to accomplish the spherical parameterization of genus-n models. Once a consistent embedding is computed for each model the positive meshes and the negative sets are paired. In the case where the number of negative meshes is not equal in the two models, extra pseudo negative meshes are generated to have an equal number of paired negative meshes. For each pair of meshes the morphing sequence is computed independently. Finally, boolean difference operation is applied to subtract each intermediate negative object from an intermediate positive object to obtain the morphing sequence. Existing methods for producing valid spherical embeddings of genus-0 models can be integrated into their framework.

Another method that use reeb-graphs for morphing topologically different objects of arbitrary genus is [16]. The method specifies the correspondence between the input models by using graph isomorphic theory. The super Reeb graph, which has the equivalent topological information to the Reeb graphs of the two input objects, is constructed and used to conduct the morphing sequence.

[20, 23] provide efficient techniques for morphing 3D polyhedral objects of genus-0. The emphasis of the method is on efficiency and requires definition of feature patches to perform 2D mapping and subsequent merging. Their method does not avoid self intersection and requires embedding merging and user intervention for mapping.

An interesting work for volume morphing is based on wavelets is presented in [11]. This is a promising approach whose principle could be applied to surface based morphing. This volume morphing technique yields rather slow algorithm which have time complexity $\Omega(n^3)$ where n is the size of the size of the volume representation.

Finally, we have presented a fully automated feature-based morphing technique that matches surface areas with similar morphological characteristics [4] The technique is based on a spherical parameterization process where the alignment is achieved without user intervention and is based on pattern matching between the feature connectivity graphs of the two morphed objects. Recently we have also presented a very efficient approach for computing spherical parameterizations and detecting feature regions on genus-0 objects [3].

3.2. Mesh fusion

Kanai et al. [15] present an approach to merge two meshes through 3D mesh-based morphing. The basic procedure of their approach is divided into two steps. In the first step, face correspondences are established between the two meshes by which each point on the face of source mesh is mapped to a point on the face of the target mesh. This step is called the *correspondence problem.* The second step, called the *interpolation problem* generates a smooth transition by interpolating corresponding points from the source to the target positions using those correspondences. To address the correspondence problem both the source feature and the target region must be homeomorphic to a disc to compute a harmonic map for each. The parameterizations are then mapped to each other and used to compute a merged topology with the combined topologies. Their main contribution is the development of several methods for resolving the interpolation problem. Further, they propose an algorithm based on three geometrical operations, *rigid transformation*, *scaling* and *deformation*, for adjusting the shape of the two feature boundaries to establish a smooth attachment. The final topology of the pasted feature is computed as a blend between the target geometry and the source geometry.

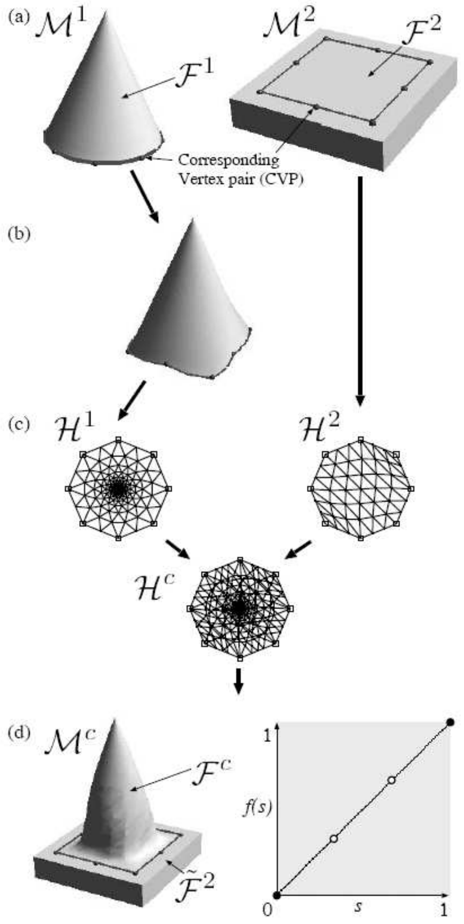

Figure 2: Merging two meshes, kanai et al [15]

Museth et al [26] use Level set models, which are deformable implicit surfaces that have a volumetric representation, to present a framework for editing operators. Their framework supports cut-and-paste by giving the ability to the user to copy,remove and merge models (using CSG operations) and automatically blend the intersecting regions. Since level set models are volumetric, the constructive solid geometry operations can be applied to them in a straightforward manner. Moreover, blending can be applied near the intersection by defining the region of influence based on the distance to the *intersection curve* shared by both input surfaces.

3.3. Implicit representations

Yu et al. [36] present a method for mesh merging,deformation and smoothing. Their approach is based on editing a Poisson-based gradient field. The distinctive feature of their approach is that the mesh geometry is modified implicitly through the gradient field manipulation. More specifically, the technique has tree components, a mesh solver based on the Poisson equation, a gradient field manipulation scheme using local transforms and a generalized boundary condition representation based on local frames. The theoretical foundation is that the Poisson equation is able to reconstruct a scalar function from a guidance vector field and a boundary condition. Thus, with these characteristics editing a function can be achieved by modifying its gradient field and boundary condition and a succeeding reconstruction with the Poisson equation. The surface is reconstructed by solving the least-squares system resulting from discretizing the Poisson equation with Dirichlet boundary conditions. The approach is able to perform merging of two meshes if a correspondence can be established between their open boundaries which serve as Poisson boundary conditions. The correspondence is not automatically established and requires user intervention. One advantage of their approach is that artifacts that can be introduced during deformation can be removed during reconstruction because least-squares minimization tends to distribute errors uniformly across the function.

Sorkine et al. [35] deformation approach is also based on the modification of differential coordinates instead of directly changing spatial coordinates. Their approach use the term *coating* to refer to the mixing of geometric details between two surfaces, and transplanting of a partial surface mesh onto another surface. The *coating* is defined as the difference between the original surface and a low-frequency band of the surface. Thus, coating transfer is the process of peeling the coating of a *source* surface and transferring it onto a *target* surface. Their surface representation is based on the Laplacian of the mesh, by encoding each vertex relative to its neighborhood. To apply their approach to meshes with different topologies a cross mapping is established by parameterizing the meshes over a common domain. To this end, they use the mean-value coordinate parameterization [9] for the mapping of patches homeomorphic to the disk over a unit square. To achieve the cross mapping a registration of the two parts in world coordinates is required. They do not describe how the feature in the source and target surfaces are defined, how they align the features and how they find the required correspondence of the feature boundaries required for the mapping. The vertices of the target surface are obtained by the interpolated Laplacians and the final mesh reconstruction requires solving a linear least-squares system.

3.4. Detail surfaces

Biermann et al. [6] present an approach to copy and paste relief features on multiresolution surfaces. In their approach they use semiregular multi resolution subdivision surfaces as their underlying pasting representation. Each surface is separated into two parts: the *base* surface and the *detail* surface. The goal is to replace the detail part of the second surface with the detail part of the first. The separation is user-guided, a base surface is selected by the user along a single flatness parameter. Subsequently, a Least-Squares fitting procedure is employed to perform the separation. The relief feature details are encoded as a scalar displacement along the normal and a tangential displacement and a subdivision defines a smooth surface recursively as the limit of a sequence of meshes. Each finer mesh is obtained from a coarse mesh by using a set of fixed refinement rules such as Catmull-Clark subdivision rules. Multiresolution surfaces extend subdivision surfaces by introduction *details* at each level. Each time a finer mesh is computed, it is obtained by adding detail offsets to the subdivided coarse mesh. To achieve the feature transfer a parameterization is computed of the source and the target feature over the plane. The idea is to map each surface onto the plane as isometrically as possible and then align the two planar parameterizations, using a linear transformation to compensate for the distortion. Consequently, their approach is difficult to generalize to pasting regions with topology different from that of a subset of a plane. In addition it may result in higher distortion than a direct mapping from one surface to the other. Their approach supports real-time transfer of relier features that are homeomorphic to a disc.

Masuda et al. [10] present a cut-and-paste method based on constrained B-spline volume fitting. Their method is a volume-pasting approach, which paste a parametric volume instead of a height-field. The volume approach allows to deform and paste a feature in the volume even if it contains overhangs or handles. They first compute the pure *feature region* and the surrounding *context region* on the source surface based on user input. Then, the base surface is calculated by approximating the *context region*. For the separation process a global search segmentation method is employed, more specifically a maximum flow minimum cut problem is solved [17]. The base surface can be used to support a cut operation by replacing the *feature region*. To perform the feature transfer an initial parametric volume is defined so that it involves the entire feature region and the control points of the volume are optimized so that the bottom surface corresponds to the base surface. The parametric volume is used to parameterize the *feature region*. To paste a feature the base volume of the feature is deformed to fit the target model. Their method supports copy and paste of features with overhangs and non-zero genus and is able to avoid self-intersections of thick features.

Fu et al [12] also present a method that allows to cut and paste features of non-zero genus onto the target surface. The user first identifies the region of interest by selecting a set of points. A flood fill algorithm is then used in the closed polygon curve implicitly defined by the points selected to get the complete region. The source surface is automatically constructed according to the boundary information of the feature. To achieve this, the user selected boundary is triangulated and then mesh optimization techniques are used to remove degenerate triangles and adjust the density of the base surface. The triangulation is solved by using *dynamic programming* techniques with an $O(n^3)$ running time, with n the number of boundary vertices. However, the base surface is not guaranteed to be self-intersection free and assume that the selected boundary is triangulable. After extracting the base surface, an intrinsic parameterization [8] is computed to map the source feature onto the base surface. Then, the source surface is attached to the target surface, replacing the target region and the feature is reconstructed.

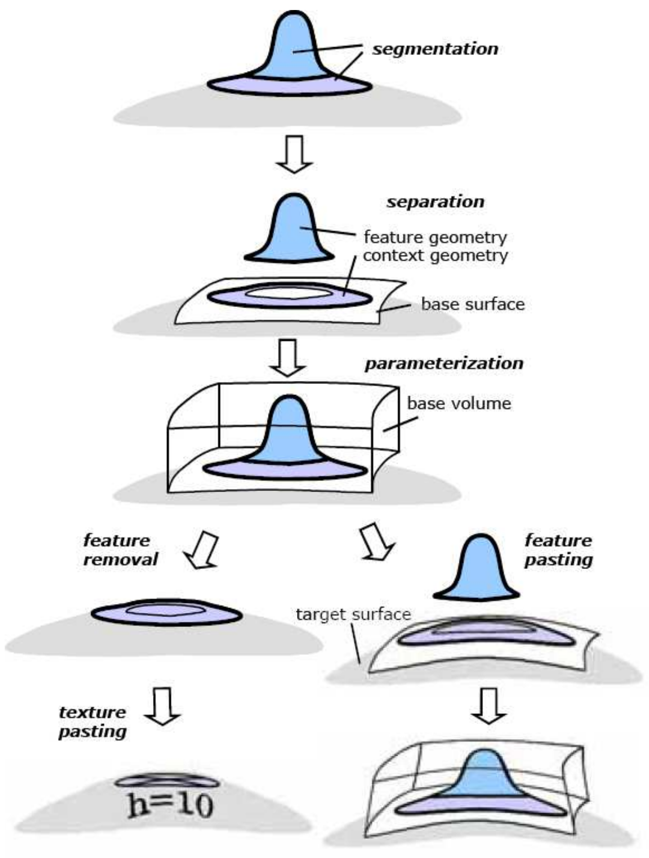

Figure 3: Feature cut and paste, Masuda et al [10]

4. Challenges

4.1. Open problems

Deformation techniques are often used to model the shape of geometric models. However, these techniques are not always convenient for supporting *feature-based* modelling in product design, because they can modify the model shape in unintended ways. Therefore it is essential to establish a new editing paradigm for the design of models that will provide robust and efficient 3D mesh deformation that respects design constraints. To this end, we wish to develop an accurate, robust and efficient 3D mesh deformation technique based on morphing to allow the part deformation of mesh models with respect to engineering constraints. Considering this, a number of important problems remains to be addressed that would allow the realization of 3D mesh deformation based on morphing techniques. Among the most challenging are :

- Automatic feature extraction
- Arbitrary genus features

- Shape constraints
- Trajectory problem
- Interactivity

Feature extraction : Although, a feature can be manually defined by the user, the procedure becomes tedious and error-prone in complex models. In addition, a product shape usually contains a considerable number of form features making this approach impractical for large models, Therefore it is important to establish a feature extraction scheme either fully automatic or user guided to efficiently extract these features.

Arbitrary genus features : In the context of product design, form features of non-zero genus are common (e.g. holes). Consequently, it is essential for any method with application in engineering to be able to handle efficiently and robustly these features. However, most of the existing methods for morphing and cut-and-paste editing are inapplicable to regions with non-zero genus.

Shape constraints : Since our target domain is mesh modelling and form features consists of subset of verticess, shape constraints refer to constraints over these vertices during deformation operations. Using these constraints a system of linear and non-linear constraints over the vertex coordinates can be formed. Nevertheless, special care must be taken to handle inconsistent and redundant shape constraints since in that case it may be impossible to solve the system formed. Furthermore, the efficient solving of these shape constraints is a major challenge.

Trajectory problem : In conjuction with the *shape constraints* problem another major challenge in solving the morphing problem is to find trajectories that corresponding elements traverse during the morphing process. The naive approach to solve the trajectory problem is to choose the trajectories to be straight lines, where every feature of the shape travels with a constant velocity along its line towards the corresponding feature of the target during the morph (*linear interpolation*). Unfortunately, this simple approach can lead to undesirable results. The intermediate morph shapes may contain degenerated elements or self-intersections even though the original models are self-intersection free. Moreover, even if the linear interpolated intermediate morph shapes is free of self-intersections and degeneracies, there may be areas or distances between form features far from those of the source and target. Therefore in the context of *feature based* design, solving the trajectory problem for morphing concentrates on trying to eliminate self-intersections and preserve the geometrical properties of the intermediate shapes. A common approach it to solve this problem in literature is to leave the matter to specialize mesh smoothing or mesh improvement tools. Unfortunately, mesh refinement techniques can lead to the degeneration of the model shape and are unable to preserve the shape of the form features.

Interactivity : Finally, interactive mesh morphing is still a challenging task. Nevertheless, to incorporate these techniques in a real Design environment efficient algorithms for real-time deformation must be developed.

5. Research Directions

5.1. Research goals

The goal of this research is to devise a robust and efficient model editing technique that respects design constraints. To achieve this, the first objective of this research is the evaluation of existing morphing techniques that provide efficient and robust morphing result and can be extended to support constraints. Once an appropriate method is selected, it will be adapted to feature-based design constraints requirements. Another issue to be addressed, not independent from the morphing technique selected, is the problem of identifying the form-features of the model and a corresponding match. In the context of this research, we plan to evaluate methods for the user-driven extraction and matching of form-features from the model. After the features have been identified and matched, appropriate constraints will be established to preserve both the design constraints and to solve the trajectory problem. Finally, the technique devised will be integrated in an editing framework to evaluate it in terms of efficiency and robustness. This way, we will be able to compare it with other related methods.

Table 1 summarises the research goals for morphing in the context of Computer Aided Design.

Research goals	
Phase 1	1. Evaluate and choose a suitable feature-based morphing technique 2. Adapt the technique to feature-based design requirements
Phase 2	1. Address the problem of automatic or user-driven feature extraction 2. Solve the registration problem to match the corresponding features 3. Define appropriate shape constraints for *form-features* 4. Address the morphing trajectory problem
Phase 3	1. Develop an interactive application 2. Evaluate the application in terms of efficiency and quality 3. Compare with other related methods

Table 1: Research goals

References

[1] M. Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1):26–37, 2000.

[2] M. Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–197, 2002.

[3] Th. Athanasiadis and I. Fudos. Parallel computation of spherical parameterizations for mesh analysis. *Computers & Graphics*, 35(3), 2011. Shape Modeling International 2011.

[4] Theodoros Athanasiadis, Ioannis Fudos, Christophoros Nikou, and Vasiliki Stamati. Feature-based 3d morphing based on geometrically constrained sphere mapping optimization. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1258–1265, New York, NY, USA, 2010. ACM.

[5] T. Beier and S. Neely. Feature-based image metamorhosis. In *Proceedings of SIGGRAPH 1992*, volume 26(2), pages 35–42. New York, Published as Computer Graphics, July 1992.

[6] Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Trans. Graph.*, 21(3):312–321, 2002.

[7] Beta CAE. ANSA, Pre-processing software, http://www.beta-cae.gr/ansa.htm.

[8] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. In *Eurographics 02*, pages 209–218, 2002.

[9] Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20:2003, 2003.

[10] Yoshiyuki Furukawa, Hiroshi Masuda, Kenjiro T. Miura, and Hiroyuki Yamato. Cut-and-paste editing based on constrained b-spline volume fitting. *Computer Graphics International Conference*, 0:222, 2003.

[11] T. He, S. Wang, and A. Kaufman. Wavelet-based volume morphing. In *Proceedings of Vizualization 1994*, pages 85–92, Washington D.C., October 1994. ieee.

[12] Chiew-Lan Tai Hongbo Fu and Hongxin Zhang. Topology-free cut-and-paste editing over meshes. In *GMP '04: Proceedings of the Geometric Modeling and Processing 2004*, page 173, Washington, DC, USA, 2004. IEEE Computer Society.

[13] Ilja Friedel and Peter Schröder and Mathieu Desbrun. Unconstrained spherical parameterization. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 134, New York, NY, USA, 2005. ACM.

[14] T. Kanai, H. Suzuki, and F. Kimura. 3d geometric metamorhosis based on harmonic map. In *Proceedings of the 5th Pacific Computer Graphics and Applications*. IEEE, 1997.

[15] Takashi Kanai, Hiromasa Suzuki, Jun Mitani, and Fumihiko Kimura. Interactive mesh fusion based on local 3d metamorphosis. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 148–156, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[16] P. Kanonchayos, T. Nishita, S. Yoshihisa, and T. L. Kunii. Topological morphing using reeb graphs. In *CW '02: Proceedings of the First International Symposium on Cyber Worlds (CW'02)*, page 0465, Washington, DC, USA, 2002. IEEE Computer Society.

[17] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 954–961, New York, NY, USA, 2003. ACM.

[18] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *Proceedings of SIGGRAPH 1992*, volume 26(2), pages 47–54. New York, Published as Computer Graphics, July 1992.

[19] Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graph.*, 23(3):861–869, 2004.

[20] T. Y. Lee and P. H. Huang. Fast and intuitive metamorphosis of 3d polyhedral models using smcc mesh merging scheme. *IEEE Transactions of Visualization and Computer Graphics*, 9(1):85–98, 2003.

[21] Tong-Yee Lee, Chih-Yuan Yao, Hung-Kuo Chu, Ming-Jen Tai, and Cheng-Chieh Chen. Generating genus-n-to-m mesh morphing using spherical parameterization: Research articles. *Comput. Animat. Virtual Worlds*, 17(3-4):433–443, 2006.

[22] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature-based volume metamorhosis. In *Proceedings of SIGGRAPH 1995*, pages 449–456. ACM SIGGRAPH, 1995.

[23] C. H. Lin and T. Y. Lee. Metamorphosis of 3d polyhedral models using progressive connectivity transformations. *IEEE Transactions of Visualization and Computer Graphics*, 11(1):2–12, 2005.

[24] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH 87, published as Computer Graphics*, pages 163–169. ACM SIGGRAPH, 1987.

[25] Hiroshi Masuda, Yasuhiro Yoshioka, and Yoshiyuki Furukawa. Preserving form features in interactive mesh deformation. *Comput. Aided Des.*, 39(5):361–368, 2007.

[26] Ken Museth, David E. Breen, Ross T. Whitaker, and Alan H. Barr. Level set surface editing operators. *ACM Trans. Graph.*, 21(3):330–338, 2002.

[27] M. Oka, K. Tsutsui, O. Akio, K. Yoshitaka, and T. Takashi. Realtime manipulation of textured mapped surfaces. In *Proceedings of SIGGRAPH 83, published as Computer Graphics*, volume 21(4), pages 181–188. ACM SIGGRAPH, Anaheim, 1983.

[28] Emil Praun and Hugues Hoppe. Spherical parametrization and remeshing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 340–349, New York, NY, USA, 2003. ACM.

[29] M. Rosenfeld. Special effects production with computer graphics and video techniques. In *Proc. SIGGRAPH 87, Course Notes #8.* ACM SIGGRAPH, July, Anaheim 1987.

[30] Shadi Saba, Irad Yavneh, Craig Gotsman, and Alla Sheffer. Practical spherical embedding of manifold triangle meshes. In *Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 258–267, 2005.

[31] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. Inter-surface mapping. *ACM Trans. Graph.*, 23(3):870–877, 2004.

[32] Avner Shapiro and Ayellet Tal. Polyhedron realization for shape transformation. *The Visual Computer*, 14(8/9):429–444, 1998.

[33] A. Sheffer, C. Gotsman, and N. Dyn. Robust spherical parameterization of triangular meshes. *Computing*, 72(1-2):185–193, 2004.

[34] T. Shou and K. Shimada. An angle-based approach to two-dimensional mesh smoothing. In *Proceedings of the 9th International Meshing Roundtable*, pages 373–384, 2000.

[35] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM.

[36] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 644–651, New York, NY, USA, 2004. ACM.

[37] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method*, volume 1-3. Butterworth Heinemann, 2005.

[38] M. Zwicker and C. Gotsman. Meshing point clouds using spherical parameterization. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, Zurich, June 2004.