Warp Sculpting

James Gain, *Member, IEEE,* and Patrick Marais, *Member, IEEE*

***Abstract*— The task of computer-based free-form shape design is fraught with practical and conceptual difficulties. Incorporating elements of traditional clay sculpting has long been recognized as a means of shielding the user from these complexities.**

We present *warp sculpting*, a variant of spatial deformation, which allows deformations to be initiated by the rigid body transformation or uniform scaling of volumetric tools. This is reminiscent of a tool imprinting, flexing and molding clay. Unlike previous approaches, the deformation is truly interactive. Tools, encoded in a distance field, can have arbitrarily complex shapes. Although individual tools have a static shape, several tools can be applied simultaneously.

We enhance the basic formulation of warp sculpting in two ways. Firstly, deformation is toggled to automatically overcome the problem of "sticky" tools, where the object's surface clings to parts of a tool that are moving away. Secondly, unlike many other spatial deformations we ensure that warp sculpting remains foldover-free and hence prevent self-intersecting objects.

***Index Terms*— Interactive Modelling, Spatial Deformation, Self-Intersection, Remeshing**

I. Introduction

VIRTUAL sculpting has long been touted [1], [2] as a natural and intuitive solution to the complex task of free-form modeling. The familiar physical action of molding and manipulating clay is linked to the difficult (and, for many artists, unfamiliar) process of computerized shape design. This is achieved by loosely emulating the molding of an inelastic substance, such as modeling clay or silicone putty.

There are four broad approaches to virtual sculpting:

1) *Surface Displacement.* Using decay functions [1] or displacement maps [3] a surface can be locally perturbed. But these deformations act only on an object's surface and are not transmitted across its interior. As Zwicker *et al.* [3] concede, they are only suitable for small to moderate modifications.
2) *Physical Simulation.* Recently, mass-spring [4], finite element [5] and boundary element [6] methods have been employed in real-time dynamically-driven deformation. This has the potential for unprecedented physical accuracy but the high computation costs place a limit on model resolution in interactive design.
3) *Boolean Operators.* A volumetric tool can be used to accrete material by boolean addition or remove material by boolean subtraction from a volumetric object represented as a 3D voxel map [7] or adaptive distance field (ADF) [8]. This is reminiscent of carving. One weakness of these operators is that global operations overwrite and destroy localized detail. Conceivably, this could be surmounted by multiresolution methods, but we have yet to see research on the implications of such a modification. For example, a bend must be placed in an object before detail is introduced. This order dependence is a hindrance to exploratory design.
4) *Spatial Deformation.* Barr [9] and Sederberg and Parry [2] were the first to indirectly reshape an object by warping the surrounding space. This approach has the virtues of computational efficiency, applicability to a variety of object representations, and order-independent localized or global effect. The results are similar to molding a malleable substance. Deformations are initiated by a range of manipulators - points, curves, frames and surfaces - but truly general volumetric tools have, until now, rarely been supported.

Boolean operators promote a tool-based *carving* metaphor. A tool-based *molding* metaphor is a helpful symmetry. This is likely to be intuitive to many artists who are familiar with using tools, such as a chisel, stylus or even their hands, during physical sculpting.

This paper introduces *warp sculpting*, a hybrid of spatial deformation and distance fields. Deformations are created by the rigid body transformation (with pose-interpolating screw motions) or uniform scaling of volumetric manipulators, loosely simulating a tool imprinting, flexing or molding clay (see figure 1). Manipulators are represented, with no restrictions on shape complexity, by a precomputed distance field. This enables efficient reconstruction of the distance from a sampled point on the object (a vertex) to the manipulator's surface. A decay function is applied to smoothly taper the distance

Manuscript received May 2004; revised November 2004.

J. Gain and P. Marais are members of the CVC laboratory at the University of Cape Town.

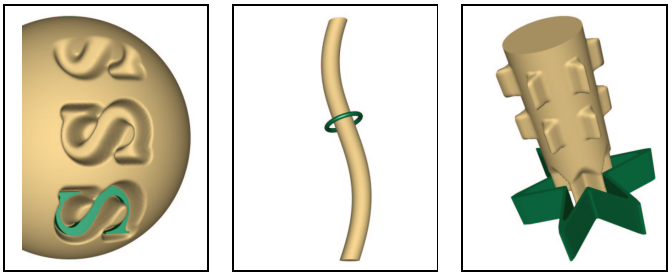

Fig. 1. Warp sculpting with volumetric tools: translation (imprinting 'S' into a sphere), rotation (twisting a tube with a torus) and scaling (a toothed ring is expanded from within and contracted from outside a cylinder).

value and also demarcate an offset region of influence around the manipulator. The movement of the manipulator is moderated by the influence value and transmitted to a vertex. In addition, we cater for multiple simultaneous manipulators. Our novel contributions are: to improve usability by activating deformations only where the manipulator presses into the object and to provide rigorous conditions for preventing self-intersection.

A limitation of warp sculpting is that individual manipulators have a static shape. This is crucial in enabling precomputation of the manipulator's distance field. We believe that this restriction can be supported by parallels with physical sculpting, which often involves using a relatively hard tool to mould a soft substance, such as a stylus with clay. Furthermore, while individual shapes are static, the relative position and orientation of a combination of manipulators is not.

Warp Sculpting exhibits many useful characteristics:

- *Smooth:* The deformation is at least C^1 continuous. This imparts a smoothness reminiscent of clay and is, incidentally, a subsidiary requirement for preventing self-intersection. This does not prevent a designer from introducing sharp creases with an edged tool (and a finely sampled distance field). Such features are sharp at course scales but C^1 smooth when examined at finer scales.
- *Foldover-Free:* Self-intersections in deformable space often results in self-intersection of an embedded object. This is highly unrealistic and contravenes two-manifold representations and single sheet textures. Formally, warp sculpting is a homeomorphism: it is foldover-free and allows a numeric inverse, necessary for ray tracing and useful for undo operations.
- *Variable Local Control:* A user is afforded dynamic and independent control over the region of influence around individual manipulators. Furthermore, global low frequency deformation with broad influence does not destroy high frequency detail.
- *Interactive:* Warp sculpting is interactive, capable of > 10 deformations/s for up to 82000 vertices on typical current processors. Vertices lying outside the region of influence can be quickly bypassed.
- *Accurate:* The deformation obeys the constraints imposed by the movement of the manipulator. Vertices cannot penetrate into a manipulator and those on the manipulator surface remain in contact while pushed. There is a proviso for multiple tools, which break these properties when they collide. There is also some approximation inherent in the use of a sampled distance field, which tends to smooth sharp features in the tool mesh, but this can be reduced with greater sampling.
- *Intuitive:* Users can apply their experience with physical sculpting to predict the behavior of warp sculpting. In other respects, deformations are repeatable, do not introduce oscillations and are reminiscent of the molding and manipulation of clay. This is not a slavish simulation. Aspects such as the lack of reversibility, strict volume conservation and effects of gravity are ignored.

Warp sculpting, as with other spatial deformations, is largely independent of the object's specific representation. It can be applied with equal facility to ADF [11], level-set [12], point-sampled [3] or polygon-mesh objects. Different representations carry different benefits. With ADFs the carving metaphor of Kizamu [8] and the molding metaphor of warp sculpting could be merged in a single system. With level-sets an object's topology could be seamlessly altered during warping. However, we have chosen a traditional triangle mesh for portability and efficiency reasons and defer integration with other representations to the future.

The remainder of this paper has the following structure. Section II covers relevant previous work in the spatial deformation literature. The underlying warp sculpting algorithm is presented in section III. Section IV shows how multiple interacting manipulators are supported. Realism issues, specifically how to deform normals, prevent foldover and overcome clinging effects, appear in section V. The paper concludes with a discussion of performance results (section VI) and a summary and recommendations for future work (section VII). Proofs of the underpinning sufficient conditions for homeomorphism appears in the appendices.

II. Related Work

Spatial deformation as a field arose from the twist, taper and bend deformations of Barr [9] and Sederberg and Parry's [2] Free-Form Deformation. The basic

premise is to indirectly deform an object by warping its ambient space. This is analogous [2] to setting a flexible shape in clear plastic of the same consistency and then flexing the plastic, with a corresponding distortion of the embedded shape. The field is well researched and Bechmann [13] and Milliron *et al.* [14] provide mathematical formalisms as well as useful surveys.

Typically deformations are specified by manipulators, including parametric hyperpatches [2], [15], points [16], [17], curves [10], [18], twisting frames [19] and $2\frac{1}{2}$D surfaces [20]. It is also common to warp a vertex by attaching it to the closest point on a manipulator and imparting the motion of this point, reduced as a function of distance, to the vertex. A given vertex may even be influenced by a combination of manipulators.

True volumetric manipulators are rare. Decaudin [21] allows simple convex volumetric manipulators such as spheres, cubes and ellipsoids. Vertices are parametrized according to their distance from the unique center of the manipulator. This technique has the virtue of allowing smooth, localized and, above all, volume conserving deformations but the variety of tool shapes is severely restricted. One strategy for building volumetric deformation tools would be to approximate them with a collection of other manipulators, say, a cloud of directly manipulated points [16]. Setting aside the difficulties of how to place these manipulators and set their parameters to achieve reasonable fidelity, the main problem is one of scalability. Even simple tools require many manipulators. As tool intricacy increases the number of manipulators (and computational overheads) rapidly become unmanageable. In contrast, the speed of our technique is independent of the tool's complexity.

This raises the question: why have fully general volumetric manipulators not been developed? One reason is that closest point queries on surfaces are computationally expensive and certainly not consistent with interactive modeling. We overcome this with static manipulators for which distance values can be precalculated and reconstructed interactively.

A second reason [14], is that closest distance parametrizations are prone to buckling artefacts, where a C^0 shearing discontinuity is introduced along medial lines where vertices are equidistant from two or more manipulator points. This is symptomatic of a lack of deformation injectivity (one-to-one mapping), which can also cause foldover or self-intersection.

In general the problems of self-intersecting deformations are well recognized [15], [18], [20] but have not been comprehensively addressed. Milliron *et al.* [14] identify and overcome buckling discontinuities in the context of Wires by a convolution strategy. Borrel and Rappoport [17] identify a "space tearing" phenomenon in connection with their Simple Constrained Deformation (ScoDef) technique. Unfortunately, they place the onus of identifying and correcting self-intersection on the user. Blendeforming [23] is a foldover-free (injective) and efficiently invertible deformation designed specifically for ray-tracing. They analyze the deformation of space and arrive at limits on the translation of point and curve manipulations. Our results are more broadly applicable to surface manipulators and cover rotation and scaling as well as translation. Gain and Dodgson [22] develop an automatic test for self-intersection of Free-Form Deformations. They incorporate the test into Directly Manipulated FFD [16] by breaking long point manipulations into shorter injective pieces. We adopt this approach of segmenting manipulations.

There are two notable exceptions to the paucity of volumetric manipulators in Spatial Deformation. Mesh Forging [24] drives deformation using conventional point manipulators but alters the locus of deformation using "occluders" (distance fields defined on implicit functions). The authors liken this to an anvil bracing the shape of an object undergoing forging. However, occluders, unlike warp sculpting tools, are statically placed.

Sweepers [25] also uses multiple tools encoded by distance fields and decay functions. But, the fundamental basis of their tool transformation is different. They exploit Alexa's [26] scalar multiplication operator on matrices to directly weight the transformation of a tool. Admittedly, this is an elegant formulation but, unfortunately, it has two serious consequences. Firstly, it is difficult, given the inherent non-linearities involved, to derive bounds that prevent self-intersection. Unlike our rigorous proof (*cf* Appendices), Angelidis *et al.*'s foldover bounds are stated as an unproven conjecture. Secondly, Alexa's matrix operations are expensive. Angelidis *et al.* do not provide performance results for Sweepers but using Alexa's reported computation costs allows a suitable comparison of the weighted transformation operation. Using Alexa's results, an interpolation costs 5.1×10^{-6}s for Sweepers (scaling from a 1GHz to 1.7GHz machine) as against 2×10^{-7}s for Warp Sculpting (or 19.7K vs. 500K interpolated vertices at 10 frames per second). So, in practice Warp Sculpting is liable to be at least an order of magnitude faster.

III. Warp Sculpting Algorithm

Warp sculpting enables a variety of tools to interact with an object in a way that loosely emulates shaping, imprinting and molding clay. The user controls the position and orientation of tools thereby generating a warping field that indirectly deforms intersected objects.

There are three stages to deforming a vertex on an object's surface in accordance with the transformation of a tool:

1) *Distance Field Reconstruction:* The shortest Euclidean distance from the vertex to the tool is closely approximated by interpolating a sampled distance field. The field samples are pre-processed, greatly accelerating run-time distance calculations. Any vertex outside the bounding box of the distance field is assumed to fall beyond the influence of the tool.
2) *Decay Function Calculation:* A decay function is applied to smoothly taper the distance value so that it is 1 on and inside the tool and 0 at and beyond an offset distance from the tool. The decay function allows a user to interactively adjust the offset distance and hence the region of influence around a tool, subject to the bounds of the distance field.
3) *Weighted Deformation:* The tool's transformation, scaled by the decay value, is applied to the vertex. Thus, vertices close to the tool receive most of the tool's rotation and translation, while those further away are imparted correspondingly less.

Once all vertices within the tool's influence have been deformed, the tool's position and orientation are updated, ready for the user to continue sculpting.

Rigid body transformations emulate a tool moving along a trajectory but it is also useful to support uniform scaling. For instance, shrinking a torus-tool allows a tube-shape to be pinched inwards. Combining rotation, translation and scaling into a single transformation presents unexpected technical difficulties (see section III-C). Also, such complexity is confusing from a user's perspective. Hence, warp sculpting separates rigid body transformations from uniform scaling.

A tool has a local coordinate frame $\mathbf{F}_T$ and origin $\mathbf{o}_T$ within world coordinate space [1]. The tool is given: (a) a rigid body rotation $\mathbf{R}(r)$, specified by an angle of rotation r about a center $\mathbf{o}_R$ and axis $\mathbf{a}_R$, and (b) a rigid body translation, specified by a vector $\mathbf{t}$. Alternatively, the tool could be uniformly scaled by a factor $s > 0$ about a center of scaling $\mathbf{o}_S$. The tool shape is encoded in a scalar distance field $\phi(\mathbf{x}) : \mathbb{R}^3 \mapsto \mathbb{R}$, where the value $\phi(\mathbf{x})$ at a position $\mathbf{x}$ within the field provides the shortest Euclidean distance to the surface of the tool. A decay function $w(d) : \mathbb{R}^+ \mapsto [0,1]$ ensures that deformations are tapered with C^1 continuity as distance $d = \phi(\mathbf{x})$ from the tool surface increases. The value w weighting the tool's transformation is: $w = w(\phi(\mathbf{x}))$. The transformation of the tool is transmitted to an object with n vertices $\mathbf{p}_i; i = 1, \ldots, n$ according to the following algorithm:

FOR $\mathbf{p}_i; i = 1, \ldots, n$
  $\mathbf{x} \leftarrow \mathbf{F_T}(\mathbf{p}_i - \mathbf{o}_T)$
  \\ *map* $\mathbf{p}_i$ *to the local tool coordinate system*
  IF $\mathbf{x} \in (0,1) \times (0,1) \times (0,1)$ THEN
  \\ $\mathbf{x}$ *falls within the distance field*
    $d \leftarrow \phi(\mathbf{x})$
    \\ *tri-quadratic reconstruction of distance*
    $w \leftarrow w(d)$ \\ *decay function smoothly tails off*
    $\mathbf{p}_i \leftarrow \mathcal{D}(\mathbf{p}_i)$ \\ *apply weighted deformation*
$\mathbf{o}_T \leftarrow \mathbf{R}(r)\mathbf{o}_T + \mathbf{t}$ OR $s(\mathbf{o}_T - \mathbf{o}_S) + \mathbf{o}_S$
$\mathbf{F}_T \leftarrow \mathbf{R}(r)\mathbf{F_T}$ OR $s\mathbf{F}_T$
\\ *update tool origin and frame*

$$\mathcal{D}(\mathbf{p}_i) = \begin{cases} \mathbf{R}(wr)(\mathbf{p}_i - \mathbf{o}_R) + w\mathbf{t} + \mathbf{o}_R & \text{rigid} \\ [w(s-1)+1](\mathbf{p}_i - \mathbf{o}_S) + \mathbf{o}_S & \text{scale} \end{cases} \quad (1)$$

We now consider the distance field, decay function and tool transformation in further detail.

A. Distance Field Representation and Reconstruction

A distance field is a shape representation that encodes, in a sampled scalar field, the shortest distance to a surface. Distance fields are widely used in intersection detection, shape blending and offsetting.

As mentioned in section II, many spatial deformation techniques attach an object vertex to the closest point on a manipulator at substantial computational cost. Using a distance field allows much of this cost to be shifted from run-time to a pre-process.

There are a number of decisions inherent in using distance fields: whether to use adaptive sampling, how to distinguish interior and exterior samples and what type of distance metric and sample interpolation to employ. Our decisions are motivated by the need for continuity (required to prevent foldover [22]) and run-time speed (necessary for interactivity).

We employ a true Euclidean metric. Samples interior to a tool are accorded a 0 distance value to ensure that vertices inadvertently trapped inside the tool receive full deformation.

Adaptive sampling of the distance field is appealing since it offers compact storage of tools with varying high and low frequency detail. However, there is an attendant cost of octree traversal [11] to locate a vertex within a lattice cell. With a uniform field $\phi(x, y, z)$ having l, m, n cells evenly distributed on the unit parallelepiped

[1]Matrices and coordinate frames are represented in bold uppercase (e.g., $\mathbf{M}$), points and vectors in bold lowercase (e.g., $\mathbf{v}$) and scalars in italicized roman or greek lowercase (e.g., s, α)

$x, y, z \in [0, 1]$ it is trivial to obtain the i, j, k cell index ($i = \lfloor \ell x \rfloor$) and u, v, w local cell coordinates ($u = \ell x - i$).

The samples of a distance field must be interpolated to obtain a continuous distance value. Since trilinear interpolation is first derivative discontinuous at cell boundaries we use the tri-quadratic reconstruction of Barthe *et al.* [27]. The reconstruction $\phi(x, y, z)$ is everywhere C^1 with $\|\nabla\phi\| \leq 1$.

B. Choosing a Decay Function

The decay function $w(d) : \mathbb{R}^+ \mapsto \mathbb{R}$ determines the smoothness of the deformation surrounding a tool. As such, it must satisfy several specific requirements: have compact support (nonzero only on $[0, 1]$) to bound the region of deformation, and be monotonically decreasing from $w(0) = 1$ to $w(1) = 0$.

There are additional considerations for preventing foldover (*cf* section V-B): $w(d)$ must be at least C^1 on $(0, 1)$ with $w'(0) = w'(1) = 0$. Also, the minimum of $1/w'(d)$ should be as large as possible since this places a practical restriction on the magnitude of individual deformations. Finally, in the interests of interactivity $w(d)$ should be rapidly computable.

There are a number of candidates which satisfy these continuity and shape demands [10], [19], [28]. We choose to use the Wires fn. of Singh and Fiume [10] since it is the least costly and has the largest maximum reciprocal derivative. Thus, the weighting value is:

$$w(d) = \begin{cases} ((\frac{d}{e})^2 - 1)^2 & \text{if } d \leq e \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where the parameter e controls the offset influence of the tool. The region of effective deformation around the tool expands and contracts as e is increased and decreased. If greater geometric continuity is needed then an appropriate decay function can be constructed with piecewise B-spline or Bézier curves.

C. Weighted Deformation from Tool Transformations

Setting aside the issue of tool scaling, a tool's trajectory is likely to be in the form of an initial and final orientation and position. From this a rotation and translation must be derived. Llamas *et al.* [19] show how to find the minimal screw motion between frames, where the translation vector ($\mathbf{t}$) is parallel to the rotation axis ($\mathbf{a}_R; \mathbf{o}_R$) and the rotation angle is minimal ($r \in [0, \pi]$). Translation no longer alters the axis of rotation. The advantages of this will become evident in section V-B.

Deformation of a vertex is an ordered composition of rotation and translation (or uniform scaling) where the rotation angle and translation distance (or scaling factor) are weighted by the decay value. This closely follows the treatment of Llamas *et al.* [19]. All that remains is to construct the rotation matrix from the rotation angle r, decay value w and rotation axis $\mathbf{a}$ and then substitute this into (1):

$$\begin{aligned} \mathbf{R}(wr) &= \sin(wr)\mathbf{M} + (1 - \cos(wr))\mathbf{M}^2 \quad (3) \\ \mathbf{M} &= \begin{bmatrix} 0 & -\mathbf{a}_{r_k} & \mathbf{a}_{r_j} \\ \mathbf{a}_{r_k} & 0 & -\mathbf{a}_{r_i} \\ -\mathbf{a}_{r_j} & \mathbf{a}_{r_i} & 0 \end{bmatrix} \end{aligned}$$

The entire transformation ((1) and (3)) could be composed into a single homogenous transformation matrix, but there are no obvious vector parallelism benefits since it would have to be reconstructed anew for each vertex.

IV. Multiple Tools

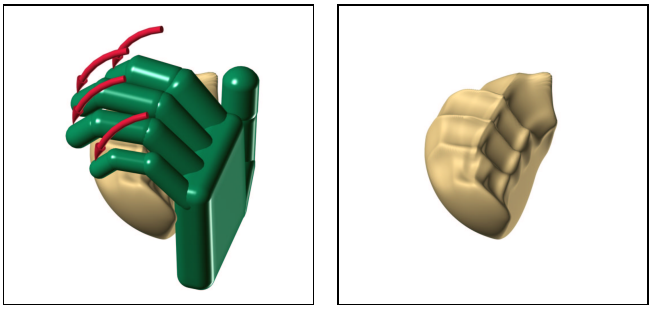

Fig. 2. A dynamic hand tool, constructed from a jointed hierarchy of 16 rigid finger segments, closing to grip a sphere.

There are clear benefits to combining multiple tools. The dynamic interaction of different tools overcomes the static nature of the individual components. Stationary tools can exert an influence, acting as anchors or anvils during stretching or pressing. Assemblies of tools can save on repeated operations. To achieve this the transformation of several tools must be combined into a single simultaneous deformation. In general, this combination of tools should not introduce unseemly bulges, nor allow the object to breach a tool's surface. One strategy for combining tools would be a weighted average in transformation space, which could then be applied to deform a point. Ideally, the average should enforce a linear correspondence between weighting values and the resulting transformation, in the same way that *slerping* (**s**pherical **l**inear **interp**olation) ensures a linear mapping during interpolation between rotations by operating on the quaternion hypersphere. Buss and Fillmore [29] generalize this to a geodesic weighted average of multiple rotations based on a least squares minimization. Unfortunately, this does not incorporate translations and so cannot support full rigid body transformations. Alexa [26] provides a weighted average of fully general

transformation matrices but without a guarantee of linear behavior. Ultimately, both these strategies are rejected as too expensive, requiring numerical iteration for every deformed vertex.

If we allow some non-linearity in the mapping then working in Euclidean space proves to be both simple and effective. Singh and Fiume [10] advocate using the decay value to determine relative weighting in the average. The closer a deformable point lies to a tool's surface the more that tool's movement dominates over others in the average. Assuming, that there are m tools, each with an independent distance field (ϕ_j), radius of influence (e_j), coordinate frame ($\mathbf{F}_{T_j}, \mathbf{o}_{T_j}$) and transformation ($\mathcal{D}_j$), a point is altered by a weighted average of individual tool deformations, $\mathcal{A}(\mathbf{p})$:

$$\begin{aligned} \mathcal{A}(\mathbf{p}) &= \frac{\sum_{j=1}^{m} \mathcal{D}_j(\mathbf{p}) w_j^k}{\sum_{j=1}^{m} w_j^k} \\ w_j &= w_j(\phi_j(\mathbf{F}_{T_j}(\mathbf{p} - \mathbf{o}_{T_j}))) \end{aligned} \tag{4}$$

where k controls the strength of localization around a tool. Setting $k = 0$ produces a conventional average. Low values of k allow squeezed objects to penetrate a tool's interior. High values compress the zone of averaging and tend to create kinks in an object. Experimentally, $k = 3$ seems to behave well as a reasonable compromise.

V. Realism Issues

A. Deformation of Normals

Correct surface properties, such as normal vectors, texture coordinates and vertex colors, are central to the effective lighting and rendering of objects. This topic is much neglected in the spatial deformation literature [2].

For the deformation of normals, the tacit assumption is that these can be averaged from the orientation of the surrounding deformed surface. But, even if the surface is adequately sampled, this is still a gross approximation of derivative behavior. A direct deformation of normals using the covariant transformation rule is preferred.

In fact, accurate normals can then be used as a driver for adaptive curvature-based mesh resampling. This is particularly necessary when a small detailed tool encounters a coarsely sampled object (as is demonstrated in Figure 3). Sharp features can then be captured without forcing the rest of the mesh to have the same level of detail. We adopt the adaptive refinement and decimation scheme of Gain and Dodgson [30].

It is unclear how to treat deformed texture coordinates and vertex colors. Careful resampling is warranted in the case of small distortions [3] but warp sculpting is

[2] with the notable exception of Barr [9]

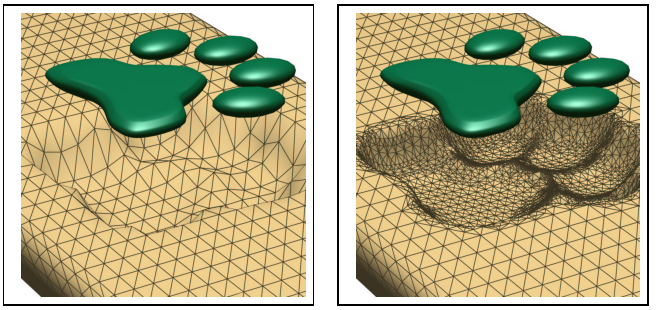

Fig. 3. The necessity for mesh resampling. A pawprint tool is impressed into the object (with wireframe overlaid). [left] Without resampling fine detail is lost. [right] With adaptive refinement tool detail is captured.

designed to allow radical shape alteration. In this instance, texturing works better as a separate post-process, perhaps using texture painting [32], in an analog to the final painting and decorating of clay sculpture. The covariant transformation rule provides a formula for the deformation of a normal ($\mathbf{n} \mapsto \mathbf{n}'$):

$$\mathbf{n}' = \det(\mathbf{J}) \cdot \mathbf{J}^{-T}\mathbf{n} = \det(\mathbf{J})^2 \mathbf{J}^{\star T}\mathbf{n}$$

where $\mathbf{J}$ is the 3×3 Jacobian matrix whose entries are partial derivatives of the deformation function. The transpose of the adjoint ($\mathbf{J}^{\star T}$) can be computed more rapidly [33] than the transpose of the full inverse ($\mathbf{J}^{-T}$) and since we intend normalizing the vector anyway, the coefficient can be dropped ($\mathbf{n}' = \mathbf{J}^{\star T}\mathbf{n}/\|\mathbf{n}'\|$).

For a single tool, the Jacobian has the form:

$$\begin{aligned} \mathbf{J}_{a,b} &= \frac{\partial \mathcal{D}_a}{\partial \mathbf{p}_b} \qquad (5) \\ &= \begin{cases} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{p}_b}(\mathbf{p} - \mathbf{o}_R)\right]_a + \mathbf{R}(wr)_{a,b} + \frac{\partial w}{\partial \mathbf{p}_b}\mathbf{t} & \text{(r)} \\ \left[\frac{\partial w}{\partial \mathbf{p}_b}(s-1)\right](\mathbf{p} - \mathbf{o}_S)_a + [w(s-1)+1]\,\delta_{ab} & \text{(s)} \end{cases} \end{aligned}$$

δ_{ab} is the Kroënecker delta. $\mathrm{col}_b(\mathbf{F}_T)$ is the bth column of $\mathbf{F}_T$. Further:

$$\frac{\partial w}{\partial \mathbf{p}_b} = 4\left[\left(\frac{d}{e}\right)^2 - 1\right]\frac{d}{e^2}\nabla\phi \cdot \mathrm{col}_b(\mathbf{F}_T) \tag{6}$$

$$\frac{\partial \mathbf{R}}{\partial \mathbf{p}_b} = r\frac{\partial w}{\partial \mathbf{p}_b}\left[\cos(wr)\mathbf{M} + \sin(wr)\mathbf{M}^2\right] \tag{7}$$

The distance field gradient $\nabla\phi$ could easily be calculated by the central difference method but a linear derivative affords more accuracy [27].

For multiple tools, the function $\mathcal{A}$ must be considered:

$$\begin{aligned} \mathbf{J}_{a,b} &= \frac{\partial \mathcal{A}_a}{\partial \mathbf{p}_b} \\ &= \frac{1}{(\sum_j w_j^k)^2}\left[\left(\sum_j w_j^k\right)\left(\sum_j \mathcal{D}_j(\mathbf{p})_a\, k w_j^{k-1}\frac{\partial w_j}{\partial \mathbf{p}_b}\right.\right. \\ &\quad \left.\left. + \frac{\partial \mathcal{D}_j(\mathbf{p})_a}{\partial \mathbf{p}_b} w_j^k\right) - \left(\sum_j \mathcal{D}_j(\mathbf{p})_a w_j^k\right)\left(\sum_j k w_j^{k-1}\frac{\partial w_j}{\partial \mathbf{p}_b}\right)\right] \end{aligned}$$

The partial derivatives $\frac{\partial w_j}{\partial \mathbf{p}_b}$ and $\frac{\partial \mathcal{D}_j(\mathbf{p})_a}{\partial \mathbf{p}_b}$ are versions of (6) and (7) with parameters particular to tool j.

Normal deformation is fairly expensive: deforming a normal is six times as costly as deforming a vertex. However, much of the calculation is reusable in other contexts ($\mathcal{D}$ and ϕ during point deformation and $\nabla\phi$ in gradient toggling). We believe that the cost is warranted by the improved accuracy.

B. Preventing Foldover

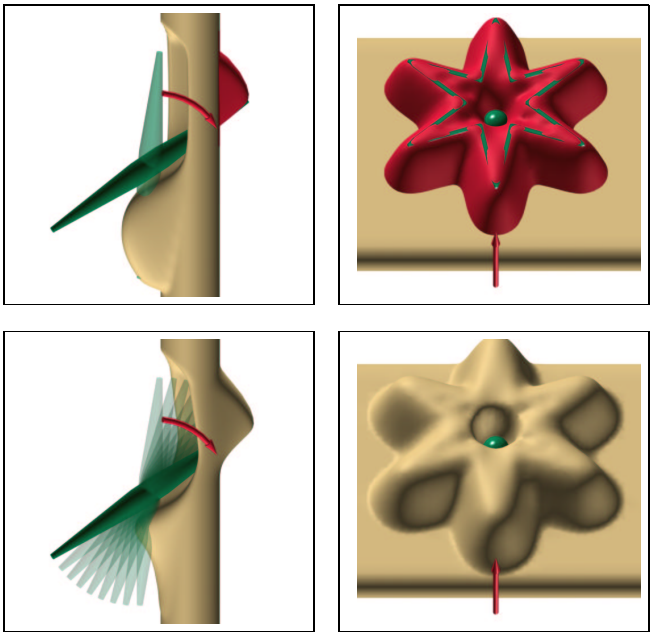

Fig. 4. Preventing self-intersection in warp sculpting: [top] Object foldover and tool-object interpenetration caused by over-large rotation or translation (with object interiors in red). [bottom] Foldover-free deformation by breaking the tool transformation into shorter steps (8 and 32 subdivisions for left and right, respectively).

Many spatial deformation methods generate foldover of the ambient space and potential self-interpenetration of the embedded object. This is physically unrealistic, breaks two-manifold mesh conditions, and prevents a numerical inverse. A typical case is illustrated in Figure 4 [top right], where a cog is raised around a sphere with the object trapped in between. The cog is translated well beyond its initial region of influence and drags the object's surface into self-intersection.

The answer lies in limiting the extent of individual transformations. Following Gain and Dodgson [22], we subdivide large tool transformations into a sequence of intermediate transformations, which are small, contiguous and foldover-free. Figure 4 [bottom right] demonstrates the success of this strategy, with 32 successive transformations moving the tool along the same path and reaching the same position and orientation as Figure 4 [top right], without foldover.

Breaking a tool transformation into n successive interpolating steps requires subdividing the degree of translation and rotation or scaling: $\mathbf{t}_n = \mathbf{t}/n$, $r_n = r/n$ or $s_n = s^{1/n}$. Warp sculpting is applied with these new parameters. After each step the tool frame and origin are updated. Earlier decisions become clear: the axis of rotation and center of scaling remain constant and the final tool configuration after n steps is the same as before. The number of subdivision steps is important; too few and foldover will not be prevented, too many and interactivity degrades. The appendices provide an upper bound on the extent of tool transformation sufficient to ensure foldover-free warp sculpting.

For individual tools, deformations are limited by the following inequalities:

$$\begin{aligned}
&\text{(r):} && \tfrac{1}{e}\left(r_n \alpha_R + \|\mathbf{t}_n\|\right) < \tfrac{\sqrt{27}}{8} && \\
&\text{(s):} && (s_n - 1)\,\tfrac{\alpha_S}{e} < \tfrac{\sqrt{27}}{8} && \text{if } s \geq 1 \\
& && (1 - s_n)\left(1 + \tfrac{8}{\sqrt{27}}\tfrac{\alpha_S}{e}\right) < 1 && \text{if } 0 < s < 1
\end{aligned}$$

The size of a deformation step thus depends on the magnitude of scaling (s), rotation (r), translation ($\|\mathbf{t}\|$) and tool effect (e); α_R or α_S, encode the distance of active deformation from the center of rotation ($\mathbf{o}_R$) or scaling ($\mathbf{o}_S$), respectively, to the furthest fringe of the tool's effect.

The inequalities for multiple tools are:

$$\begin{aligned}
&\sqrt{3}c_n + c_n^2 + \tfrac{1}{\sqrt{27}}c_n^3 < 1 \\
&\text{(r)}: c = \tfrac{8}{\sqrt{27}} \max_j \left[\tfrac{1}{e_j}\left(\alpha_{R_j}\left(r_{n_j} + k\sqrt{2 - 2\cos r_{n_j}}\right)\right)\right. \\
&\qquad \left. + (k+1)\|\mathbf{t}_{n_j}\|\right] \\
&\text{(s)}: c = \tfrac{8}{\sqrt{27}}\overline{s}_n \max_j \left[\tfrac{1}{e_j}\alpha_{S_j}\right] + \overline{s}_n \\
&\overline{s}_n = \overline{s}^{\frac{1}{n}} \\
&\overline{s} = \begin{cases} \max_j(s_j) - 1 & \text{if } \max_j(s_j) > 2 - \min_j(s_j) \\ 1 - \min_j(s_j) & \text{otherwise} \end{cases}
\end{aligned}$$

To find the number of subdivision steps, solve for n in the above inequalities (using a bisection or Newton search) and round up to the nearest integer ($\lceil n \rceil$).

These conditions ensure that Warp Sculpting will not produce a foldover of coordinate space. However, a poorly sampled object, which does not follow the deformation, may still self-intersect. Fortunately, this type of self-intersection can always be removed by adaptively refining the object. In contrast, a spatial foldover is far more damaging because it cannot be circumvented by resampling.

C. Gradient Controls

Warp sculpting acts by influencing the ambient space surrounding a volumetric tool. Unfortunately, this sometimes causes surfaces to unrealistically cling to a tool.

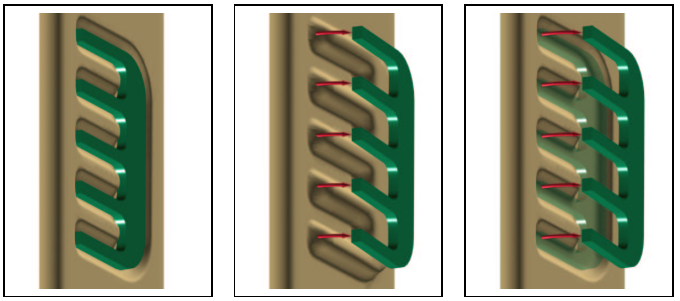

Fig. 5. Preventing clinging objects: [left] A comb tool is pressed into a pad object. [middle] The object is dragged as the tool withdraws. [right] A gradient toggle automatically deactivates warping when required.

For example, during translation an object will conform to and follow a nearby tool even as it is withdrawn (Figure 5 [middle]). In scaling, the contraction of a tool causes the neighboring surface to collapse inwards. Finally, a tool rotated in place imparts a friction-like twist to the surrounding shape. These effects are sometimes desirable, but they need a more controlled and intuitive interface. For instance, extruding a portion of the surface is better achieved with a pincer tool or by protruding a tool from under the skin of the object.

The solution is to automatically deactivate deformation depending on the *swathe* of tool movement and the relative position of tool and object. We refer to this as a *Gradient Toggle*. By examining the deviation of the distance field gradient ($\nabla\phi(\mathbf{x})$) from the normalized deformation vector ($\text{norm}(\mathcal{D}(\mathbf{p}) - \mathbf{p})$) it can be established whether an object point lies on the leading or trailing side of a deformation. Warp sculpting is active if the following inequality is satisfied for any object point within the zone of active deformation of a tool:

$$\tau = \nabla\phi(\mathbf{x}) \cdot [\mathcal{D}(\mathbf{p}) - \mathbf{p}]/\|\mathcal{D}(\mathbf{p}) - \mathbf{p}\| > \tau_\ell \qquad (8)$$

Here, the value τ captures the cosine angle relationship between tool position and tool movement relative to a point, ranging from $\tau = 1$ (aligned) to $\tau = -1$ (opposite). A cutoff of $\tau_\ell = 0.2$ works well in practice. This test is comparatively inexpensive since most of its quantities are needed in any event for the deformation of points and normals.

A simple gradient toggle will not succeed if a tool needs to be simultaneously active and inactive in different regions. For example, as a tube tool is narrowed it is useful if the distance field is operational within the inner radius but suppressed beyond the outer radius.

One solution is to design "multi-tools" with the desired behaviour. In Figure 6[center] a two-part tube with separate inner and outer casings facilitates narrowing and widening without the deformation spreading unduly into the surrounding space. Unfortunately, such multi-tools can be difficult to design since the result depends on the relative influence and positioning of the components.

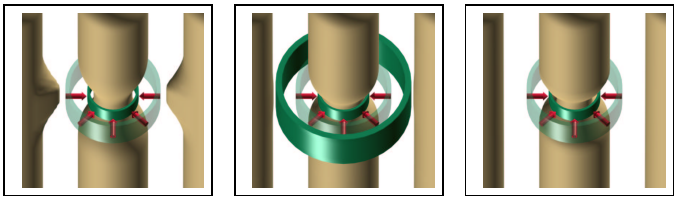

Fig. 6. Delimiting deformation with multiple tools: [left] Shrinking a tube tool affects adjacent objects. [middle] Overcome by adding a static outer tube. [right] Overcome using movement swathes ($\tau_u = 0.0, \tau_\ell = -0.8$)

An alternative is to moderate the decay value according to a piecewise function on τ: $w = f(\tau)w(d)$. We call this a *Gradient Swathe*. We introduce two parameters, τ_u above which deformation is full and τ_ℓ below which deformation is nil. The Wires fn. is used to smoothly interpolate between these bounds

$$f(\tau) = \begin{cases} 1 & \text{if } \tau > \tau_u \\ 0 & \text{if } \tau < \tau_\ell \\ ((\frac{\tau_u - \tau}{\tau_u - \tau_\ell})^2 - 1)^2 & \text{otherwise} \end{cases}$$

Figure 6[right] shows how vertices beyond the outer radius are unaffected because the tool shrinks inwards while the gradient in these regions faces outwards.

Of course suitable changes must also be made in eqn. 6 and the Appendices. The upper bound on the derivative of w in the foldover condition now becomes:

$$\max \|\frac{\partial w}{\partial \mathbf{p}}\| = \frac{8}{\sqrt{27}}(\frac{1}{e} + \frac{1}{\tau_u - \tau_\ell}\frac{2}{c})$$

where c is the axial length of a direction field cell. Incorporating this into the foldover conditions of section V-B thus introduces a dependence on cell size, since even the simplest tool will have medial lines or points with a diametric gradient change across a single cell. Smaller cells merely localize these instantaneous direction changes. So, it is important to achieve a careful balance in the distance field between undersampling (losing tool detail) and oversampling (increasing foldover steps).

The problem can be overcome to an extent by decoupling the sampling resolution of distance and direction. The distance field can be supersampled with a Gaussian filter for the purposes of calculating the gradient, allowing larger lattice cells at the expense of less detailed discrimination between the leading and trailing edges of fine tool detail.

VI. Results

Figure 7 shows stills from the design of a Celtic chalice. The initial circular symmetric form is achieved

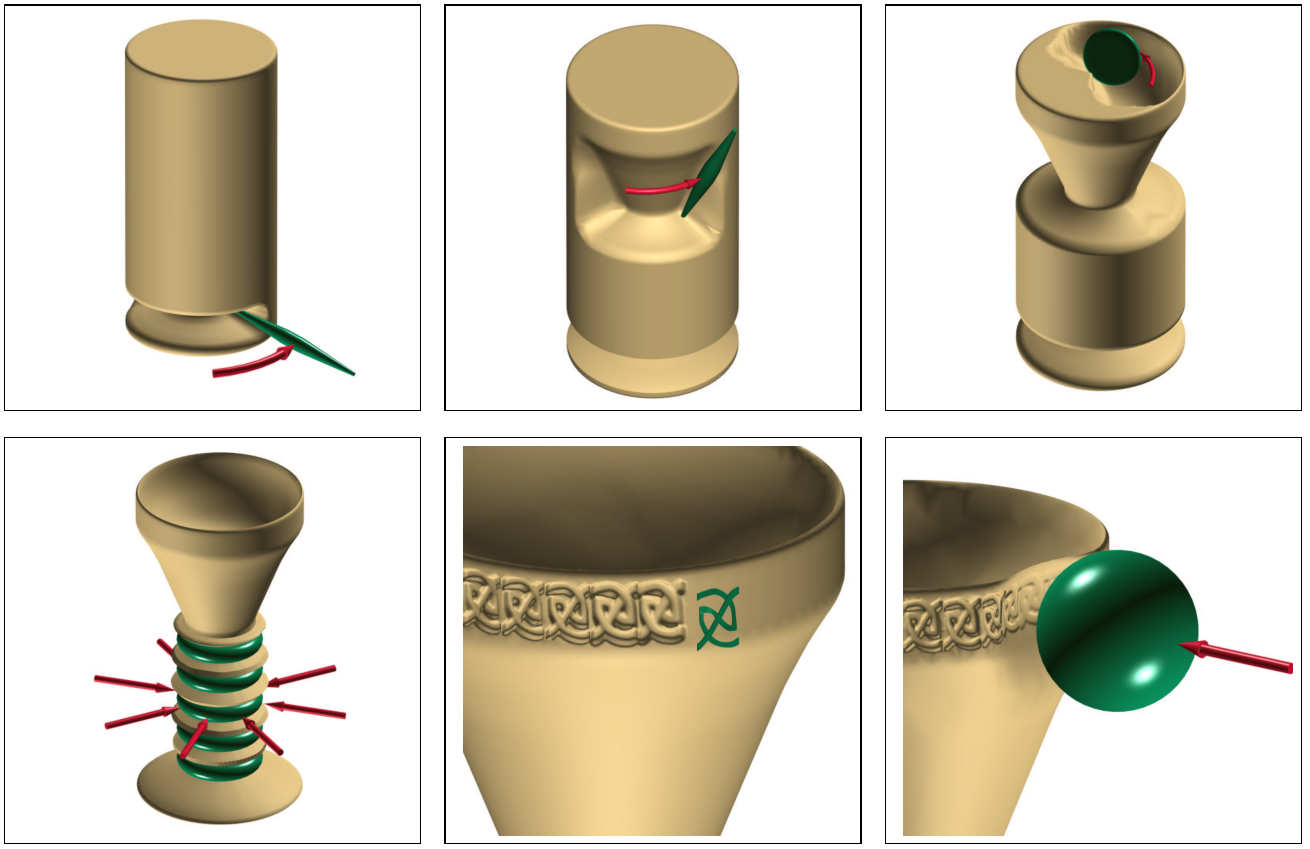

Fig. 7. Sculpting a chalice using a potter's wheel method and various tools, including a spindle, disc, torus, knotwork stamp and sphere.

by emulating a potter's wheel and revolving the object on its central axis. Unfortunately, if the tool is merely held in place the object tends to bunch along the leading side. Rather, the tool is punched repeatedly inwards on each rotation step. By varying the angle of punching and translation along the axis of revolution different effects are possible, such as hollowing out the inner bowl of the chalice. Several other aspects of figure 7 are noteworthy. A rubbed effect is created by subtly expanding the radius of effect for parts of the celtic braiding. Later, a spherical dent flexes but does not destroy the detail of the pattern. The final chalice was created in less than an hour of modelling time.

The performance of warp sculpting has been tested on a 3GHz P4 with 512 MB RAM. For a single tool and one homeomorphic step ($n = 1$ in section V-B) up to 82000 vertices within the tools influence can be interactively deformed at a rate of 10Hz. The impact of other factors is highly variable:

- *Distance Field Density:* Sampling is a once of preprocess and has no direct influence on interactivity, with the notable exception of movement swathes. However, detailed tools (with high density) generally require greater mesh refinement.
- *Radius of Effect (e):* Expanding the influence of the tool (by increasing e) has two contradictory outcomes: more object vertices are warped but in fewer foldover steps.
- *Number of Tools (m):* Warp sculpting is linear in the number of overlapping tools (for $m > 1$). However, there is a sharp jump in cost when moving from one to two tools, since the foldover bounds for multiple tools are more stringent. As a point of reference, warp sculpting with a dynamic hand as in figure 2 ($m = 16$ and $n = 1$) deforms 13250 vertices at 10 Hz. Typically, 1 to 6 finger components influence a given vertex. In practice, the influence of all components seldom overlap completely.
- *Foldover Steps (n):* Tool transformations that are large relative to the area of tool influence require a significant number of subdivisions. For instance, figure 4 requires $n = 8$ and 32 for the single and multiple tool cases. Each foldover step is treated as a separate deformation. Of course, small controlled motions generate fewer subdivisions and are, thus, more interactive.
- *Gradient Controls (τ_u, τ_ℓ, c):* Here, the performance costs of the gradient toggle are negligible, as long as intermediate calculations are cached for reuse during actual deformation. Gradient swathes in-

crease the number of foldover steps ($n = 16$ in figure 4[left] for $\tau_u - \tau_\ell = 1.0$ and $c = 0.05$).

VII. Conclusions

This paper has introduced warp sculpting, a modeling technique that facilitates spatial deformation through an underlying distance field. Deformations are instigated by the rigid body transformation or uniform scaling of complex volumetric tools. The influence of a tool tapers to a user adjustable distance from the tool surface. The effect is reminiscent of physical sculpting where a variety of tools are used to imprint, mold and reshape clay.

Warp sculpting is smooth (C^1 continuous) by construction and foldover-free because large tool transformations are split up into small steps that are guaranteed to be homeomorphic. The tool shape is kept static so that the scalar distance field can be pre-computed. This enables interactive deformation of significantly-sized objects ($>$ 10Hz for up to 82000 vertices). The scale of sculpting can be expanded or contracted by resizing the tool or adjusting its radius of influence. Tools can be combined in a single deformation that accurately reflects multiple shape movement. Global deformations alter but do not erase local details. Finally, the usability of warp sculpting is improved by introducing a gradient toggle that combines the direction of tool transformation with the radiating orientation of the tool's surface to prevent the object unrealistically clinging to the tool.

There are several directions in which warp sculpting can fruitfully be extended, particularly with regard to user-interface issues. A numerical inverse would allow a memory-saving "undo" operation and also support ray-tracing. We currently control tool transformations with a 2-D mouse. A 3-D tracker or force-feedback device would be preferable. Constructing tools with multiple components requires considerable trial and error and some automated design guidelines could help here. In short, warp sculpting would benefit from a cycle of usability evaluation and interface refinement.

Like other spatial deformation methods, warp sculpting is well suited to animation, particularly in approximating the interaction of hard with soft objects. Figure 3 demonstrates this potential with a pawprint imprinted into sand.Warp sculpting is ideal for interactive applications such as computer games and virtual reality, where conventional physics simulations are too costly.

Appendix I
Conditions for Homeomorphism

A spatial deformation function $\mathcal{D}$ with associated Jacobian matrix $\mathbf{J}$ is a homeomorphic mapping and thus foldover free *iff* [22]: $\mathcal{D}$ has continuous first partial derivatives and $\det(\mathbf{J}) > 0$. The first condition holds because the decay function (w), distance reconstruction (ϕ) and transformations are all C^1. All that remains is to find a lower bound on $\det(\mathbf{J})$ for single and mutliple tool rigid transformation and scaling.

Operating in the 3D Geometric Algebra [3] greatly simplifies these proofs. We note that the Jacobian ($\det(\mathbf{J})$) is equivalent to the magnitude of the derivative 3-blade $\|\frac{\partial \mathcal{D}}{\partial \mathbf{p}_1} \wedge \frac{\partial \mathcal{D}}{\partial \mathbf{p}_2} \wedge \frac{\partial \mathcal{D}}{\partial \mathbf{p}_3}\|$. Subsequent appendices summarize the derivation of bounds for single and multiple tools.

Appendix II
Bounds for Single Tools

Proof: In geometric algebra the rigid transformation of a single tool takes the form:

$$\mathcal{D}(\mathbf{p}) = \mathbf{R}(\mathbf{p} - \mathbf{o}_R)\mathbf{R}^{-1} + w\mathbf{t} + \mathbf{o}_R$$

where $R = \exp(-\frac{1}{2} r w \mathbf{A})$ is the rotor in the bivector plane of rotation $\mathbf{A} = \mathbf{I}_3 \mathbf{a}$. The partial derivative are:

$$\begin{aligned}
\frac{\partial \mathcal{D}}{\partial \mathbf{p}_b} &= \mathcal{R}_b + \mathcal{E}_b + \frac{\partial w}{\partial \mathbf{p}_b}\mathbf{t}, \text{where} \\
\mathcal{R}_b &= \left[\mathbf{R}(\mathbf{p} - \mathbf{o}_R)\mathbf{R}^{-1}\right] \rfloor \left[r \frac{\partial w}{\partial \mathbf{p}_b} \mathbf{A} \right] \qquad (9) \\
\mathcal{E}_b &= \mathbf{R}\mathbf{e}_b\mathbf{R}^{-1} \qquad (10)
\end{aligned}$$

The 2-blades with duplicate $\mathcal{R}$ and $\frac{\partial w}{\partial \mathbf{p}_b}\mathbf{t}$ vector components vanish due to linear dependence: $\mathcal{R}_i \wedge \mathcal{R}_j = 0$ and $\frac{\partial w}{\partial \mathbf{p}_i}\mathbf{t} \wedge \frac{\partial w}{\partial \mathbf{p}_j}\mathbf{t} = 0 \;\; \forall i, j \in \{1, 2, 3\}$. Mixed $\mathcal{R}$ and $\frac{\partial w}{\partial \mathbf{p}_b}\mathbf{t}$ 2-blades cancel using the antisymmetry of the vector outerproduct: $\mathcal{R}_i \wedge \frac{\partial w}{\partial \mathbf{p}_j}\mathbf{t} + \frac{\partial w}{\partial \mathbf{p}_i}\mathbf{t} \wedge \mathcal{R}_j = \frac{\partial w}{\partial \mathbf{p}_i}\frac{\partial w}{\partial \mathbf{p}_j}(\mathbf{R}(\mathbf{p} - \mathbf{o}_R)\mathbf{R}^{-1} \rfloor r\mathbf{A} \wedge \mathbf{t}) + \frac{\partial w}{\partial \mathbf{p}_i}\frac{\partial w}{\partial \mathbf{p}_j}(\mathbf{t} \wedge \mathbf{R}(\mathbf{p} - \mathbf{o}_R)\mathbf{R}^{-1} \rfloor r\mathbf{A}) = 0$.

The derivative 3-blade $\mathbf{B} = \frac{\partial \mathcal{D}}{\partial \mathbf{p}_1} \wedge \frac{\partial \mathcal{D}}{\partial \mathbf{p}_2} \wedge \frac{\partial \mathcal{D}}{\partial \mathbf{p}_3}$ thus reduces to:

$$\begin{aligned}
&\mathcal{R}_1 \wedge \mathcal{E}_2 \wedge \mathcal{R}_3 + \mathcal{E}_1 \wedge \mathcal{R}_2 \wedge \mathcal{E}_3 + \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{R}_3 + \tfrac{\partial w}{\partial \mathbf{p}_1}\mathbf{t} \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \\
&\quad + \mathcal{E}_1 \wedge \tfrac{\partial w}{\partial \mathbf{p}_2}\mathbf{t} \wedge \mathcal{E}_3 + \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \tfrac{\partial w}{\partial \mathbf{p}_3}\mathbf{t} + \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \\
&= r\tfrac{\partial w}{\partial \mathbf{p}_1}(\mathbf{p} - \mathbf{o}_R \rfloor \mathbf{A})_1 \mathbf{I}_3 + r\tfrac{\partial w}{\partial \mathbf{p}_2}(\mathbf{p} - \mathbf{o}_R \rfloor \mathbf{A})_2 \mathbf{I}_3 \\
&\quad + r\tfrac{\partial w}{\partial \mathbf{p}_3}(\mathbf{p} - \mathbf{o}_R \rfloor \mathbf{A})_3 \mathbf{I}_3 + \tfrac{\partial w}{\partial \mathbf{p}_1}\mathbf{t}_1 \mathbf{I}_3 + \tfrac{\partial w}{\partial \mathbf{p}_2}\mathbf{t}_2 \mathbf{I}_3 + \tfrac{\partial w}{\partial \mathbf{p}_3}\mathbf{t}_3 \mathbf{I}_3 + \mathbf{I}_3
\end{aligned}$$

because rotation is distributive over the outerproduct and does not alter blade magnitude.

$$\begin{aligned}
\det(\mathbf{J}) &= \|\tfrac{\partial \mathcal{D}}{\partial \mathbf{p}_1} \wedge \tfrac{\partial \mathcal{D}}{\partial \mathbf{p}_2} \wedge \tfrac{\partial \mathcal{D}}{\partial \mathbf{p}_3}\| \\
&= r\tfrac{\partial w}{\partial \mathbf{p}} \cdot (\mathbf{p} - \mathbf{o}_R) \rfloor \mathbf{A} + \tfrac{\partial w}{\partial \mathbf{p}} \cdot \mathbf{t} + 1
\end{aligned}$$

To guarantee homeomorphism requires:

$$\begin{aligned}
&\min(\det(\mathbf{J})) > 0 \\
\Rightarrow\; & r \;\max\|\tfrac{\partial w}{\partial \mathbf{p}}\| \max\|\mathbf{p} - \mathbf{o}_R\| + \max\|\tfrac{\partial w}{\partial \mathbf{p}}\| \;\|\mathbf{t}\| < 1 \\
\Rightarrow\; & \frac{8}{\sqrt{27}}\frac{1}{e}(r\alpha_R + \|\mathbf{t}\|) < 1
\end{aligned}$$

[3] We adopt the notation of Dorst and Mann [34].

Establishing bounds for scaling of a single tool follows a similar path to the proof for rigid body transforms:

$$\begin{aligned}\tfrac{\partial \mathcal{D}}{\partial \mathbf{p}_b} &= \mathcal{S}_b + \mathcal{E}_b, \text{where}\\ \mathcal{S}_b &= \frac{\partial w}{\partial \mathbf{p}_b}(s-1)(\mathbf{p}-\mathbf{o}_S)\\ \mathcal{E}_b &= [w(s-1)+1]\,\mathbf{e}_b\end{aligned}$$

Due to linear dependence $\mathcal{S}_i \wedge \mathcal{S}_j = 0$.

$$\begin{aligned}\mathbf{B} &= \mathcal{S}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 + \mathcal{E}_1 \wedge \mathcal{S}_2 \wedge \mathcal{E}_3\\ &\quad + \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{S}_3 + \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3\\ \det(\mathbf{J}) &= [w(s-1)+1]^2\,[w(s-1)+1\\ &\quad + (s-1)\tfrac{\partial w}{\partial \mathbf{p}}\cdot(\mathbf{p}-\mathbf{o}_S)\Big]\\ \min(\det(\mathbf{J})) &> 0\end{aligned}$$

$s \geq 1$:

$$\begin{aligned}&-(s-1)\left[\min(w) - \max\|\tfrac{\partial w}{\partial \mathbf{p}}\|\max\|\mathbf{p}-\mathbf{o}_S\|\right] < 1\\ \Rightarrow\ &(s-1)\frac{8}{\sqrt{27}}\frac{\alpha_S}{e} < 1\end{aligned}$$

$s < 1$:

$$\begin{aligned}&(1-s)\left[\max(w) + \max\|\tfrac{\partial w}{\partial \mathbf{p}}\|\max\|\mathbf{p}-\mathbf{o}_S\|\right] < 1\\ \Rightarrow\ &(1-s)\left[1 + \frac{8}{\sqrt{27}}\frac{\alpha_S}{e}\right] < 1\end{aligned}$$

■

APPENDIX III
BOUNDS FOR MULTIPLE TOOLS

Proof: In finding a bound for the homeomorphism of multiple tools, an alternative form of (4) is appropriate:

$$\begin{aligned}\mathcal{A} &= \mathbf{p} + \left[\textstyle\sum_j(\mathcal{D}_j(\mathbf{p})-\mathbf{p})w_j^k\right]/\textstyle\sum_j w_j^k\\ \tfrac{\partial \mathcal{A}}{\partial \mathbf{p}_b} &= \textstyle\sum_j \beta_j\left(\mathcal{R}_{j_b} + \frac{\partial w_j}{\partial \mathbf{p}_b}\mathbf{t}_j\right) + \sum_j \beta_j \mathcal{E}_{j_b} + \sum_j \gamma_{j_b}(\mathcal{D}(\mathbf{p})-\mathbf{p})\\ \beta_j &= \frac{w_j^k}{\sum_j w_j^k}\\ \gamma_{j_b} &= \tfrac{1}{\sum_j w_j^k}k\,w_j^{k-1}\tfrac{\partial w_j}{\partial \mathbf{p}_b} - \tfrac{1}{(\sum_j w_j^k)^2}\left[\textstyle\sum_j k w_j^{k-1}\frac{\partial w_j}{\partial \mathbf{p}_b}\right]w_j^k\end{aligned}$$

Here $\mathcal{R}_{j_b}$ and $\mathcal{E}_{j_b}$ are tool specific versions of (9) and (10) from appendix II. Since the weighted combination $\sum_j \beta_j \mathcal{E}_{j_b}$ maintains orthogonality and normalization of the basis vectors:

$$\|\left(\textstyle\sum_j\beta_j\mathcal{E}_{j_1}\right)\wedge\left(\sum_j\beta_j\mathcal{E}_{j_2}\right)\wedge\left(\sum_j\beta_j\mathcal{E}_{j_3}\right)\| = 1$$

This provides an upper bound for the other pseudoscalar components. Note that the volume of a pseudoscalar is at a maximum when the constituent vectors are orthogonal: $\|a \wedge b \wedge c\| \leq \|a\|\ \|b\|\ \|c\|$. A weighted average achieves its maximum when the largest element is the sole contributor. Thus:

$$\begin{aligned}&\max\|\textstyle\sum_j\beta_j\left(\mathcal{R}_{j_b} + \frac{\partial w_j}{\partial \mathbf{p}_b}\mathbf{t}_j\right)\|\\ &= \max_j\left[\max(\tfrac{\partial w_j}{\partial \mathbf{p}_b})(r_j\alpha_{R_j} + \|\mathbf{t}_j\|)\right]\\ &\max\|\textstyle\sum_j\gamma_{j_b}(\mathcal{D}(\mathbf{p})-\mathbf{p})\|\\ &= k\,\max_j\left[\max(\tfrac{\partial w_j}{\partial \mathbf{p}_b})\|\mathcal{D}(\mathbf{p})-\mathbf{p}\|\right]\\ &= k\,\max_j\left[\max(\tfrac{\partial w_j}{\partial \mathbf{p}_b})(\|\mathbf{t}_j\| + \alpha_{R_j}\sqrt{2-2\cos r_j})\right]\\ &\|\textstyle\sum_j\beta_j\mathcal{E}_{j_b}\| = \frac{1}{\sum_j w_j^k}\sum_j w_j^k\|\mathcal{E}_{j_b}\| = 1\end{aligned}$$

Let $c_b = \max_j[\max(\frac{\partial w_j}{\partial \mathbf{p}_b})(\alpha_{R_j}(r_j + k\sqrt{2-2\cos r_j}))$ $+(k+1)\|\mathbf{t}_j\|]$ then:

$$\begin{aligned}\|c\| = \tfrac{8}{\sqrt{27}}\max_j\Big[&\tfrac{1}{e_j}(\alpha_{R_j}(r_j + k\sqrt{2-2\cos r_j}))\\ &+ (k+1)\|\mathbf{t}_j\|\Big]\end{aligned}$$

Finally the condition $\det(\mathbf{J}) > 0$ equates to:

$$\begin{aligned}c_1 + c_2 + c_3 + c_1c_2 + c_2c_3 + c_1c_3 + c_1c_2c_3 &< 1\\ [1\ \ 1\ \ 1]^T\cdot c + [c_1\ \ c_2\ \ c_3]^T\cdot[c_2\ \ c_3\ \ c_1]^T + \max(c_1c_2c_3) &< 1\\ \sqrt{3}\|c\| + \|c\|^2 + \tfrac{1}{\sqrt{27}}\|c\|^3 &< 1\end{aligned}$$

A similar derivation process produces bounds for scaling of multiple tools:

$$\begin{aligned}\|c\| &= \frac{8}{\sqrt{27}}\ \overline{s}\ \max_j\left[\tfrac{1}{e_j}\alpha_{S_j}\right] + \overline{s}\\ \overline{s} &= \begin{cases}\max_j(s_j) - 1 & \text{if } \max_j(s_j) > 2 - \min_j(s_j)\\ 1 - \min_j(s_j) & \text{otherwise}\end{cases}\end{aligned}$$

■

ACKNOWLEDGMENT

This research was funded by the National Research Foundation (NRF) Grant GUN2053416.

REFERENCES

[1] R. Parent, "A system for sculpting 3-d data," *In Computer Graphics (Proceedings of ACM SIGGRAPH 77)*, vol. 11, no. 2, pp. 138–147, 1977.

[2] T. Sederberg and S. Parry, "Free-form deformation of solid geometric models," *In Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, vol. 20, no. 4, pp. 161–171, 1986.

[3] M. Zwicker, M. Pauly, O. Knoll, and M. Gross, "Pointshop 3d: an interactive system for point-based surface editing," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. ACM Press, 2002, pp. 322–329.

[4] F. Dachille, H. Qin, A. Kaufman, and J. El-Sana, "Haptic sculpting of dynamic surfaces," in *1999 Symposium on Interactive 3D Graphics*, ACM. ACM Press / ACM SIGGRAPH, 1999, pp. 103–110.

[5] M. Bro-Nielsen and S. Cotin, "Real-time volumetric deformable models for surgery simulation using finite elements and condensation," *Computer Graphics Forum (Eurographics '96)*, vol. 15, no. 3, pp. 57–66, 1996.

[6] D. James and D. Pai, "Artdefo: Accurate real time deformable objects," in *Proceedings of ACM SIGGRAPH 99*, ser. Computer Graphics Proceedings, Annual Conference Series, ACM. ACM Press / ACM SIGGRAPH, 1999, pp. 65–72.

[7] T. Galyean and J. F. Hughes, "Sculpting: An interactive volumetric modeling technique," *In Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, vol. 25, no. 4, pp. 267–274, 1991.

[8] R. N. Perry and S. F. Frisken, "Kizamu: A system for sculpting digital characters," in *Proceedings of ACM SIGGRAPH 2001*, ser. Computer Graphics Proceedings, Annual Conference Series, ACM. ACM Press / ACM SIGGRAPH, 2001, pp. 47–56.

[9] A. Barr, "Global and local deformations of solid primitives," *In Computer Graphics (Proceedings of ACM SIGGRAPH 84)*, vol. 18, no. 3, pp. 21–30, 1984.

[10] K. Singh and E. Fiume, "Wires: A geometric deformation technique," in *Proceedings of ACM SIGGRAPH 98*, ser. Computer Graphics Proceedings, Annual Conference Series, ACM. ACM Press / ACM SIGGRAPH, 1998, pp. 405–414.

[11] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: A general representation of shape for computer graphics," in *Proceedings of ACM SIGGRAPH 2000*, ser. Computer Graphics Proceedings, Annual Conference Series, ACM. ACM Press / ACM SIGGRAPH, 2000, pp. 249–254.

[12] K. Museth, D. E. Breen, R. T. Whitaker, and A. H. Barr, "Level set surface editing operators," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. ACM Press, 2002, pp. 330–338.

[13] D. Bechmann, "Space deformation models survey," *Computers and Graphics*, vol. 18, no. 4, pp. 571–586, 1994.

[14] T. Milliron, R. J. Jensen, R. Barzel, and A. Finkelstein, "A framework for geometric warps and deformations," *ACM Transactions on Graphics*, vol. 21, no. 1, pp. 20–51, 2002.

[15] R. MacCracken and K. I. Joy, "Free-form deformation with lattices of arbitrary topology," in *Proceedings of ACM SIGGRAPH 96*, ser. Computer Graphics Proceedings, Annual Conference Series, ACM. ACM Press / ACM SIGGRAPH, 1996, pp. 181–188.

[16] W. M. Hsu, J. F. Hughes, and H. Kaufman, "Direct manipulation of free-form deformation," *In Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, vol. 26, no. 2, pp. 177–184, 1992.

[17] P. Borrel and A. Rappoport, "Simple constrained deformation for geometric modeling and interactive design," *ACM Transactions on Graphics*, vol. 13, no. 2, pp. 137–155, 1994.

[18] F. Lazarus, S. Coquillart, and P. Jancéne, "Axial deformations: An intuitive deformation technique," *Computer Aided Design*, vol. 26, no. 8, pp. 607–613, 1994.

[19] I. Llamas, B. Kim, J. Gargus, J. Rossignac, and C. D. Shaw, "Twister: a space-warp operator for the two-handed editing of 3d shapes," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 663–668, 2003.

[20] J. Feng, L. Ma, and Q. Peng, "A new free-form deformation through the control of parametric surfaces," *Computers and Graphics*, vol. 20, no. 4, pp. 531–539, 1996.

[21] P. Decaudin, "Geometric deformation by merging a 3d-object with a simple shape," in *Proceedings Graphics Interface '96*, 1996, pp. 55–60.

[22] J. E. Gain and N. A. Dodgson, "Preventing self-intersection under free-form deformation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 4, pp. 289–298, 2001.

[23] D. Mason and G. Wyvill, "Blendeforming: Ray traceable localized foldover-free space deformation," in *Computer Graphics International 2001 (CGI'01)*, 2001, pp. 183–190.

[24] G. H. Bendels and R. Klein, "Mesh forging: Editing of 3d-meshes using implicitly defined occluders," in *Proceedings of the Eurographics Symposium on Geometry Processing*, Eurographics. The Eurographics Association, 2003, pp. 207–216.

[25] A. Angelidis, G. Wyvill, and M.-P. Cani, "Sweepers: Swept user-defined tools for modeling by deformation," in *Proceedings of Shape Modeling International*, 2004.

[26] M. Alexa, "Linear combination of transformations," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. ACM Press, 2002, pp. 380–387.

[27] L. Barthe, B. Mora, N. Dodgson, and M. Sabin, "Triquadratic reconstruction for interactive modelling of potential fields," in *Proceedings of SMI2002: International Conference on Shape Modeling and Applications*, IEEE. IEEE Press, 2002, pp. 145–153.

[28] G. Wyvill, C. McPheeters, and B. Wyvill, "Animating soft objects," *The Visual Computer*, vol. 2, no. 4, pp. 235–242, 1986.

[29] S. R. Buss and J. P. Fillmore, "Spherical averages and applications to spherical splines and interpolation," *ACM Trans. Graph.*, vol. 20, no. 2, pp. 95–126, 2001.

[30] J. E. Gain and N. A. Dodgson, "Adaptive refinement and decimation under free-form deformation," in *Eurographics UK '99*, Eurographics. Eurographics, 1999.

[31] H. K. Pedersen, "Decorating implicit surfaces," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM Press, 1995, pp. 291–300.

[32] P. Hanrahan and P. Haeberli, "Direct wysiwyg painting and texturing on 3d shapes," in *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. ACM Press, 1990, pp. 215–223.

[33] R. Carling, "Matrix inversion," in *Graphics Gems*, A. Glassner, Ed. Boston: Academic Press, 1990, pp. 470–471.

[34] L. Dorst and S. Mann, "Geometric algebra: A computational framework for geometrical applications (part 1)," *IEEE Computer Graphics and Applications*, vol. 22, no. 3, pp. 24–31, 2002.

James Gain received his BSc(Hons) and MSc degrees in Computer Science from Rhodes University, South Africa, in 1994 and 1996 respectively. In 2000 he obtained his PhD entitled "Enhancing Spatial Deformation for Virtual Sculpting" from the University of Cambridge. He is currently a lecturer in the Computer Science Department at the University of Cape Town, South Africa, and a member of its Collaborative Visual Computing Laboratory.

Patrick Marais received his BSc(Hons) and MSc degrees in Computer Science from the University of Cape Town, South Africa, in 1991 and 1994 respectively. In 1998 he obtained his DPhil entitled "The Segmentation of Sparse MR Images" from the University of Oxford. He is currently a senior lecturer in the Computer Science Department at the University of Cape Town, and a member of its Collaborative Visual Computing Laboratory.