# A Framework for Geometric Warps and Deformations

TIM MILLIRON, ROBERT J. JENSEN, and RONEN BARZEL
Pixar Animation Studios
and
ADAM FINKELSTEIN
Princeton University

We present a framework for geometric warps and deformations. The framework provides a conceptual and mathematical foundation for analyzing known warps and for developing new warps, and serves as a common base for many warps and deformations. Our framework is composed of two components: a generic modular algorithm for warps and deformations; and a concise, geometrically meaningful formula that describes how warps are evaluated. Together, these two elements comprise a complete framework useful for analyzing, evaluating, designing, and implementing deformation algorithms. While the framework is independent of user-interfaces and geometric model representations and is formally capable of describing *any* warping algorithm, its design is geared toward the most prevalent class of user-controlled deformations: those computed using geometric operations. To demonstrate the expressive power of the framework, we cast several well-known warps in terms of the framework. To illustrate the framework's usefulness for analyzing and modifying existing warps, we present variations of these warps that provide additional functionality or improved behavior. To show the utility of the framework for developing new warps, we design a novel 3-D warping algorithm: a *mesh warp*—useful as a modeling and animation tool—that allows users to deform a detailed surface by manipulating a low-resolution mesh of similar shape. Finally, to demonstrate the mathematical utility of the framework, we use the framework to develop guarantees of several mathematical properties such as commutativity and continuity for large classes of deformations.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation

General Terms: Algorithms

Additional Key Words and Phrases: Deformation, warp

## 1. INTRODUCTION

Warps and deformations have found a wide variety of applications in modeling, animation, and special effects.[1] Driven by a spectrum of applications, a host of warps have been been described in the literature, each posing a seemingly separate set of problems and solutions. In this article, we cast many of these warps into a single common framework, giving us a unifying base to facilitate the analysis, design,

[1]The words "warp" and "deformation" appear in various forms in the literature to describe parameterized reshaping of objects or images. In this article, as in much of the literature, we will use the terms interchangeably.

This work was supported by NSF CAREER award 98-75562 and Pixar.
Authors' addresses: T. Milliron, R. Jensen, and R. Barzel, Pixar Animation Studios, 1200 Park Ave., Emeryville, CA 94608, e-mails: {tm, rj}@pixar.com; ronen@barzel.org; A. Finkelstein, Computer Science Department, Princeton University, 35 Olden Street, Princeton, NJ 08544-2087, e-mail: af@cs.princeton.edu.

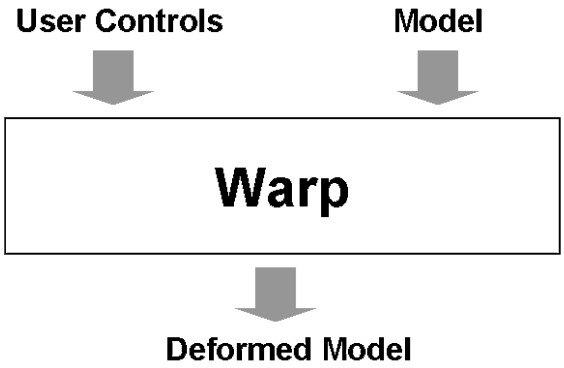


Fig. 1. A generic "black box" view of user-controlled deformations. Given user controls to define the warp, the deformation computes a point-to-point warp mapping from model points to deformed model points.

and implementation of warping algorithms. Within the framework, it is easy to see ways to extend and fine-tune a specific warp by understanding the mathematical expression of the warp and how similar problems have been addressed in other contexts. The framework also makes it easier to design new warps, either by using the framework itself as a design context, or by mixing and matching components from existing warps. Furthermore, the framework is amenable to mathematical analysis and serves as a tool for proving mathematical properties for large classes of deformations. Finally, there are several implementation advantages, such as an elegant software architecture in which it is easy to try out ideas for new warps or to make application-specific warps, and in which code can be reused for many deformation algorithms.

The framework is composed of two parts: first, a concise set of geometrically meaningful equations that define how warps are computed; and second, a generic modular algorithm structure into which individual deformations "plug in" specialized components. These two elements work in concert to form a complete picture of how warps are constructed and evaluated, and to provide a geometric, conceptual, and mathematical base on which to examine and design warping algorithms. We are most interested in interactive applications, such as modeling and animation systems, in which a user deforms an object or image. To this end, we focus on *user-controlled* warps; while the framework is capable of describing other types of warps, these will be largely neglected here.

To build an interactive application supporting deformations, one must address questions of model representation, user interface, and efficient sampling of the resulting warped model—issues that lie beyond the scope of this article. Here, we focus on the core of the deformation: a "black box" that takes a model and user-controlled values to define the warp, and produces a deformed model via a point-to-point mapping from undeformed points to deformed points (see Figure 1). The framework provides an inner structure for that box specific enough to provide a powerful tool for analysis and design, yet general enough to accommodate any model representation or user-control paradigm, and to describe most warping algorithms concisely and elegantly.

The remainder of this article is organized as follows: Section 2 provides an overview of previous frameworks and a more in-depth background of warps and deformations, placing the framework in the context of the literature. In Section 3, we develop our framework using feature-based warps, leading to a description of the complete framework. In Sections 4, 5, and 6, we illustrate the descriptive power of the framework by expressing four well-known warps in terms of the framework, show the analytical and conceptual advantages of the framework by extending three of these warps to achieve greater flexibility and improved behavior, and demonstrate the ease with which warps can be designed in the

framework by developing a completely novel *mesh warp*. Section 7 exercises the mathematical utility of the framework by developing conditions for several important mathematical properties for warps expressed in the framework. Finally, Section 8 closes with a broader discussion of the framework and areas for future research.

## 2. RELATED WORK

### 2.1 Previous Frameworks

To our knowledge, only two previous frameworks exist. Barr [1984] describes a framework in which warps take the form of geometric transformations—rotations, translations, and shears—parameterized by some function of the points to be deformed. This framework is not general enough to describe many user-controlled warps; in particular, the large class of deformations defined by geometric features. The second framework, proposed by Bechmann [1994], expresses deformations defined by feature points along with weights defining the influence of each point on the resulting deformation. This framework generalizes freeform deformations and other point-based warps, but does not provide support for more complex features. Moreover, Bechmann's framework relies on discrete matrix equations, which are of limited use for mathematical analysis.

### 2.2 Warps and Deformations

The schematic in Figure 1 suggests a taxonomy of warps based on the structure of their two inputs: model representation and user controls. Here, we classify warps in the literature by these two areas.

*Model Representation.* One way to classify warps and deformations is by examining the model representation for which they are designed and the dimensionality (2-D or 3-D) of that representation. One model representation is a 2-D image—a uniform grid of pixels in the plane. An image warp maps pixel coordinates to new locations, carrying the color value at each pixel to its new location. Such warps include Beier and Neely's [1992] feature-based warp, Lee et al.'s [1995] snakes and freeform deformations, spline and polynomial basis function deformations [Wolberg 1990], and Litwinowicz and William's [1994] energy-minimization method. Voxel-based volumes extend the uniform 2-D pixel grid to 3-D, where a uniform 3-D grid of scalar values defines the volume of an object. Warps on volumes include Barr's [1984] solid deformations, freeform deformations (FFDs) [Coquillart 1990; Greissmair and Purgathofer 1989; Hsu et al. 1992; MacCracken and Joy 1996; Sederberg and Parry 1986], Lerios et al.'s [1995] extension of Beier and Neely image warping, and Cohen-Or's [1998] distance-field metamorphosis. Finally, warps employed in computer animation tend to act on surfaces in 3-D including polygonal meshes, Bezier or B-spline patches, NURBS, and subdivision surfaces. Each of these representations encodes a 2-D manifold in 3-D as a polygonal mesh with vertices and faces. Examples of surface warps include curve-feature warps [Corrêa et al. 1998; Lazarus et al. 1994; Singh and Fiume 1998], mesh-based warps [van Overveld and Stalpers 1997], and Decaudin's [1996] warps based on simple convex shapes. Our framework is independent of model representation, since it considers all warps as point-to-point mappings in either 2-D or 3-D.

*User Controls.* Warps may also be classified by the type of interface they provide to the user. In some warps, the user manipulates values that are used directly in numerical, nongeometric expressions to compute the deformed model. For example, some warps rely on the specification of coefficients of polynomial basis functions: points are considered as combinations of these basis functions, and the warp changes the position of points based on specified coefficients [Wolberg 1990]. Other warps allow the user to specify values that have geometric meaning. For example, the user of a Barr taper deformation

[Barr 1984] specifies an axis along which the taper will occur as well as a function whose value represents the amount of taper at a position along the axis (see Figure 12). Warp interfaces based on radial basis functions [Cohen-Or et al. 1998; Wolberg 1990] generally provide geometric control, as well. Most warps, however, are *feature-based*—defined by a set of geometric (*source*, *target*) feature-pairs, such as points and curves. For example, FFDs [Coquillart 1990; Greissmair and Purgathofer 1989; Hsu et al. 1992; MacCracken and Joy 1996; Sederberg and Parry 1986] Bechmann's [1994] warping framework and others [Borrel and Rappoport 1994; Cohen-Or et al. 1998] all use feature points to define the deformation. Beier and Neely [1992], Lee et al. [1995], and Lerios et al. [1995] use line-segment features. Curve features have been used by Corrêa et al. [1998], Lazarus et al. [1994], Singh and Fiume [1998], and Litwinowicz and Williams [1994]. Recent work has focused on using skeletal meshes [van Overveld and Stalpers 1997] and other geometric models [Decaudin 1996] as features. Our framework is independent of the type of user-controls used in a warp, but is designed with feature-based deformations in mind, and expresses these warps most elegantly.

## 3. THE FRAMEWORK

This section presents our framework for warps and deformations. We will develop the framework by considering increasingly complex feature-based warps, generalize this development to other warps, and conclude with a discussion of the complete structure of the framework.

### 3.1 Preliminaries

Before developing the framework, it will be helpful to define some concepts, terminology, and notation. First, recall from Section 1 that every deformation is a mapping from points to points. Also, every geometric model is "point-valued," in the sense that deformations operate on model points. A convenient way of formalizing this concept is to consider a geometric model to be a point-valued[2] function $M(u)$ defined on some domain $U$:

$$M(u) = \textit{model point}, \textit{ for } u \in U.$$

The specifics of the model representation depend on the type of model being warped. For concreteness, two common model representations are:

*Vertex Mesh.* $U$ is the set of $n$ vertex indices $\{1, \ldots, n\}$, and $M(u)$ is the vertex indexed by $u \in U$. The warp will move the vertices to new positions in space.[3] The mesh may represent a polygonal model or the control vertices of a smooth surface. Note that the connectivity of the mesh is maintained through the warp: only points are moved.

*Image.* $U$ is the 2-D plane, and $M(u) = u$ is defined at every point $u$ in the plane. The warp will create a mapping between pixel coordinates in the original and deformed images, for subsequent sampling of the image data.

For brevity, we will often refer to the model function $M(u)$ simply as $M$.

We express the output of a warp as a *deforming function* $\mathcal{D}(u, M)$, which is a function defined on the domain $U$, taking a value $u \in U$ and the model $M$ as input, and yielding a transformation as output. To compute the warped model $\tilde{M}(u)$, which is a point-valued function defined on the domain $U$ just

[2]In general, we denote all points and point-valued functions by capital letters, as with the model $M(u)$. See Table I for a complete listing of notation used in this paper.

[3]We generally sample the deformation only at vertices. In principle, however, every point in the continuous surface could be warped, whether the mesh represents a polygonal mesh or the control points of a smooth surface.

Table I. Table of Notation

| Description | Denotation | Examples |
|---|---|---|
| scalars & scalar-valued functions | $a$ | $s_i(v, u, M)$ $w_i(v, u, M)$ |
| points, tuples, domains & like-valued functions | $A$ | $U$ $M(u)$ $\tilde{M}(u)$ $F_i$ $V_i$ |
| transformations & transformation-valued functions | $\mathcal{A}$ | $\mathcal{D}(u, M)$ $\mathcal{T}(v)$ |
| transformation application | $\langle A \rangle$ | $\mathcal{T}_i(v)\langle M(u)\rangle$ $\mathcal{D}(u, M)\langle M(u)\rangle$ |
| sets | $\bar{A}$ | $\bar{F} = \{F_i\}$ $\bar{s} = \{s_i\}$ $\bar{w} = \{w_i\}$ |
| matrices | $\mathbf{A}$ | $\mathbf{T}$ $\mathbf{I}$ |
| source-target feature pairs | $a, a^\star$ | $L_i, L_i^\star$ |

as $M(u)$ is, the deforming function is evaluated at $u$ and applied to the point given by evaluating the model at $u$:

$$\tilde{M}(u) = \mathcal{D}(u, M)\langle M(u)\rangle.$$

Here, and in the rest of this article, transformations and transformation-valued functions are denoted by calligraphic font—as in $\mathcal{D}(u, M)$—and the application of a transformation $\mathcal{T}$ to a point $P$ is denoted by $\mathcal{T}\langle P\rangle$ (see Table I). We sometimes denote the deforming function $\mathcal{D}(u, M)$ by the shorthand $\mathcal{D}$, and denote the application of a deforming function $\mathcal{D}$ to a model $M$ by $\mathcal{D}\langle M\rangle$.

As discussed in Section 2.2, one important class of warps are those defined by geometric (*source*, *target*) feature pairs: such warps are called *feature-based warps.* Since the framework is designed with feature-based warps in mind, it is helpful to develop a specialized terminology and notation for the defining inputs of these warps. To represent the source/target feature mappings that define a feature-based warp, we use a collection of *feature specifications*: tuples of values that include a source feature, the corresponding target feature, and related parameters to control the deformation. For example, in a warp with point features, if $P_i$ is the position of a source feature-point and $P_i^\star$ is its target position, we would have a feature specification $F_i = (P_i, P_i^\star)$. Likewise, for a warp that uses curves $C_i(v)$ as source features and two parameters to control the warp, we would have a feature specification $F_i = (C_i, C_i^\star, \alpha_i, \beta_i)$. Here, and in the rest of this article, we denote the target feature corresponding to source feature $f$ by $f^\star$ (see Table I).

### 3.2 Simplest Case: Point Features

To begin developing the framework, we start with a simplistic example: a feature-based warp defined by a single feature point. In this warp, $P$ is the source position of the feature-point and $P^\star$ is the target position, giving us the feature specification:

$$F = (P, P^\star).$$

The most obvious deforming function that moves $P$ to $P^\star$ is a constant translation[4] $\mathcal{T} = P^\star - P$ (see Figure 2). Thus,

$$\mathcal{D}(u, M) = \mathcal{T}. \tag{1}$$

A natural way of extending the translation to yield a more useful deformation is to vary the effect of the translation across the model. To do this, we introduce a *strength field* $s(u, M)$, that indicates how "strongly" the translation will be applied to the point $M(u)$. That is, $s(u, M)$ for $u \in U$ is a scalar-valued

[4]That is, $\mathcal{T}$ is a transformation that adds the vector $P^\star - P$ to every point. Here and elsewhere in this article, the notation $\mathcal{T} = P^\star - P$ serves as a convenient shorthand.

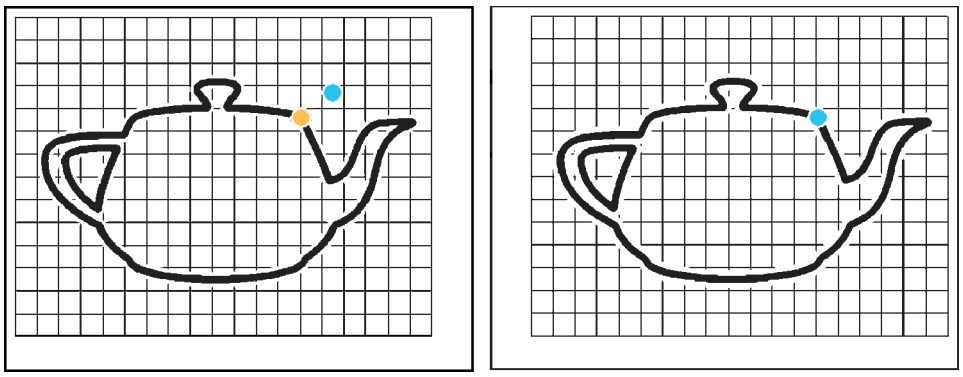

Fig. 2. A very simple feature-point warp. *Left*: Undeformed model, with source (orange) and target (blue) feature points. *Right*: Deformed model, translated so that the source point moves to the target (blue).

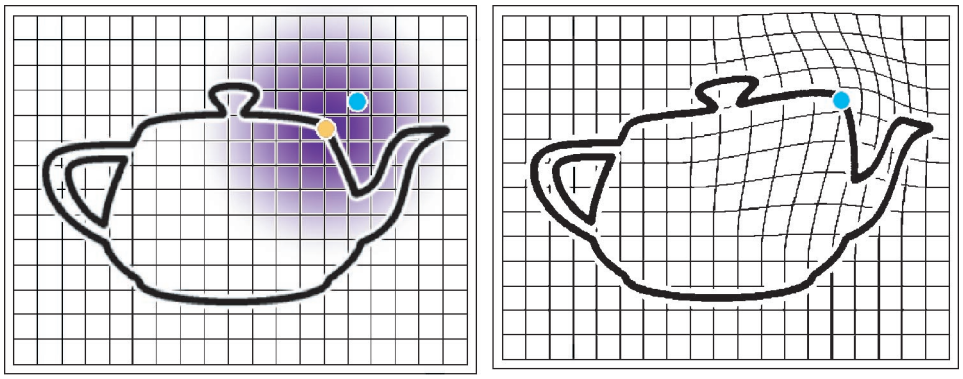

Fig. 3. A feature-point warp with a strength field. *Left*: Undeformed model, with strength field (purple). *Right*: Deformed model. The translation is scaled by the strength field.

function[5] taking on values in the range $[0, 1]$. A value of 1 for $s(u, M)$ indicates the translation is applied with full effect, while a value of 0 indicates the translation has no effect. We use the value of $s(u, M)$ to multiply the translation $\mathcal{T}$. It will be convenient to denote the multiplication of a translation $\mathcal{T}$ by a scalar $s$ by $s \cdot \mathcal{T}$. Thus, modifying Eq. (1) to include the strength field, we have:

$$\mathcal{D}(u, M) = s(u, M) \cdot \mathcal{T}. \tag{2}$$

Figure 3 shows a warp where $s(u, M)$ is 1 at the point $P$ and falls off with distance from $P$.

Now, suppose there are $n$ point-features with feature specifications $F_i = (P_i, P_i^\star)$, $1 \le i \le n$. Each feature specification is accompanied by a strength field $s_i(u, M)$ and can be described by a translation $\mathcal{T}_i = P_i^\star - P_i$. We would like to combine the effects of these translations at any given model point. To this end, we introduce a scalar-valued *weighting field* $w_i(u, M)$ for each feature specification $F_i$. All values $w_i(u, M)$ are positive, and higher values indicate that $\mathcal{T}_i$ has greater influence in the warp at $M(u)$. We compute the deforming function at each model-domain point $u \in U$ as a weighted average of

[5]In general, scalar values and scalar-valued functions will be denoted in lowercase, as indicated in Table I.

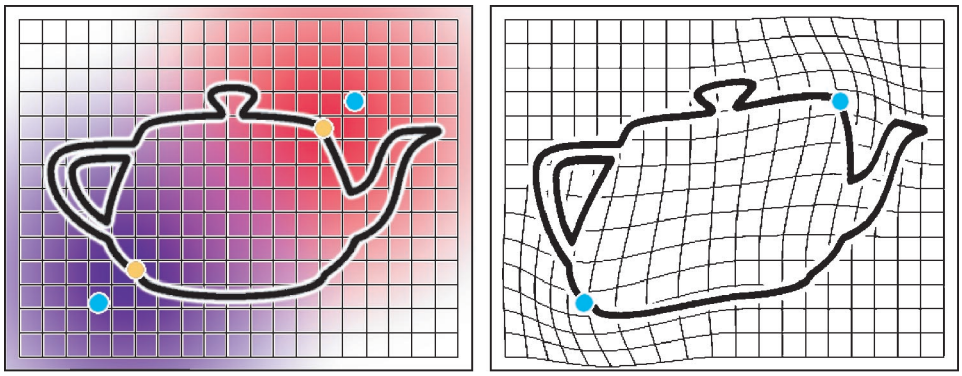

Fig. 4. A feature-point warp with weighting fields. *Left*: Undeformed model with two weighting fields (purple and red). *Right*: Deformed model, using a weighted combination of translations scaled by their strength fields.

translations scaled by their associated strength fields:

$$\mathcal{D}(u, M) = \sum_{i=1}^{n} \{\widehat{w}_i(u, M) \cdot (s_i(u, M) \cdot \mathcal{T}_i)\} \tag{3}$$

using normalized weights:[6]

$$\widehat{w}_i(u, M) = \frac{w_i(u, M)}{\sum_{j=1}^{n} w_j(u, M)}.$$

Figure 4 shows a warp where $w_i(u, M)$ is 1 at $P_i$ and falls off radially (at a different rate than $s_i(u, M)$).

Both strength fields and weighting fields define the "region of influence" of the feature specifications $F_i$ (and their associated translations $\mathcal{T}_i$) in different ways. The relationship between the two types of fields can be subtle. By providing separate fields, we decouple how much a given feature deforms the model (strength field) from its influence relative to other features (weighting field). Many warps in the literature combine these effects, or ignore one or the other of them entirely, but we have found the decoupling of these distinct concepts to be beneficial.

Notice that since weighting fields are normalized, they do not scale the overall effect of each translation as strength fields do. Rather, they alter the influence of translations relative to each other. For example, "Wires" (Section 4.4) uses weighting fields to establish a correspondence between each point on the model and a point on each source curve-feature, and strength fields to attenuate the deformation away from the curve.

In many cases, it is desirable for a uniform translation of all $\mathcal{T}_i$ to result in a rigid transformation of the model. To achieve this property, the strength fields can be set to 1 uniformly. We will see this in several of the warps discussed in this paper, such as Beier and Neely's [1992] image warp (Sections 4.2 and 5.2) and our new mesh warp (Section 6).

Notice that—evan at this early stage in our development of the framework—this formulation already has many desirable properties. It is geometrically based and independent of model representation, since it allows any model expressed as a point-valued function $M(u)$. In addition, Eq. (3) is amenable

[6]In practice, sometimes no feature has influence on the model at a particular point, in which case $w_j(u, M) = 0 \; \forall j \in \{1, \ldots, n\}$; in this case we use $\widehat{w}_i(u, M) = 0$.

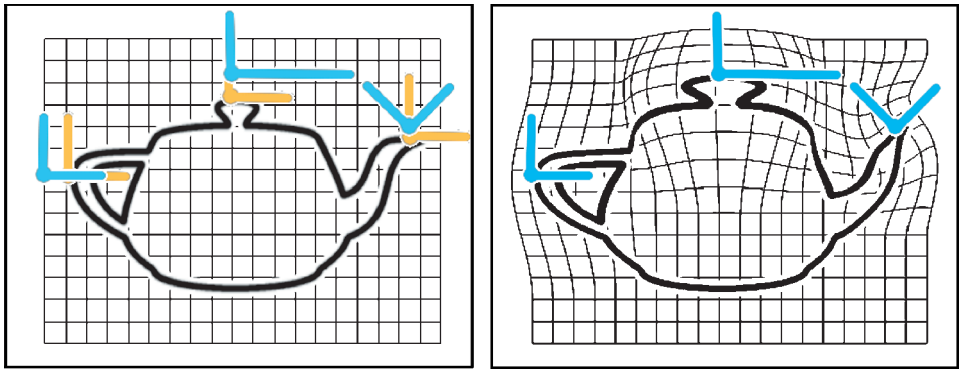

Fig. 5. A coordinate-frame based warp. *Left*: Undeformed model, with source (orange) and target (blue) coordinate frames. *Right*: Deformed model. Source coordinate frames map to targets (blue).

to mathematical analysis. However, it is built on very primitive elements (simple translations) and fails to concisely express many geometrically based warps. Despite its limitations, this formulation is powerful enough to express Bechmann's "space deformations" framework [Bechmann 1994], Borrel and Rapport's [1994] "constrained deformations," deformations based on radial basis functions (e.g., Arad et al. [1994]), and freeform deformations [Sederberg and Parry 1986] and their variants [Coquillart 1990; Greissmair and Purgathofer 1989; Hsu et al. 1992; Lee et al. 1995; MacCracken and Joy 1996]. In Section 4.1, we express freeform deformations in the framework.

### 3.3 More Complex Features

Continuing the development of the framework, we introduce the ability to build warps from primitives that are more complex than simple translations. The most obvious way to extend the framework in this way is to generalize translations to be general geometric transformations—compositions of translations, rotations, scales, and shears. Toward this end, consider a warp whose features are *oriented points*: that is, points with local coordinate frames (see Figure 5). If $f_i$ and $f_i^\star$ are the source and target coordinate frames, we have feature specifications:

$$F_i = (f_i, f_i^\star).$$

Each feature specification $F_i$ defines a transformation $\mathcal{T}_i$ that maps the source frame $f_i$ to the target frame $f_i^\star$. Usually (as in this case), a transformation is described directly by how it maps a source frame origin and basis vectors to a target. For some warps, however, the transformation is best described as an ordered composition of translate, rotate, scale, and shear components.

As in the last section, we have a strength field $s_i(u, M)$ that modulates the deformation of feature specification $F_i$ over the model. In the last section, we computed this modulation by scaling the translation $\mathcal{T}_i$ by $s_i(u, M)$. Here, we would like to scale the effect of a more general transformation. In general, one can define the multiplication $s \cdot \mathcal{T}$ of a transformation $\mathcal{T}$ by a scalar $s$ in many ways—we describe two:

*Displacement Method.* Linearly interpolate between the identity transformation $\mathcal{I}$ and the transformation $\mathcal{T}$. That is, $s \cdot \mathcal{T} = s\mathbf{T} + (1-s)\mathbf{I}$, where $\mathbf{I}$ and $\mathbf{T}$ are the matrix representations of the transformations $\mathcal{I}$ and $\mathcal{T}$, respectively. Using this method, the effect of applying $s \cdot \mathcal{T}$ to a point $P$ is to multiply the displacement caused by $\mathcal{T}$ by the scalar $s$, so that $(s \cdot \mathcal{T})\langle P\rangle = P + s(\mathcal{T}\langle P\rangle - P)$.

*Composition Method.* For a transformation defined as a composition of rotate, translate, scale, and shear components, multiply the parameters of each component of $\mathcal{T}$ by $s$ and then re-compose them to produce a new transformation $\mathcal{T}'$.

The warps we review here (and all the warps we have found in the literature) use one of these two scaling methods. When we discuss a warp, we will note the method it uses.

As in Section 3.2, we would like to combine the effects of multiple feature specifications at each model point by computing a weighted average of the transformations they imply. To compute this weighted average for our more general transformations, we must not only define the result of the multiplication $s \cdot \mathcal{T}$, but also the result of the addition of transformations. Just as there are many ways of defining the multiplication of a transformation by a scalar, there are many ways to define this addition. Fortunately, only one is necessary to cover all the warps we have found in the literature:

*Matrix Addition.* Add the matrix representations of the transformations. If the displacement method is used to compute the product of weights and transformations, the computed weighted average is equivalent to a weighted average of the displacements caused by the transformations.

In every warp we have found in the literature, the weighted average of transformations is computed as the weighted average of the displacements caused by the transformations. In terms of the concepts outlined above, the product of normalized weights $\widehat{w}_i(u, M)$ and transformations $(s_i(u, M) \cdot \mathcal{T}_i)$ is computed using the displacement method (regardless of the method used to compute the product $s_i(u, M) \cdot \mathcal{T}_i$), and matrix addition is used to compute the sum of transformations. We will assume this method for computing weighted averages of transformations in the remainder of this article.

Equation (3) still defines the deforming function for this new formulation, but now we interpret $\mathcal{T}_i$ as an arbitrary transformation, $\left(s_i(u, M) \cdot \mathcal{T}_i\right)$ as using one of our more general methods for multiplying a transformation by a scalar, and the weighted average of transformations as the weighted average of displacements caused by those transformations, as outlined above. This second formulation is the natural extension of the weighted average of scaled translations in Section 3.2, and allows us to describe a much broader range of warps, including van Overveld and Stalper's [1997] deformations with a polygonal skeleton mesh, vortex warps [Wolberg 1990] (illustrated near the spout of the teapot in Figure 5), Beier and Neely's [1992] image warp, and Lerios et al.'s [1995] volume warp. In Section 4.2, we express the latter two warps in the framework.

### 3.4 Continuous Features

In this section, we extend the formulation in Section 3.3 to include continuous features such as curves and surfaces. Without loss of generality, we assume that each such feature is parameterized on some continuous domain $V_i$. For concreteness, we consider an example warp that uses curve features (see Figure 6). Let $C_i(v)$ be a source curve and $C_i^\star(v)$ be the corresponding target curve, both parameterized on a domain $V_i = [0, 1]$. We then have feature specifications:

$$F_i = (C_i(v \in V_i), C_i^\star(v \in V_i)).$$

Now, the feature specification $F_i$ refers to continuous quantities parameterized on the domain $V_i$. To reflect this, we extend the transformation expressed by $F_i$ to be a *parameterized transformation* $\mathcal{T}_i(v)$, defined on the same domain $V_i$. In other words, $\mathcal{T}_i(v)$ is a function that returns a transformation when evaluated for any given $v \in V_i$. In Figure 6, the parameterized transformation for the curve-feature is given by the equation in Section 4.4.

The strength fields $s_i(u, M)$ and weighting fields $w_i(u, M)$ should now be parameterized on $V_i$, as well, yielding $s_i(v, u, M)$ and $w_i(v, u, M)$ for $v \in V_i$ and $u \in U$. The deforming function is computed as

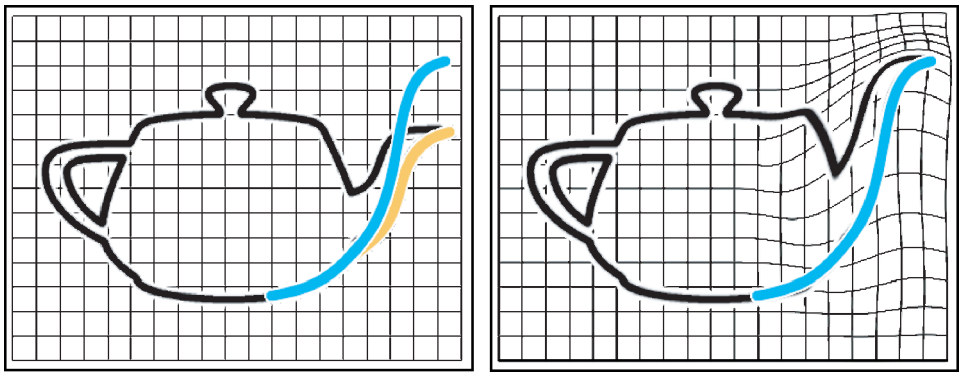

Fig. 6. A continuous curve-feature warp. *Left*: Undeformed model, with source (orange) and target (blue) curves. *Right*: Deformed model. Source curve is mapped to the target (blue).

before: as a weighted average of a (now infinite) set of scaled transformations. With a collection of $n$ transformation continuums (one transformation continuum $\mathcal{T}_i(v)\ \forall v \in V_i$ for each feature specification $F_i$, $1 \le i \le n$), the weighted average from Eq. (3) becomes a summation of integrals:

$$\mathcal{D}(u, M) = \sum_{i=1}^{n} \int_{v \in V_i} (\widehat{w}_i(v, u, M) \cdot (s_i(v, u, M) \cdot \mathcal{T}_i(v)))\, dv \tag{4}$$

with normalized weights $\widehat{w}_i(v, u, M)$ also given by integrating[7]:

$$\widehat{w}_i(v, u, M) = \frac{w_i(v, u, M)}{\sum_{j=1}^{n} \int_{x \in V_j} w_j(x, u, M)\, dx}.$$

In practice, $V_i$ may not be a continuous domain; it could be a discrete collection (for example, a collection of indices), in which case the integral becomes a summation and Eq. (4) becomes a double summation. In the limiting case, when $V_i$ is the null domain $\emptyset$ for all $1 \le i \le n$, Eq. (4) reduces to Eq. (3).

This final formulation of the framework provides the sophistication to compute very complicated warps, including Barr's [1984] deformation framework, and many recent warps in the literature such as Decaudin's [1996] deformations using convex shapes as features, Corrêa et al.'s [1998] depth-preserving curve-based warp, Lazarus et al.'s [1994] "axial deformations," and Singh and Fiume's [1998] "Wires." In Section 4.3, and 4.4, we cast Barr deformations and "Wires," respectively, in terms of the framework.

### 3.5 The Complete Framework

The preceding sections develop mathematical formulations for computing warps as weighted averages of collections of transformations constructed from feature pairs. In this section, we generalize these formulations to deformations that are not feature based, and present the complete framework, including a generic modular algorithm structure in which warps are constructed and evaluated.

Throughout the development of the framework thus far, we have used feature-based warps for concreteness and because the framework is designed to facilitate the design and analysis of feature-based warps in particular. But, feature-based warps—and the feature specifications used to define them—are simply one style of warp user control (see Section 2.2). As we noted in Section 1, a more general warp

[7]As before, if all weighting field values are zero, $\widehat{w}_i(v, u, M)$ is taken to be zero.

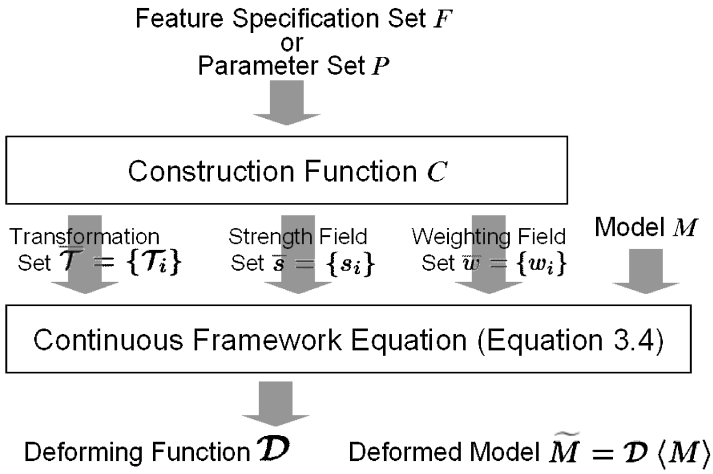


Fig. 7. An idealized schematic for warps in the framework. The warp's construction function generates continuous parameterized transformations, strength fields, and weighting fields that are used to evaluate the deforming function directly using Eq. (4).

structure simply takes "user-controlled values" as input: feature specifications are simply one type of such "values." More generally, a warp takes a parameter set $P$ as input to define the deforming function. Armed with this generalization, we are ready to state the complete framework.

Figure 7 shows schematically the components of the most straightforward warp structure within the framework. To design a particular warping algorithm, the warp designer creates specialized components to fill that structure. The warp designer chooses the model representation and the form of the parameter set $P$, and defines a customized *construction function* $C : P \to (\bar{\mathcal{T}}, \bar{s}, \bar{w})$ whose input is the set of parameters $P$ and whose outputs are the set[8] of parameterized transformations $\bar{\mathcal{T}} = \{\mathcal{T}_i(v)\}$, the set of corresponding strength fields $\bar{s} = \{s_i(v, u, M)\}$, and the set of corresponding weighting fields $\bar{w} = \{w_i(v, u, M)\}$. These can then be used to compute the warp using Eq. (4). Note that in the feature-based case, the warp designer chooses a feature specification representation, and defines a construction function $C : \bar{F} \to (\bar{\mathcal{T}}, \bar{s}, \bar{w})$, where $\bar{F} = \{F_i\}$ is the set of all feature specifications.

Although the integral in Eq. (4) could be numerically evaluated directly, in practice it is not. Instead, if any of the parameterized transformation domains $V_i$ are continuous (caused, for example, by continuous features in a feature-based warp), the warp designer also creates a sampling algorithm to discretize the continuous quantities in Eq. (4). This is illustrated in Figure 8. The discretizing algorithm is defined by a *sampling function* $S : (M, \bar{\mathcal{T}}, \bar{s}, \bar{w}) \to (\dot{\bar{\mathcal{T}}}, \dot{\bar{s}}, \dot{\bar{w}})$ whose inputs are the model, parameterized transformations, strength fields, and weighting fields and whose outputs are a set[9] $\dot{\bar{\mathcal{T}}} = \{\dot{\mathcal{T}}_j\}$, and new strength fields $\dot{\bar{s}} = \{\dot{s}_j(u, M)\}$ and weighting fields $\dot{\bar{w}} = \{\dot{w}_j(u, M)\}$ which are defined only on the domain $U$. These discrete quantities are then used in Eq. (3) to compute the warp. Programmatically, it is typically most convenient to merge the construction function and sampling function into a single construction function whose outputs are always discrete, as illustrated in Figure 9.

The formulation of a construction function has several advantages. First of all, in feature-based warps, it is often important for the warp designer to consider global deformation effects implied by all features, rather than simply considering local effects caused by individual features. The formulation of a construction function considers this need well. The construction function need not handle each feature

[8]Here, we denote the set comprised of a collection of indexed elements by an overbar, as in $\bar{s} = \{s_i(v, u, M)\}$.
[9]Discretized quantities are denoted with a dot, as in $\dot{s}_j(u, M)$.

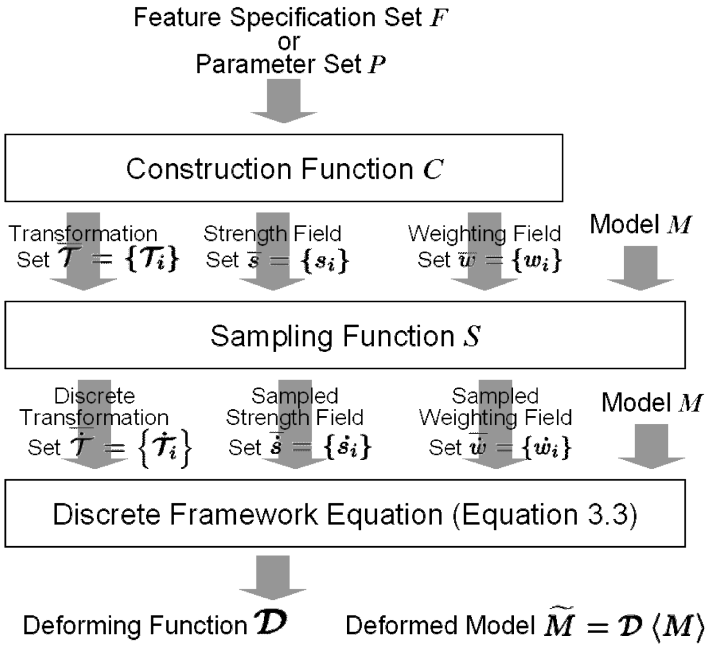


Fig. 8. A more realistic schematic for warps in the framework. The warp's construction function generates continuous parameterized transformations, strength fields, and weighting fields that are sampled by the warp's sampling function and evaluated using Eq. (3).

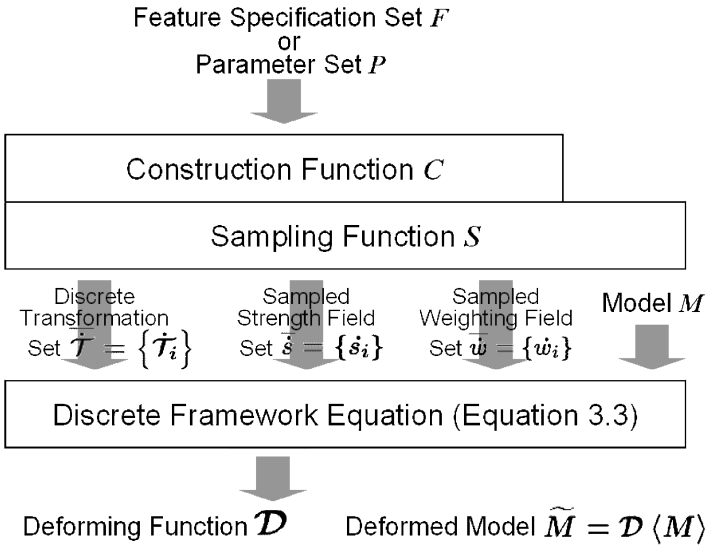


Fig. 9. A schematic view of how warps are actually expressed in the framework. The warp's construction function and sampling function are wrapped into a single function whose outputs are always discrete.

independently, but may take into account the interaction of features in producing the final transformations and fields. Since most features leave some elements of a transformation unconstrained (as we will see in Section 5.2), the warp designer can take advantage of his knowledge of all features to produce individual transformations that both match the local target features and consider the effects of nearby features. Second, by having an explicit construction step (embodied by the construction function) that separates the warp's parameters from the values it produces to compute the deforming function using Eq. (3), the user interface to the warp is decoupled from the warp's underlying implementation. This may allow the warp designer to create an intuitive interface while relying on less intuitive mathematics to guarantee certain properties. We will see this technique in the "direct manipulation" variant of freeform deformations [Hsu et al. 1992] (see Section 4.1) and Litwinowicz's [1994] energy-minimizing warp method.

### 3.6 Summary

Our framework consists of two key components. First, the framework relies on a concise geometrically meaningful equation to evaluate deformations, encapsulated in Eqs. (3) and (4). Second, the modular algorithm structure described in Section 3.5 and depicted in Figures 7, 8, and 9 provides a general method for constructing specific warps by defining specialized components to "plug in" to the generic structure. Together, these two elements form the complete framework, providing a conceptual and mathematical basis for describing warps.

Our framework is formally capable of trivially describing *any* warping algorithm, by encoding the deformation as a displacement look-up buried in the deforming function $\mathcal{D}(u, M)$. However, it expresses most warps in a form more amenable to analysis, evaluation, and conceptual simplicity. In particular, because it relies on geometric transformations and operations as its building blocks, the framework concisely and elegantly describes geometrically implemented warps and deformations.

For warps implemented using geometric concepts and operations, the framework offers many benefits. By focusing on the core implementation of deformation algorithms, the framework spans user control paradigms and model representations, allowing the warp designer to consider each of these components independently and to select the best user controls, model representation, and implementing construction function for the application at hand. Moreover, by providing a generic algorithm structure, the framework helps to organize the warp designer's thoughts as he creates new deformation algorithms. This generic structure also facilitates the comparison and evaluation of existing warping algorithms, and encourages reuse of components from various deformations. Finally, since the framework describes how warps are evaluated in a concise mathematical form, it is a valuable tool for mathematical analysis.

## 4. EXISTING WARPS EXPRESSED IN THE FRAMEWORK

In this section, we demonstrate the framework's ability to describe a variety of warps by considering four examples from the warping literature and showing how they can be expressed in terms of the framework. We present three feature-based warps and one parameter-based warp. To best illustrate the framework's flexibility, the selected deformations represent warps designed for a variety of applications and model representations: Sederberg and Parry's [1986] freeform deformation for solid modeling, Beier and Neely's [1992] line-segment based warp for image warping and metamorphosis, Barr's [1984] parameter-based deformations for solid modeling, and Singh and Fiume's [1998] curve-based "Wires" deformations for surface modeling and animation.

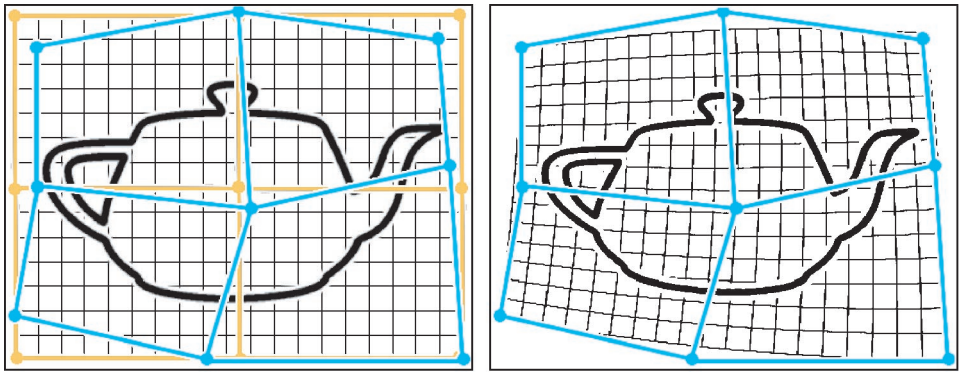

Fig. 10. A freeform deformation, with source and target feature-point lattices. *Left*: Undeformed model, source feature-point lattice (orange), and target feature-point lattice (blue). *Right*: Deformed model and target feature-point lattice (blue).

### 4.1 Freeform Deformation

Freeform deformation (FFD) [Sederberg and Parry 1986] is a feature-based warping algorithm designed for solid modeling. FFD is defined by uniformly spaced feature points in a parallelepiped lattice (see Figure 10). The user controls the deformation by moving points in the lattice: the original point positions constitute the source features, while the new point positions are the target features. The warp deforms objects embedded in the lattice in a way that approximates (but does not interpolate) the movement of the feature points from their source positions to their target positions (see Section 7.2 for further discussion of interpolation and approximation).

The model $M(u)$ in FFD is any point-valued function. That is, the parameters $u \in U$ and $M$ passed to the deforming function $\mathcal{D}(u, M)$ are used only to compute the point position $M(u)$, so any point-valued model can be used. Because of this, FFD is a *spatial warp*: it relies only on the spatial location of the undeformed points. The features are a 3-D grid of $(l+1) \times (m+1) \times (n+1)$ control points located in a local coordinate system imposed on a parallelepiped region. The local coordinate system has origin $O$ and three basis vectors $\vec{x}$, $\vec{y}$, and $\vec{z}$. The location of the *ijkth* control point $P_{ijk}$ in the undeformed lattice is given by $O + \frac{i}{l}\vec{x} + \frac{j}{m}\vec{y} + \frac{k}{n}\vec{z}$ for $0 \le i \le l, 0 \le j \le m, 0 \le k \le n$. The strength fields are always 1, and the weighting fields are generated using Bernstein polynomials [10] $B_i^d(t)$. Thus:[11]

$$\begin{aligned} F_{ijk} &= (P_{ijk}, P^\star_{ijk}) \\ \mathcal{T}_{ijk} &= P^\star_{ijk} - P_{ijk} \\ s_{ijk}(u, M) &= 1 \\ w_{ijk}(u, M) &= B_i^l(M_x(u)) B_j^m(M_y(u)) B_k^n(M_z(u)), \end{aligned}$$

where $M_x(u)$, $M_y(u)$, and $M_z(u)$ are the coordinates of $M(u)$ in the local coordinate system $(O, \vec{x}, \vec{y}, \vec{z})$. In this case, since the transformations $\mathcal{T}_i$ are translations, the composition method for multiplying transformations by scalars is identical to the displacement method.

Several authors have proposed variations on the original FFD algorithm. These can be expressed in the framework as slight modifications of the original formulation. For example, Greissmair and

[10] The Bernstein polynomial $B_i^d(t)$ is defined as $B_i^d(t) = \binom{d}{i} t^{d-i}(1-t)^i$.

[11] For convenience, we index the terms by trivariate *ijk* indices, even though the framework indexes them only by univariate indices. It is trivial to map the triple index *ijk* to a single index.

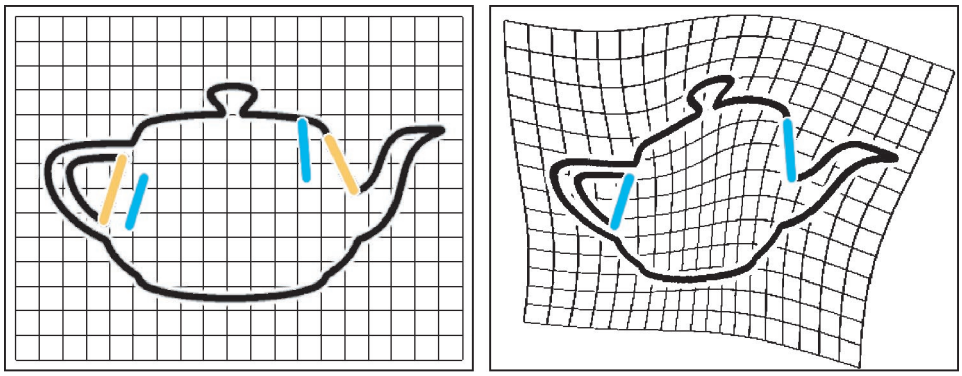

Fig. 11. A Beier & Neely image warp, with source and target line-segment features. *Left*: Undeformed model, source line-segment features (orange), and target line-segment features (blue). *Right*: Deformed model and target line-segment features (blue).

Purgathofer [1989] use trivariate b-spline polynomials for $w_{ijk}(u, M)$ instead of trivariate Bernstein polynomials: this yields a freeform deformation with local control. Various authors [Lee et al. 1995, 1996a, 1996b] have suggested layered application of successively higher-resolution lattices. Coquillart [1990] allows manipulation of the lattice prior to deformation, at the cost of increased complexity in computing the local coordinates $M_x(u)$, $M_y(u)$, and $M_z(u)$ used in $w_{ijk}(u, M)$. MacCracken and Joy [1996] allow lattices of arbitrary topology, further complicating the computation of $M_x(u)$, $M_y(u)$, and $M_z(u)$ in order to gain greater flexibility in the structure of the lattice. Finally, Hsu et al. [1992] present the user with a direct manipulation user interface ("place *this* point on the model *there*") and solve for an FFD target lattice satisfying the user-defined point constraints. Within the framework, this algorithm is most naturally understood as using feature specifications defined by direct manipulation constraints, and mapping those feature specifications to FFD control-point translations by way of a more complex construction function.

### 4.2 Beier and Neely's Image Warp

Beier and Neely [1992] describe a feature-based warp that operates in 2-D image space and whose features are directed line segments (see Figure 11). The warp is used as part of an image metamorphosis algorithm to align two images before a cross-dissolve between pixel colors completes the morph. The user controls the deformation by positioning source and target line segments and tuning parameters that define the weighting fields of the features.

This warp uses inverse mapping, meaning that the position of each pixel in the destination image is warped to a position in the source image to determine which source-image pixels should contribute to the color of the destination pixel (see Section 8.3 for further discussion of inverse mapping). As in FFD, the warp is defined spatially: any 2-D point-valued model can be deformed. In practice, the model is sampled at the $(x, y)$ pixel coordinates of the destination image. Each feature specification $F_i$ is described by a pair of line segments $L_i$ and $L_i^\star$, having endpoints $(P_i, Q_i)$ (in the *destination* image, because of inverse mapping) and $(P_i^\star, Q_i^\star)$ (in the *source* image), along with scalar parameters $a_i$, $b_i$, and $p_i$ whose use is explained later. The transformation corresponding to each feature specification is selected to take the source line segment $L_i$ to the target line segment $L_i^\star$:

$$F_i = (L_i, L_i^\star, a_i, b_i, p_i)$$
$$\mathcal{T}_i = map\ from\ f_i \rightarrow f_i^\star,$$

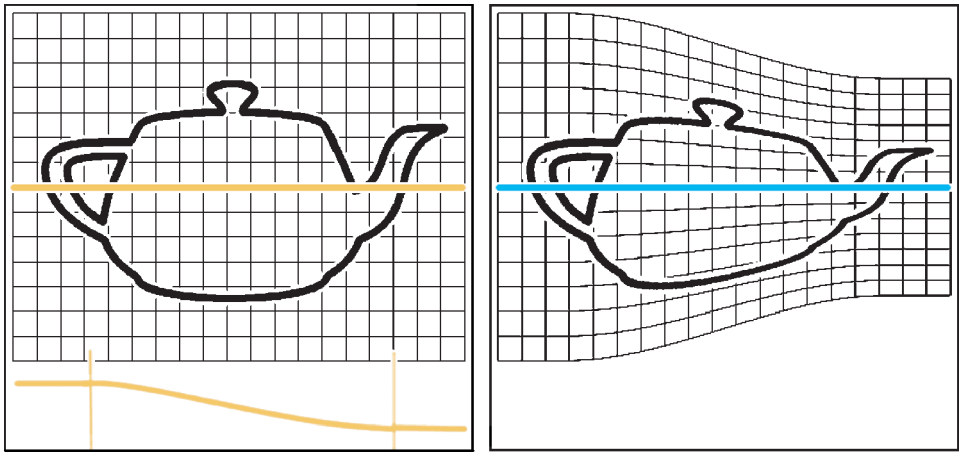

Fig. 12. A Barr taper deformation, with deformation axis and scalar-valued taper function. *Left*: Undeformed model, deformation axis (orange, shown overlaid on the model), and user-defined taper function (orange, shown below the model). *Right*: Deformed model and deformation axis (blue, shown overlaid on the model).

where $f_i$ is a 2-D coordinate frame with origin $P_i$, x-basis equal to $Q_i - P_i$, and y-basis equal to the normalized vector perpendicular to $Q_i - P_i$, and similarly for $f_i^\star$. The effect of a transformation is scaled using the displacement method.

To ensure that a rotation or scale of a single line segment produces the corresponding rotation or scale of the image, the strength fields are always 1 (see Section 3.2). The weighting fields fall off with distance, and give longer line segments greater influence:

$$s_i(u, M) = 1$$
$$w_i(u, M) = \left( \frac{length_i^{p_i}}{a_i + dist_i} \right)^{b_i},$$

where $length_i = \|Q_i - P_i\|$ is the length of line $L_i$ and $dist_i$ is the minimum distance from the pixel coordinate $M(u)$ to the line $L_i$.

Lerios et al. [1995] extend Beier and Neely's [1992] 2-D technique to the 3-D volume-warping domain. In this extension, the user manipulates point, line-segment, rectangle, and box features to control the warp. In the framework, these become feature specifications that constrain zero, one, two, or three basis vectors of the source and target coordinate frames in 3-D. As in the original Beier and Neely warp, the unconstrained basis vectors are chosen to be perpendicular to the constrained basis vectors and to each other. This leaves the model undeformed in the direction of these automatically chosen vectors. While this choice is reasonable, it can lead to undesirable effects in certain cases, as we will see in Section 5.2.

## 4.3 Barr Deformations

Barr [1984] describes a family of parameter-based warps whose parameters are geometric quantities allowing intuitive user control. All of his deformations fit neatly within the context of the framework: this section focuses on one—taper deformations—and also examines a warping framework comprised of generalized Barr deformations. Barr deformations do not depend on a particular model representation. Like FFD and Beier and Neely's warp, they are spatially defined.

4.3.1 *Taper Deformation.* The taper deformation imparts a global taper to an object along a particular axis (see Figure 12). Essentially, a taper is a scaling transformation whose effect is varied smoothly

along a chosen axis. To be faithful to Barr's original publication [Barr 1984], we will only express tapers along the unit z-axis $\vec{z} = (0, 0, 1)$. This formulation can be easily extended to other axes.

The parameter set for a taper deformation contains a single element: a user-defined function from scalar values to scalar values. The user controls the deformation by manipulating this function. The function is interpreted geometrically: its input is a value along $\vec{z}$ and its output is the amount by which the deforming function will scale the model in directions orthogonal to $\vec{z}$. This parameter set maps to a single parameterized transformation whose input is a scalar value, and whose output is the scaling transformation:

$$P = \{f(t)\}$$
$$\mathcal{T}_1(v) = \textit{scale by } (f(v),\ f(v),\ 1),$$

where $f(t) : t \in V_1 \subset \mathbb{R} \rightarrow \mathbb{R}$ is the forementioned user-defined function, and $v \in V_1 \subset \mathbb{R}$. Either the displacement method or the composition method can be used to scale the effect of transformations.[12]

The single strength field in a taper deformation is set to 1 uniformly, creating a deformation that scales the model exactly as specified by the function $f(t)$. The weighting field is selected so that the only transformation that affects a model point $M(u)$ is the transformation corresponding to $M(u)$'s projection onto $\vec{z}$. Formally:

$$s_1(v, u, M) = 1$$
$$w_1(v, u, M) = \begin{cases} 1 & \text{if } M(u) \cdot \vec{z} = v \\ 0 & \text{otherwise.} \end{cases}$$

This expression of Barr's taper deformation mirrors Barr's original formulation, where a separate transformation is constructed for each point to be warped, based on its projection onto the deformation axis. In fact, the formulation in the framework, with its weighting field singularities, suggests per-point sampling of the parameterized transformations,[13] which Barr makes explicit.

4.3.2 *Generalized Barr Deformations.* A straightforward generalization of Barr's original deformations forms a simple framework for warps and deformations. In this framework, deformations are defined by geometric parameters—such as the user-defined taper function in Section 4.3.1—that define a single parameterized transformation. Generally, these geometric parameters, and the parameterized transformations they define, vary in a geometrically meaningful way, such as along an axis. For each model point, a single transformation is selected from the parameterized transformation's continuous spectrum of transformations, and this transformation is applied to the model point. In the taper deformation, for example, the transformation is selected by the axis location onto which the model point projects.

### 4.4 "Wires"

Singh and Fiume [1998] present a feature-based deformation called "Wires" that uses 3-D curve features to manipulate smooth surfaces for modeling and animation (see Figure 6). The user controls the deformation by first "binding" a curve to a surface and then editing the curve. The "bound" curve shape is used as the source feature while the edited curve shape is used as the target feature. "Wires" provides a large set of options to control the deformation. While the framework expresses all of these, for brevity this section will cover only the basic functionality.

[12] In this case, the method used makes no difference, since (as we will see) the strength field is always 1.

[13] That is, each parameterized transformation is sampled once for each model point $M(u)$, creating a distinct transformation for each model point.

The model is a set of control points defining a smooth surface, such as the vertices of a NURBS mesh. Thus, $U$ is the set of indices $\{1, 2, \ldots, l\}$, where $l$ is the number of control points, and $M(u)$ is the control point with index $u$.

The features for the warp are source and target curves and a few related scalar parameters. Each wire feature is represented by a feature specification:

$$F_i = (C_i(v \in V_i), C_i^\star(v \in V_i), f_i(t \in \mathbb{R}), r_i, x_i, m_i),$$

where $C_i(v)$ is the source or "reference" curve, $C_i^\star(v)$ is the target or "wire" curve, $f_i(t)$ is a fall-off function,[14] and $r_i$, $x_i$ and $m_i$ are user-defined parameters whose use is described later. Without loss of generality, the wire and reference curves are assumed to be parameterized on the domain $V_i = [0, 1]$.

Each feature specification $F_i$ maps to a parameterized transformation $\mathcal{T}_i(v), v \in V_i = [0, 1]$ such that $\mathcal{T}_i(v)$ takes the point $C_i(v)$ to $C_i^\star(v)$. $\mathcal{T}_i(v)$ is given by a composition:

$$\mathcal{T}_i(v) = \mathcal{X} \circ \mathcal{R} \circ \mathcal{S},$$

where $\mathcal{X}$ is the translation $C_i^\star(v) - C_i(v)$, $\mathcal{R}$ is the smallest rotation that takes the vector $dC_i(v)/dv$ to $dC_i^\star(v)/dv$, and $\mathcal{S}$ is a uniform scale of magnitude $x_i$ centered at the point $C_i(v)$. Transformations are scaled using the composition method.

The most basic strength field in "Wires" for a given feature specification $F_i$ decreases from 1 to 0 as the distance from $M(u)$ to $C_i(v)$ varies from 0 to $r_i$. The rate of decrease is defined by the fall-off function $f_i$:

$$s_i(v, u, M) = f_i\left(\frac{\|M(u) - C_i(v)\|}{r_i}\right)$$

The weighting field for each feature specification is selected to warp each model point $M(u)$ based solely on the transformation value at the nearest location on the reference curve.[15] In the most elementary case, the nonzero value of the weighting field depends on how much the parameterized transformation $\mathcal{T}_i(v)$ displaces the point $M(u)$:

$$w_i(v, u, M) = \begin{cases} 0 & \text{if } \exists\, c < v \text{ such that } \|M(u) - C_i(c)\| < \|M(u) - C_i(v)\| \\ \|(s_i(v, u, M) \cdot \mathcal{T}_i(v))\langle M(u)\rangle - M(u)\|^{m_i} & \text{otherwise,} \end{cases}$$

where $m_i$ determines the relative effect of wires that cause large versus small displacement.

Lazarus et al. [1994] also describe a class of feature-based deformations, termed "Axial Deformations," that use curves as warping features. The algorithm is similar to "Wires," but uses the displacement method for computing scaled transformations, and provides less exotic control of strength fields and weighting fields. Corrêa et al. [1998] describe another curve-based warp for use in cel animation. The framework expresses both of these warps in a straightforward manner.

## 5. VARIATIONS OF EXISTING WARPS DEVELOPED IN THE FRAMEWORK

By expressing warps within a modular algorithm structure, the framework isolates warp components. This modular view helps to clarify the strengths and weaknesses of various approaches, making it easy to see ways of improving components in particular deformation algorithms. Moreover, the framework reveals the mathematical structure of warps in the concise mathematical equation used for evaluating

[14] $f_i(t) : \mathbb{R}^+ \to [0, 1]$, is monotonically decreasing and at least $C^1$-continuous, with $f_i(0) = 1$, $f_i(t) = 0\ \forall t \geq 1$, and $f_i'(0) = f_i'(1) = 0$.

[15] If two or more points on the curve $C_i$ are equidistant from the model point $M(u)$, the point with the smallest parameter value is chosen.

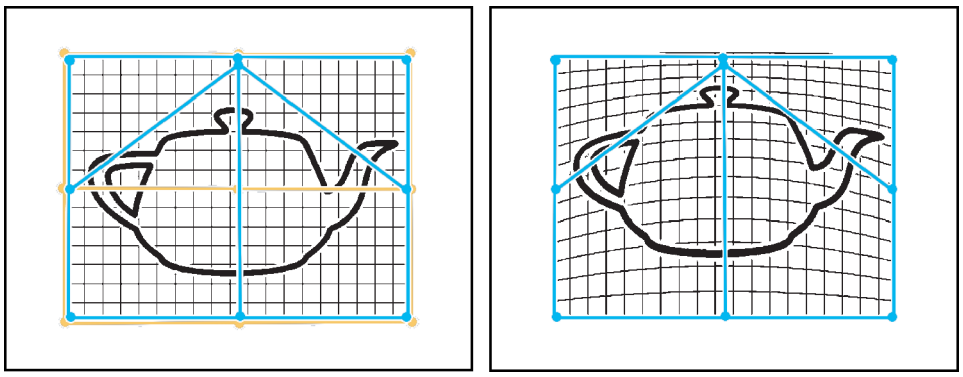

Fig. 13. A freeform deformation unable to create a local rotational effect. *Left*: Moving the center point of the lattice. *Right*: The deformed model; there is no way to create a local rotational effect.

them. By examining that mathematical structure, we can gain insight into how to refine and fine-tune individual warps. In this section, we demonstrate these advantages by developing new variants of three warps from Section 4.

### 5.1 Variant of Freeform Deformation

As described in Section 4.1, freeform deformation (FFD) is a feature-based warp defined by point features in a uniform 3-D lattice imposed on a parallelepiped volume. While the structure of this interface is important for the mathematical guarantees of FFDs, it is highly restrictive, and makes certain editing operations difficult or impossible.

For instance, given a fixed-density control lattice, it is not possible to create a deformation with a rotational effect centered around a single control point (see Figure 13). Some FFD variants provide limited ways of dealing with this restriction. In particular, the user of multi-level FFDs [Lee et al. 1995, 1996a, 1996b] can approximate localized rotational deformations by editing finer resolutions of the control lattice. However, this technique substantially increases the user intervention and computational cost of the warp, and does not yield true rotational effects.

By casting FFDs in terms of the framework, a more elegant approach becomes obvious. In the context of the framework, it is clear that the features of FFDs are particularly simple: source and target points that map to simple translations. One obvious approach for obtaining more complicated effects is to introduce more complex features. Instead of simple points, oriented points like those discussed in Section 3.3 may be used. In this case, the user manipulates the deformation by adjusting a lattice of local coordinate frames, each of which has an origin and basis vectors. Now, feature specifications contain source and target coordinate frames instead of source and target points. Feature specifications map to general transformations that take source frames to target frames. With this new warp, it is possible to create a rotational effect centered at a particular control frame by rotating the frame's basis vectors (see Figure 14).

One of the advantages of the original FFD interface (uniformly spaced feature points in a parallelepiped volume), is that it facilitates the derivation of mathematical continuity guarantees. Fortunately, we can prove that our new warp has the same continuity guarantees by analyzing its mathematical structure in the context of the framework (see Section 7.3).

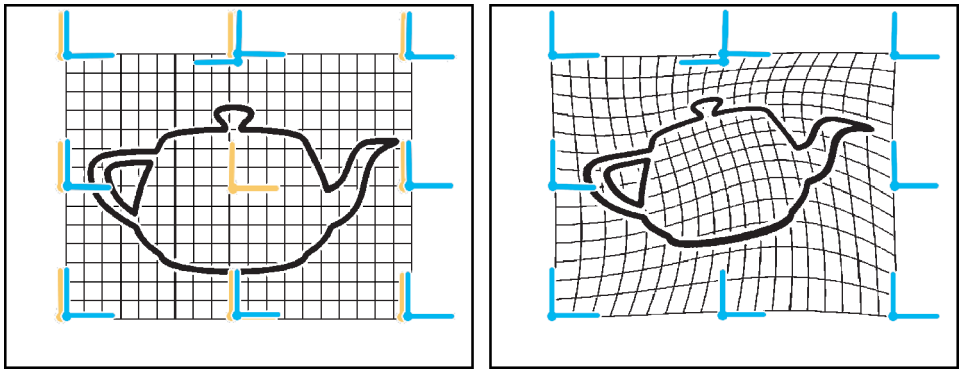

Fig. 14. Our freeform deformation variant using coordinate frames as features. *Left*: Rotating the center frame of the lattice. *Right*: The deformed model, with local rotational effect.

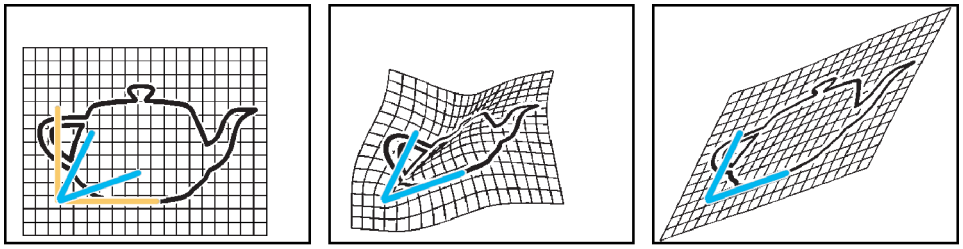

Fig. 15. A Beier and Neely warp variant that considers global effects. *Left*: Feature line segments. *Center*: Buckling artifacts caused by competing features in standard Beier and Neely warping. *Right*: The new warp considers global effects, yielding a shear.

### 5.2 Variant of Beier and Neely's Image Warp

Recall from Section 4.2 that Beier and Neely's image warp is defined by line-segment features. When two line-segments "squeeze" the image, the resulting image can exhibit spatial buckling artifacts, where the 2-D plane folds back on itself (see Figure 15).[16] Just as a rotation of two line segments results in a rotation of the image, we would like the warped image resulting from a shear of two line segments to better approximate the shear.[17]

In the context of the framework, this buckling behavior is easily understood as a failure to consider global deformation effects in the Beier and Neely construction function. Specifically, Beier and Neely make the assumption that the transformation for a line segment feature should not deform the image in the direction perpendicular to the line segment. For a single line segment, this assumption is reasonable, and results in the expected rigid-body transformation. For multiple line segments, however, this assumption can lead to nearby transformations that conflict dramatically enough in their "opinions" of the shape of the deformation to cause the buckling artifacts we observe. To avoid these artifacts, global deformation effects must be taken into consideration.

[16]For illustration, Figure 15 uses forward mapping to compute the warp. The artifacts we address occur with either forward mapping or inverse mapping, but are more noticeable in the forward mapping case.

[17]Beier and Neely [1992] observe that their technique may suffer from "ghosting" artifacts. The variant we describe here addresses only buckling, not ghosting.

When solving a similar problem in the domain of 3-D curve-based warping, Corrêa et al. [1998] develop a technique in which they recognize that, when creating coordinate frames on source and target features, the frame basis vector lying along the feature should transform in accordance with the feature specification (a line segment in Beier and Neely warping), but the other basis vector need not be constrained. Instead of applying a null transformation to the unconstrained basis vector, they compute a target vector based on the effect of other transformations in the warp. We use a similar approach here to compute target coordinate frames.

In the new warp, we map feature specifications to transformations as follows. The construction function input is the feature set $\bar{F} = \{F_i\}$ where $F_i = (L_i, L_i^\star, a_i, b_i, p_i)$, as in Section 4.2. Note that $L_i$ and $L_i^\star$ are directed line segments defined by their endpoints $(P_i, Q_i)$ and $(P_i^\star, Q_i^\star)$, respectively. We compute the set of transformations $\bar{\mathcal{T}} = \{\mathcal{T}_i\}$ by setting the transformation $\mathcal{T}_i$ for $F_i$ to be the transformation that maps the 2-D source coordinate frame $f_i$ to the 2-D target coordinate frame $f_i^\star$, where $f_i$ has origin $P_i$, x-basis vector $Q_i - P_i$, and y-basis equal to the normalized vector $N_i$ perpendicular to $L_i$, and $f_i^\star$ has origin $P_i^\star$, x-basis $Q_i^\star - P_i^\star$, and y-basis given by:

$$\sum_j \{(\widehat{w_{ij}} \cdot \mathcal{R}_j)\langle N_i \rangle)\}$$

where

$$w_{ij} = \left( \frac{length_j^{p_j}}{(a_j + dist_{ij})(a_j + (Q_i - P_i) \cdot (Q_j - P_j))} \right)^{b_j}$$

where $length_j$ is the length of line-segment $L_j$, $dist_{ij}$ is the minimum distance between the line-segments $L_i$ and $L_j$, $\mathcal{R}_j$ is the rotation that takes $L_j$ to $L_j^\star$, and $\widehat{w_{ij}}$ are normalized weights, $\widehat{w_{ij}} = w_{ij} / \sum_{j=1}^{n} w_{ij}$. This corrects the transformation by setting the unconstrained target vector to be a weighted average of the target vectors implied by all line-segments. The first term in the denominator of $w_{ij}$ gives preference to nearby line-segments, while the second term gives preference to line segments that are initially perpendicular to $L_i$, since their directions are closest to $N_i$.

Figure 15 shows the effect of the new warping algorithm. The local buckling artifact is removed, resulting in a smoother warp that better considers the global shearing effect of the deformation. The new warp is more expensive than the original Beier and Neely warp, requiring additional computation for the transformations. This cost is $O(n^2)$, where $n$ is the number of feature specifications. However, since the number of pixels in the image is much greater than the number of line segment features, the extra cost is negligible in comparison to warp computation on the image itself.

### 5.3 Variant of "Wires"

In certain cases, "Wires" deformations, introduced in Section 4.4, cause buckling and tearing artifacts. This occurs when a given wire's reference curve is shaped so that it is equidistant from a set of points in the model, and different parts of the wire curve specify competing deformations on this "medial set" (see Figure 16).

In the context of the framework, this artifact is easily understood. Recall that the weighting fields for wires are discontinuous, containing singularities to ensure that only the nearest point on a curve contributes to the deformation of each model point. Naturally, the resulting warp, which is a function of these fields, will also contain discontinuities (see Section 7.3). If we remove discontinuities in the weighting fields, we will succeed in removing discontinuities in the warp itself. Removing these discontinuities is straightforward, but to ensure that the deformation of a model point is still dominated by the section of the source curve nearest to it, we must also incoporate a distance-based fall-off similar

Fig. 16. A variant of "Wires" that removes discontinuities in the warp. *Left*: Curve features for the warp (closeup). *Center*: Buckling artifacts caused by discontinuity in the warp. *Right*: The new warp, with discontinuities removed.

to that used for the strength fields:

$$w_i(v, u, M) = f_i\left(\frac{\|M(u) - C_i(v)\|}{r_i}\right) \|(s_i(v, u, M) \cdot \mathcal{T}_i(v))\langle M(u)\rangle - M(u)\|^{m_i},$$

$m_i$ is a parameter described in Section 4.4. Figure 16 shows the result of our new algorithm. The warp no longer contains a spatial discontinuity.

This simple change alters the "Wires" algorithm significantly. Most notably, for each model point, the algorithm now approximates an integral taken along the curve, rather than considering only a single sample on the curve. This incurs an added computational cost: every transformation sample along each wire curve now contributes to the deformation of each model point. Thus, rather than sampling each curve exactly once for every model point as in the original algorithm, we need to integrate along the curve—or, in practice, approximate the integral by sampling. In Figure 16, we sample transformations uniformly along the curve. Finally, the new warp no longer exactly maps source curves to target curves in a strict mathematical sense. However, in practice it still *approximately interpolates* the feature curves, which is ample for the animation applications toward which "Wires" is geared (for further discussion, see Section 7.2).

## 6. A NEW WARP DESIGNED IN THE FRAMEWORK

In this section, we illustrate the design strengths of the framework by presenting a novel warping algorithm: a *mesh warp* that uses vertices of a low-resolution mesh as features in an interpolating feature-based warp. Users of this warp manipulate a coarse mesh to deform a detailed surface model. The warp is designed with two target applications in mind:

*Simulation Correction.* In cloth dynamics simulation, unwanted object intersections sometimes occur due to limitations and simplifications of the simulation technique. Often, instead of revisiting the simulation, it is more convenient to perform a slight ad hoc deformation on the simulated cloth to eliminate these intersections. To do so, we create a low-resolution mesh from a few vertices of the cloth, and manipulate the vertices of this simple mesh to adjust the complex cloth mesh (see Figure 17).

*Model Variation.* It is often useful to create variations of a complex surface, while maintaining its basic shape. While simple transformation mechanisms are useful for this sort of control, they often do not provide the required level of flexibility. In these cases, we can use an approximate low-resolution version of the high-resolution surface to deform the surface while maintaining its basic shape (see Figure 18).

A number of technologies provide tools that might be useful for these applications, but none of them is ideal. Multi-resolution surface editing schemes [Forsey and Bartels 1988; Kobbelt et al. 1998; Zorin

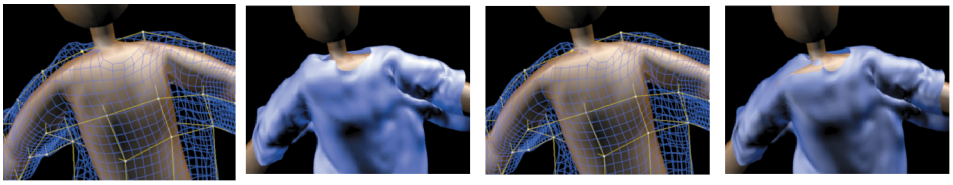

Fig. 17. Using the mesh warp for cloth simulation correction. *Left to right*: a cloth simulation gone awry—the shirt penetrates the right collarbone; low-resolution mesh imposed on the simulated mesh; low-resolution mesh adjusted to remove the penetration; the adjusted model.

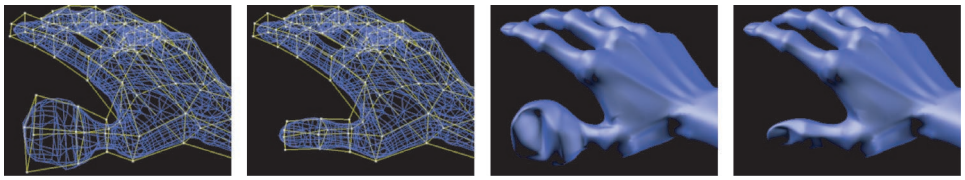

Fig. 18. Using the mesh warp to vary the shape of a hand model. *Left to right*: the original surface; a low-resolution mesh imposed on the surface; editing vertices in the deformation mesh; the deformed model.

et al. 1997] provide similar control, but generally restrict the relationship between the coarse mesh and the fine surface mesh, in terms of topology or mesh connectivity. The mesh warp requires only proximity—the coarse mesh should be near the surface. Coquillart [1990] and MacCracken and Joy [1996] describe freeform deformations which relax the usual restrictions on the FFD control lattice, creating deformations that use mesh vertices as features. Both of these warps, however, approximate the movement of edited lattice points. In contrast, to provide intuitive direct manipulation for our target applications, we want our mesh warp to interpolate the movement of edited points (see Section 7.2). van Overveld and Stalpers [1997] describe deformations using a polygonal skeleton mesh within an object. Our mesh warp provides greater flexibility, because the low-resolution mesh in our warp can be located outside the surface and constructed simply by sampling the object surface.

To develop the mesh warp, we first take $M(u)$ to be the set of control vertices defining the fine surface model, indexed by $u \in U = \{1, \ldots, l\}$. The features of the warp are the vertices of a coarse *deformation mesh,* which is assumed to be in close proximity to the surface. Each feature specification encodes a vertex point's position and its incident edges. That is, for vertex $i$ in the mesh, we have a feature specification:

$$F_i = (P_i, \overline{E_i}, P_i^\star, \overline{E_i^\star}),$$

where $P_i$ and $P_i^\star$ are the source and target positions of vertex $i$, and $\overline{E_i}$ and $\overline{E_i^\star}$ are the sets of edges incident on vertex $i$ in the source and target meshes, respectively. Each edge $\vec{E}_{ij} \in \overline{E_i}$ is represented as the vector from $P_i$ to its adjacent vertex, and likewise for $\overline{E_i^\star}$. Note that $\overline{E_i}$ has the same number of elements $k$ as $\overline{E_i^\star}$, in the same order—the target mesh must have the same topology as the source.

We map each feature specification $F_i$ to a single transformation composition:

$$\mathcal{T}_i = \mathcal{A} \circ \mathcal{X},$$

where $\mathcal{X} = P_i^\star - P_i$ is the translation mapping the source vertex position to the target position, and $\mathcal{A}$ maps the source edge set edge set $\overline{E_i}$ to the target edge set $\overline{E_i^\star}$. In general, there are morethan three

edges incident on vertex $i$, so it is impossible to compute $\mathcal{A}$ such that it will map $\overline{E_i}$ to $\overline{E_i^\star}$ exactly. We choose $\mathcal{A}$ to be the $3 \times 3$ matrix that best maps $\overline{E_i}$ to $\overline{E_i^\star}$ in the least-squares sense. That is, if $\mathbf{E}$ is a $k \times 3$ matrix having the vectors of $\overline{E_i}$ as its rows, and $\mathbf{E}^\star$ a $k \times 3$ matrix having the vectors of $\overline{E_i^\star}$ as its rows, we compute:

$$\mathbf{A} = \mathbf{E}^+\mathbf{E}^\star,$$

where $\mathbf{E}^+$ is the pseudo-inverse of $\mathbf{E}$ and $\mathbf{A}$ is the matrix representation of the transformation $\mathcal{A}$. We use the displacement method to scale the effect of transformations.

It remains to choose the strength fields and weighting fields. To ensure that a rigid-body transformation applied to the entire coarse deformation mesh will result in the same transformation of the surface, we set the strength field to be 1:

$$s_i(u, M) = 1.$$

Choosing the weighting fields is more complex. We would like $w_i(u, M)$ to have two properties. First of all, the value of $w_i(u, M)$ should be relatively large near the point $P_i$. Second, we would like $w_i(u, M)$ to be 0 at the end of each edge $\vec{E}_{ij}$; otherwise, the transformation $\mathcal{T}_i$ will compete with another transformation $\mathcal{T}_j$ corresponding to the other vertex incident on $\vec{E}_{ij}$. To achieve these properties, we set $w_i(u, M)$ to:

$$w_i(u, M) = f_{\alpha\beta}\left(\frac{\|M(u) - P_i\|}{\|\vec{E}_{im}\|}\right)\left(\prod_j f_{\gamma\mu}\left(\frac{(M(u) - P_i) \cdot \vec{E}_{ij}}{\vec{E}_{ij} \cdot \vec{E}_{ij}}\right)\right),$$

where $f_{ab}(t)$ is a function that is 1 for $t \leq a$, 0 for $t \geq b$, and is an interpolating cubic in between. $\vec{E}_{im}$ denotes the vector in $\vec{E}_i$ of maximum length, and $\alpha$, $\beta$, $\gamma$, and $\mu$ are scalar parameters that control the size of the maximum-value and non-zero regions of $w_i(u, M)$, respectively. The first term here ensures that the weighting field is large near $P_i$ and falls off as the distance from $M(u)$ to $P_i$ increases, while the second product term ensures that the weighting field is large at the beginning of each edge $E_{ij}$ and falls to zero by the end of the edge. The combination of these two terms fulfills the requirements for the weighting field outlined above.

Our mesh warp has several important properties. First of all, deformation meshes of arbitrary topology and genus may be used. Secondly, unlike many previous methods that use meshes to control a deformation [Coquillart 1990; MacCracken and Joy 1996; Sederberg and Parry 1986], the warp is interpolating—a point located at a vertex position in the original mesh is deformed to the vertex position in the target mesh. Third, the warp is defined spatially, so that the only relationship required between the low-resolution deformation mesh and the fine surface mesh is that the deformation mesh be located near the surface mesh. This makes constructing a deformation mesh particularly easy: the vertices of the deformation mesh can be any subset of the vertices of the original mesh, or some entirely new enclosing mesh. Finally, since the weighting field for a mesh vertex falls to zero, moving a single point in the deformation mesh has localized effect in the warp.

## 7. MATHEMATICAL PROPERTIES OF WARPS IN THE FRAMEWORK

One benefit of our framework is its amenability to mathematical analysis. Specifically, the framework—as expressed in Eqs. (3) and (4)—helps us prove properties about a warp's deforming function and the deformed model it produces based on simpler properties of the more easily analyzed components of the warp: parameterized transformations, strength fields, and weighting fields. In this section, we examine four deformation properties and develop conditions relating these warp properties to similar properties

of more easily analyzed warp components. For brevity, we cover only the most interesting results, and omit formal proofs; for a more formal and thorough description, see Milliron [1999].

### 7.1 Commutativity

In many warping applications, it is desirable for warps to commute: that is, for the order in which warps are performed to be irrelevant to the final result. In interactive modeling and animation applications, this guarantees that the order of operations does not matter, and provides for more predictable behavior.

Two warps $A$ and $B$ commute if evaluating and applying $A$ then $B$ to a model yields the same deformation as evaluating and applying $B$ then $A$. In the context of our framework, warps $A$ and $B$ with deforming functions $\mathcal{D}_A$ and $\mathcal{D}_B$, respectively, commute if and only if $\mathcal{D}_B(u, \mathcal{D}_A\langle M\rangle) = \mathcal{D}_A(u, \mathcal{D}_B\langle M\rangle)$ for all $u \in U$.

In the framework, we can guarantee this property if we guarantee two subconditions: first that neither $\mathcal{D}_A$ or $\mathcal{D}_B$ is computed differently if $\mathcal{D}_B$ or $\mathcal{D}_A$, respectively, is previously applied to the model $M$; second, that the order of application of $\mathcal{D}_A$ and $\mathcal{D}_B$ doesn't affect the final result. Together, these conditions provide a guarantee that the composition of the warps will not depend on the order of application, because (by the first rule) each evaluates to the same transformation even if the other is applied first, and (by the second rule) both application orders of the evaluated transformations yield the same final result. We can meet the first condition by ensuring that the parameterized transformation, strength field, and weighting field values of each warp do not change if the other warp is applied first. We can meet the second condition by ensuring that all weighted averages of warp $A$'s transformations commute with all weighted averages of warp $B$'s transformations. Finally, these restrictions may be removed when the weighting fields evaluate to zero. This leads to the following conditions for warps $A$ and $B$ to commute:

*Warp Commutativity.* Two warps $A$ and $B$ with deforming functions $\mathcal{D}_A$ and $\mathcal{D}_B$, strength fields $s_{A_i}$ and $s_{B_j}$, weighting fields $w_{A_i}$ and $w_{B_j}$, and parameterized transformations $\mathcal{T}_{A_i}$ and $\mathcal{T}_{B_j}$, with $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, respectively, commute if for every $u \in U$:

—$w_{A_i}(v, u, M) = w_{A_i}(v, u, \mathcal{D}_B\langle M\rangle)\ \forall i \in \{1, \ldots, n\}, v \in V_{A_i}$;

—$w_{B_j}(v, u, M) = w_{B_j}(v, u, \mathcal{D}_A\langle M\rangle)\ \forall j \in \{1, \ldots, m\}, v \in V_{B_j}$;

—any weighted average of $\mathcal{T}_{A_i}(v), i \in \{1, \ldots, n\}, v \in V_{A_i}$ commutes with any weighted average of $\mathcal{T}_{B_j}(x)$, $j \in \{1, \ldots, m\}, x \in V_{B_j}$.

—if $w_{A_i}(v, u, M) \neq 0$ for some $i \in \{1, \ldots, n\}$ and $v \in V_{A_i}$, then it is also required that:
  —$s_{A_i}(v, u, M) = s_{A_i}(v, u, \mathcal{D}_B\langle M\rangle)$; and
  —$\mathcal{T}_{A_i}(v)$ is not altered by first applying $\mathcal{D}_B$ to $M$.

—if $w_{B_j}(v, u, M) \neq 0$ for some $j \in \{1, \ldots, m\}$ and $v \in V_{B_j}$, then it is also required that:
  —$s_{B_j}(v, u, M) = s_{B_j}(v, u, \mathcal{D}_A\langle M\rangle)$; and
  —$\mathcal{T}_{B_j}(v)$ is not altered by first applying $\mathcal{D}_A$ to $M$.

In practice, it may be difficult to arrange for the values of the strength fields and weighting fields to remain unaltered after the application of another warp.[18] Within the framework, a simple alternative is available that provides commutativity between some warps without requiring transformation and field values of one warp to be unchanged by the application of another. Instead of computing the strength field and weighting field values on the warped model $\tilde{M}$ (which is either $\mathcal{D}_A\langle M\rangle$ or $\mathcal{D}_B\langle M\rangle$, depending on

[18]The reason for this is that many strength fields and weighting fields are defined spatially, so that if the point positions of the model are altered by another deforming function, so are the field values.

which warp is applied first), we simply precompute the transformation and field values on the original model $M$ and use these to define the warp. In this case, the only requirement remaining is that the parameterized transformation values commute. This can be formalized by the following more practical condition:

*Deforming Function Commutativity.* If the deforming functions $\mathcal{D}_A$ and $\mathcal{D}_B$ for two warps $A$ and $B$ (with parameterized transformations $\mathcal{T}_{A_i}$ and $\mathcal{T}_{B_j}$, strength fields $s_{A_i}$ and $s_{B_j}$, and weighting fields $w_{A_i}$ and $w_{B_j}$, with $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, respectively) are computed on a model $M$: that is, $\mathcal{D}_A(u, M)$ and $\mathcal{D}_B(u, M)$ are computed for all $u \in U$, then $\mathcal{D}_A$ and $\mathcal{D}_B$ commute if any weighted average of $\mathcal{T}_{A_i}(v), i \in \{1, \ldots, n\}, v \in V_{A_i}$ commutes with any weighted average of $\mathcal{T}_{B_j}(x), j \in \{1, \ldots, m\}$, $x \in V_{B_j}$.

This single condition is easily met in at least one important case. A weighted average of translations is a translation, and all translations commute; thus, if all transformation values in a set of warps are translations, then their precomputed deforming functions commute. This provides commutativity for large classes of warps, including all warps based on point features.

### 7.2 Interpolation and Approximation

In the case of feature-based warps, an important property is whether a warp is *interpolating* or *approximating*. In the framework, two useful concepts dealing with interpolation can be defined: *interpolation* and *approximate interpolation*.

An interpolating warp such as "Wires" [Singh and Fiume 1998] maps model points located on source features to corresponding locations on target features (see Figure 6). Approximating warps such as freeform deformation [Sederberg and Parry 1986], on the other hand, approximate the movement of source features to their target positions in the deformation they apply to model points. Usually, approximating warps are smoother than interpolating warps, but interpolating warps can provide more intuitive user control, since model points respond to feature manipulation as though the features are part of the model.

As we noted in Section 3, feature-based warps map feature specifcations to parameterized transformations. We generally construct parameterized transformations so that each point on a source feature corresponds to a single transformation value that maps the point on the source feature to a corresponding point on the target feature. In this case, a few simple conditions for strength fields and weighting fields guarantee that a feature-based warp is interpolating:

*Interpolation of Feature-Based Warps.* Suppose $M(u), u \in U$ is a model point with the same position as some point on a source feature, and that $\mathcal{T}_i(v), v \in V_i$ is the transformation value associated with that point. Then, a feature-based warp is interpolating if, for all such points:

—$\mathcal{T}_i(v)$ takes the point on the source feature to its corresponding location on the target feature;
—$s_i(v, u, M) = 1$;
—$w_i(v, u, M) \neq 0$;
—$w_i(x, u, M) = 0 \; \forall x \in V_i, x \neq v$; and
—$w_j(y, u, M) = 0 \; \forall j \neq i, y \in V_j$.

In practice, these conditions can be difficult to arrange, especially while maintaining desired smoothness in the warp. However, we can obtain satisfactory results with a warp that is visually interpolating, or *approximately interpolating*. To achieve this, a feature's transformation must act with essentially full strength and dominate the weight contribution of other transformations. Thus, the conditions for approximate interpolation are:

*Approximate Interpolation of Feature-Based Warps.* Suppose $M(u), u \in U$ is a model point that has the same position as some point on a source feature, and that $\mathcal{T}_i(v), v \in V_i$ is the transformation value associated with that point. Then, a feature-based warp is approximately interpolating if, for all such points:

—$\mathcal{T}_i(v)$ takes the point on the source feature to its corresponding location on the target feature;
—$s_i(v, u, M) \simeq 1$;
—$w_i(v, u, M) \gg w_i(x, u, M)\ \forall x \in V_i, x \neq v$; and
—$w_i(v, u, M) \gg w_j(y, u, M)\ \forall j \neq i,\ y \in V_j$.

### 7.3 Continuity

One of the most important properties a warp can possess is a continuity guarantee. This is useful in many applications to guarantee that the warp has a certain quantitatively defined smoothness. Within the framework, it is possible to develop continuity guarantees for a warp's deforming function $\mathcal{D}(u, M)$ and for the warped model $\tilde{M}$, obtained by applying the deforming function to the model $M$.

There are three continuity values that are of interest: first, the continuity of the deforming function with respect to the space in which the model is embedded; second, the continuity of the warped model $\tilde{M}$ with respect to space; and third, the continuity of the deforming function with respect to the model itself (useful primarily to determine if applying the deforming function introduces discontinuities to the model). In this section, we use the framework to develop relations between these continuities and the continuities of strength fields and weighting fields. For brevity, we cover only the first two continuity values; for a more complete discussion, see Milliron [1999].

We would like to derive a minimum continuity guarantee for a warp's deforming function with respect to the space in which the model $M$ is defined. This is useful for several reasons. First of all, depending on the warp, this may be completely independent of the particular model being warped. In particular, if the warp is defined spatially, the continuity of the deforming function with respect to space is independent of the continuity of the model with respect to space. This has the consequence that many models can be passed through a single deforming function and similar continuity guarantees can be made about each of the resulting warped models. Second, the most important continuity in an application is usually the continuity of the resulting model $\tilde{M}$ with respect to space. As we will see, the continuity of the deforming function $\mathcal{D}(u, M)$ is important for the continuity of $\tilde{M}$.

*Deforming Function Continuity with Respect to Space.* Let $\mathbb{R}^n$ be the space in which the model $M$ is embedded (i.e., $\text{Range}(M) \subset \mathbb{R}^n$). Then, a warp's deforming function $\mathcal{D}(u, M)$ is $C^k$ continuous with respect to $\mathbb{R}^n$ at a model point $M(u),\ u \in U$, where $k$ satisfies:

$$k \geq \text{MIN}_{i,v}(\text{CONTIN}_{\mathbb{R}^n}(w_i(v, u, M)), \text{CONTIN}_{\mathbb{R}^n}(s_i(v, u, M)))$$

where $\text{CONTIN}_f(g) = i$ if $g$ is $C^i$ continuous with respect to $f$ and $\text{MIN}_{a,b}(f, g) = j$ if the minimum value of $f$ and $g$ over all values of $a$ and $b$ is $j$.

This continuity guarantee follows from the following two facts: first, all transformations are $C^\infty$ continuous functions; and second, the continuity of the weighted average $\mathcal{D}(u, M)$ is bounded by the continuities of the functions being averaged (the transformation values multiplied by the strength field values: $s_i(v, u, M) \cdot \mathcal{T}_i(v)$) and the continuities of the weights themselves (the weighting field values: $w_i(v, u, M)$). Note that it is impossible to exactly compute the continuity of the deforming function in this most general scheme: $\mathcal{D}(u, M)$ might have higher continuity, depending on specific relationships between all weighting fields and strength fields.

Usually, the most important continuity quantity in an application is the continuity of the deformed model itself. A method for computing this continuity gives the warp designer a valuable tool for ensuring that the warped model is smooth in a strict mathematical sense. Within the framework, and given the continuity guarantee above, we can develop an obvious minimum guarantee for the continuity of the warped model:

*Deformed Model Continuity with Respect to Space.* Let $\mathbb{R}^n$ be the space in which the model $M$ is embedded (i.e., Range($M$) $\subset \mathbb{R}^n$). Then, the warped model $\tilde{M} = \mathcal{D}\langle M \rangle$ is $C^k$ continuous with respect to $\mathbb{R}^n$ at a model point $\tilde{M}(u),\ u \in U$, with $k$ satisfying:

$$k \geq \text{MIN}(\text{CONTIN}_{\mathbb{R}^n}(M), \text{CONTIN}_{\mathbb{R}^n}(\mathcal{D}))$$

$$= \text{MIN}(\text{CONTIN}_{\mathbb{R}^n}(M), \text{MIN}_{i,v}(\text{CONTIN}_{\mathbb{R}^n}(w_i(v,u,M)), \text{CONTIN}_{\mathbb{R}^n}(s_i(v,u,M)))),$$

where $\text{CONTIN}_f(g) = i$ if $g$ is $C^i$ continuous with respect to $f$, $\text{MIN}(f,g) = j$ if the minimum value of $f$ and $g$ is $j$, and $\text{MIN}_{a,b}(f,g) = l$ if the minimum value of $f$ and $g$ over all values of $a$ and $b$ is $l$.

### 7.4 Invertability

In many cases, it is useful to find a warp's inverse, or to determine if a warp is invertible at all. Within the framework, we can compute the inverse of *any* warp that has one. To see how, note that every deformation expressed in the framework produces a transformation-valued deforming function $\mathcal{D}(u, M)$. To invert the deformation caused by this deforming function, we can simply apply the inverse transformation[19] $(\mathcal{D}(u, M))^{-1}$. Formally, the inverse of the deforming function $\mathcal{D}(u, M)$ is given by $\mathcal{D}^{-1}(u, M) = (\mathcal{D}(u, M))^{-1}\ \forall u \in U$.

While this formulation provides a way to compute the inverse of any warp, it suffers from a serious problem: it does not give us much information. A more useful way of computing the inverse would compute the parameterized transformations, strength fields, and weighting fields for the inverse, rather than just the final transformations. Unfortunately, computing these values is not generally possible; however, we can consider a special case in which these values can be computed.

Within the framework, one obvious way to attempt to construct a warp's inverse is by inverting its parameterized transformations, and using these in the computation of a deforming function. It is easy to see that this will correctly compute the inverse in the special case where the strength fields and weighting fields evaluate identically on both the undeformed and deformed models.[20] More formally:

*Inverting a Warp by Inverting its Transformations.* Given a warp with parameterized transformations $\mathcal{T}_i(v)$, strength fields $s_i(v, u, M)$, weighting fields $w_i(v, u, M)$, and deforming function $\mathcal{D}(u, M)$ (with $i \in \{1, \ldots, n\}, v \in V_i$), an inverse warp can be constructed by using parameterized transformations $\mathcal{T}_i^{-1}(v)$, strength fields $s_i(v, u, M)$, and weighting fields $w_i(v, u, M)$ if the following conditions hold for all $u \in U$:

—$w_i(v, u, M) = w_i(v, u, \mathcal{D}\langle M \rangle)\ \forall i \in \{1, \ldots, n\}, v \in V_i$;
—if $w_i(v, u, M) \neq 0$ for some $i \in \{1, \ldots, n\}, v \in V_i$, then it is also required that $s_i(v, u, M) = s_i(v, u, \mathcal{D}\langle M \rangle)$.

This form of warp inverse is particularly interesting for feature-based warps. To see why, consider that the natural way to invert a feature-based warp is to swap source features for target features and vice-versa and compute a new warp. Swapping the features has the effect of inverting the warp's

[19] Of course, $\mathcal{D}(u, M)$ may not have an inverse, as in the case of a projection transformation, in which case the framework obviously cannot help to compute one.

[20] Note that, as in Section 7.1, the condition can be lifted for transformations whose weighting fields evaluate to zero, since these have no effect on the warp.

parameterized transformations. So, this feature-swapping approach will correctly compute the inverse deformation if (by the conditions above) the strength fields and weighting fields evaluate the same on both the undeformed and deformed models. This is very seldom the case, and is difficult to arrange in many warps. Fortunately, this restriction can be removed with a technique similar to the one we present in Section 7.1: here, we precompute field values on the undeformed model and use them later to compute the inverse warp.

## 8. DISCUSSION

In this section, we discuss various implications and characteristics of the framework in greater depth, provide an overview of implementation and efficiency, and describe a few areas of possible future research.

### 8.1 Varying Warps Over Time

Controlling a warp's evolution over time plays an important role in morphing and animation. The framework provides several mechanisms for controlling this evolution.

In morphing, the user smoothly varies a deformation from zero effect to full effect, to create a smooth interpolation from undeformed model to deformed model. Within the framework, there are four straightforward ways to achieve this goal. The first is to compute a deforming function $\mathcal{D}(u, M)$ that warps the undeformed model $M$ to the fully deformed model $\tilde{M}$ and then to interpolate between this deforming function and the identity transformation. A second approach interpolates between the identity transformation and the transformations $\mathcal{T}_i(v)$ for each $i \in \{1, \ldots, n\}$, $v \in V_i$. In this case, each transformation can be controlled independently for greater flexibility. Since each parameterized transformation typically relates to a specific part of the model, this technique can be used to schedule the morph so that different parts of the model morph at different rates. A third approach in the same vein scales the strength fields by animated morph parameters varying from zero (no deformation) to one (full deformation). If the displacement method is used to multiply transformations by scalars, this is identical to interpolating between $\mathcal{T}_i(v)$ and the identity transformation. For feature-based warps, a fourth way to control the morph over time is to animate the target features themselves from source positions to target positions.

In animation, deformations may be controlled over time to achieve a variety of other effects. In this more general case, we change the deformation over time by varying parameterized transformations, strength fields, or weighting fields. Most commonly, users vary parameterized transformations, or the parameters or features that control them. This is especially true of feature-based warps, where parameterized transformations are determined by source and target features, and features serve as animation controls. For example, this is the case in FFDs, Beier and Neely's image warp, "Wires," and our new mesh warp (see Sections 4.1, 4.2, 4.4, and 6).

### 8.2 Other Types of Construction Functions

Because warps defined geometrically are the most popular class of deformations, and because our framework is specifically designed to express these warps elegantly, we focus primarily on construction functions that compute parameterized transformations, strength fields, and weighting fields in a geometric manner. However, other types of construction functions are appropriate for particular applications. Here, we examine two important cases.

Some warps use numerical methods to solve for deformations that satisfy certain mathematical conditions. For instance, Litwinowicz and Williams [1994] define an image warp that minimizes an energy function, Whereas Hsu et al. [1992] solve for an FFD that satisfies direct manipulation constraints. Within the framework, we can think of these techniques as construction functions that solve

for transformations and fields to satisfy constraints, rather than allowing direct user control of those transformations and fields.

Physics-based deformations comprise another class of warps. A physical simulation computes the warped model shape given the original model and a description of forces acting on it. Warps in this category include clothing simulation and particle systems. In the framework, we can think of these warps as having construction functions that implement physical simulation engines and output parameterized transformations and fields based on the output of the simulation. Our framework does not describe such deformations well, since they typically act on each model point independently. One possibility for physics-based deformations does present itself within the framework: a physics-based deformation may be able to sample the model coarsely and use the strength field and weighting field mechanisms to interpolate a few samples smoothly across the finely sampled model. This technique has been explored in "Wires," where a physical simulation performed on wire curves indirectly deforms a smooth surface [Singh and Fiume 1998]. This approach greatly reduces the complexity of the physical simulation.

### 8.3 Inverse Mapping

In the discussion so far, we have used "forward mapping" to deform a model. Often, for regularly sampled models—such as images or voxel volumes—"inverse mapping" may be appropriate. In inverse mapping, the destination model (i.e., the model that will comprise the deformed model) is sampled and each model point warped to determine to which point of the source model it corresponds. In this case, parameterized transformations are the inverse of the forward mapping ones, and we evaluate strength fields and weighting fields on the destination model instead of on the source model.

### 8.4 Implementation and Efficiency

We created all examples in this paper using a single common warping library implemented in C++. A central class serves as an engine that computes the discrete mathematical formulation of the framework, given in Eq. (3). The warp designer derives subclasses from an abstract base class that represents a construction function, implementing the appropriate methods.

Although we have shown how to express existing warps in our framework, we cannot claim that the general framework implementation will be the most efficient. The utility of the framework is that warps may be easily analyzed, compared, designed, and implemented. In a production system, optimized implementations of particular warps may be appropriate.

### 8.5 Future Work

The framework we present in this article is useful both as a conceptual tool and for its practical implementation advantages. In addition, this research suggests a number of areas for future work:

*Surface Region Warping.* We are currently working on a new warp we call the *surface region warp*. The warp deforms regions on one surface to align with regions on another. The application we have in mind for this warp is deforming surfaces in contact during animation, in particular when one object is rigid and another is deformable. The rigid surface might cause a dent in the other, or might stick to it and stretch it out of shape. We can create the surface regions to define the warp by automatically computing intersections when two surfaces come into contact.

*Distance Metrics.* Many warps incorporate a measure of distance in order to specify a smooth fall off of strength fields and weighting fields. For many applications in image and volume warping, a simple Cartesian measure for distance works well. However, for warps on surfaces, the fields might better fall

off according to distance along the surface. Research into efficient and effective distance metrics for more complex cases such as surfaces should prove to be a valuable area of research.

*Efficient Hierarchical Sampling.* Recall that the framework as it is presented in Sections 3.4 and 3.5 suggests defining continuous parameterized transformations, strength fields, and weighting fields and then sampling them to produce discrete quantities. Usually, however, warps compute and sample at the same time. Sometimes, as in the case of image models, the sampling performed can be expensive. We believe that research into general algorithms for efficient sampling using hierarchical methods is a promising avenue for future research.

## ACKNOWLEDGMENTS

We are grateful for the advice and support of Delia Markiewicz, Tom Porter, and Eugene Fiume, as well as helpful suggestions from the anonymous reviewers.

## REFERENCES

ARAD, N., DYN, N., REISFELD, D., AND YESHURUN, Y. 1994. Image warping by radial basis functions: Application to facial expressions. *CVGIP*: *Graphical Models and Image Processing 56*, 2 (Mar.), 161–172.

BARR, A. H. 1984. Global and local deformations of solid primitives. In *Computer Graphics* (*SIGGRAPH '84 Proceedings*) Hank Christiansen, Ed. ACM, New York, pp. 21–30.

BECHMANN, D. 1994. Space deformation models survey. *Comput. Graph. 18*, 4, 571–586.

BEIER, T. AND NEELY, S. 1992. Feature-based image metamorphosis. In *Computer Graphics* (*SIGGRAPH '92 Proceedings*), Edwin E. Catmull, Ed. ACM, New York, pp. 35–42.

BORREL, P. AND RAPPOPORT, A. 1994. Simple constrained deformations for geometric modeling and interactive design. *ACM Trans. Graph. 13*, 2 (Apr.), 137–155.

COHEN-OR, D., SOLOMOVICI, A., AND LEVIN, D. 1998. Three-dimensional distance field metamorphosis. *ACM Trans. Graph. 17*, 2 (Apr.), 116–141.

COQUILLART, S. 1990. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In *Computer Graphics* (*SIGGRAPH '90 Proceedings*), Forest Baskett Ed. ACM, New York, pp. 187–196.

CORRÊA, W. T., JENSEN, R. J., THAYER, C. E., AND FINKELSTEIN, A. 1998. Texture mapping for cel animation. In *SIGGRAPH 98 Conference Proceedings*, Michael Cohen, Ed. ACM, New York, pp. 435–446.

DECAUDIN, P. 1996. Geometric deformation by merging a 3D-object with a simple shape. In *Graphics Interface '96*, Wayne A. Davis and Richard Bartels, Ed. Canadian Information Processing Society, Canadian Human-Computer Communications Society, pp. 55–60.

FORSEY, D. AND BARTELS, R. 1998. Hierarchical B-spline refinement. *Comput. Graph. 22*, 4, 205–212.

GREISSMAIR, J. AND PURGATHOFER, W. 1989. Deformation of solids with trivariate B-Splines. In *Eurographics '89*, W. Hansmann F. R. A. Hopgoood, and W. Strasser, Eds. Eurographics, North Holland, Amsterdam, The Neterlands, pp. 137–148.

HSU, W. M., HUGHES, J. F., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. In *Computer Graphics* (*SIGGRAPH '92 Proceedings*), Edwin E. Catmull, Ed. ACM, New York, pp. 177–184.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98 Conference Proceedings*, Michael Cohen, Ed. ACM, New York, pp. 105–114.

LAZARUS, F., COQUILLART, S., AND JANCENE, P. 1994. Axial deformations: An intuitive deformation technique. *Comput.-Aided Des. 26*, 8 (Aug.), 607–613.

LEE, S.-Y., CHWA, K.-Y., HAHN, J., AND SHIN, S. Y. 1996a. Image morphing using deformation techniques. *J. Visual. Comput. Animat. 7*, 1, 3–23.

LEE, S.-Y., CHWA, K.-Y., SHIN, S. Y., AND WOLBERG, G. 1995. Image metamorphosis using snakes and free-form deformations. In *SIGGRAPH 95 Conference Proceedings*, Robert Cook, Ed. ACM, New York, pp. 439–448.

LEE, S-Y., WOLBERG, G., CHWA, K.-Y., AND SHIN, S. Y. 1996b. Image metamorphosis with scattered feature constraints. *IEEE Trans. Visual. Comput. Graph. 2*, 4 (Dec.).

LERIOS, A., GARFINKLE, C. D., AND LEVOY, M. 1995. Feature-Based volume metamorphosis. In *SIGGRAPH '95 Conference Proceedings*, RobertCook, Ed. ACM, New York, pp. 449–456.

LITWINOWICZ, P. AND WILLIAMS, L. 1994. Animating images with drawings. In *Proceedings of SIGGRAPH '94*, (Orlando, Fla., July 24–29). Andrew Glassner, Ed. ACM, New York, pp. 409–412.

MacCracken, R. and Joy, K. I. 1996. Free-Form deformations with lattices of arbitrary topology. In *SIGGRAPH '96 Conference Proceedings*, Holly Rushmeier, Ed. ACM, New York, pp. 181–188.

Milliron, T. S. 1999. A framework for geometric warps and deformations in computer graphics. B.S.E. Senior Thesis, Princeton Univ., Princetion, N.J.

Sederberg, T. W. and Parry, S. R. 1986. Free-form deformation of solid geometric models. In *Computer Graphics* (*SIGGRAPH '86 Proceedings*), David C. Evans and Russell J. Athay, Ed. ACM, New York, pp. 151–160.

Singh, K. and Fiume, E. 1998. Wires: A geometric deformation technique. In *SIGGRAPH 98 Conference Proceedings*. Michael Cohen, Ed. ACM, New York, pp. 405–414.

van Overveld, C. W. A. M. and Stalpers, M. G. J. R. 1997. Deforming geometric models based on a polygonal skeleton mesh. *J. Graph. Tools 2*, 3.

Wolberg, G. 1990. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, Calif.

Zorin, D., Schröder, P., and Sweldens, W. 1997. Interactive multiresolution mesh editing. In *SIGGRAPH '97 Conference Proceedings*, Turner Whitted, Ed. ACM, New York, pp. 259–268.

Received June 1999; revised September 2001; accepted November 2001