

# Boosted Surfaces: Synthesis of Meshes using Point Pair Generators as Curvature Operators in the 3D Conformal Model

Pablo Colapinto

**Abstract.** This paper introduces a new technique for the formulation of parametric surfaces. Applying translation operations to tangent vectors  $n_o v$  results in null point pairs  $\tau$ . We treat these null point pairs as surface and mesh curvature control points which can be interpolated and exponentiated to construct continuous topological transformations  $\mathcal{K}$  of the form  $e^{-\frac{\tau}{2}}$ . Some basic algorithms are proposed, including the boost which bends a line to a circle of curvature  $\kappa$ , and the twisted boost which generates the Hopf fibration. We investigate methods to control curvature in two orthogonal directions  $u$  and  $v$  and examine a few distance-based and linear weighting techniques for synthesizing surface patches using multiple curvature control points. We consider the expressivity of the technique in manipulating meshes, and find that applying these rotors to mesh points provides a novel and computationally efficient method for creating *boosted forms*.

**Keywords.** Conformal geometric algebra, computer graphics, parametric surfaces, point pair, special conformal transformation, surface topology, Hopf fibration.

## 1. Background: Conformal Geometric Algebra

5-dimensional conformal geometric algebra [CGA] is a **compact** and **expressive** representation of 3-dimensional Euclidean space and its admitted transformations. Initially developed by David Hestenes and later outlined by Li, Hestenes, and Rockwood in [6], CGA maps Euclidean vectors in  $\mathbb{R}^3$  into null vectors in  $\mathbb{R}^{4,1}$  through stereographic projection, providing a basis blade to

---

This work is funded in part by the Deutsch Foundation via the AlloSphere Research Group at UCSB.

represent the point at infinity. All Euclidean transformations such as translations, rotations, dilations, and twists, as well as non-Euclidean boosts (sometimes called *transversors* or *special conformal transformations*) can be generated through exponentiation of the various bivector elements in the algebra. A good introduction to these transforming *rotors* (i.e. *spinors*) embedded in the model can be found in [2]. Alternative non-Euclidean geometries admitted by the model have been explicitly investigated by physicists such as Anthony Lasenby (for instance in [5]).

For immediate reference, table 1 provides an overview of the basic elements of the conformal model. Readers seeking a more thorough introduction to the conformal model are encouraged to explore the references.

## 2. Goal of the Present Work

In the fields of computer graphics, computer vision, and robotics, much attention has been paid to CGA's encapsulation of rigid body movements using **dual lines** to generate **twists**, for example in [3]. Sometimes referred to as *motors* or *screws*, twists concatenate rotation and translation operations into one transformation isomorphic to  $SE(3)$ , which lends it many uses in kinematics. In [7], Wareham, Cameron, and Lasenby demonstrate that twists generated by linear interpolation of dual lines can be used to deform meshes in a direct and intuitive way.

Less-explored are applications of the more complex and less intuitive transformation generator, the **point pair**, which can be exponentiated to generate **boosts**. Whereas the twist deformations directly manipulate local surface *normals*, the boost deformations described here manipulate global surface *curvature*. Building geometric intuition about these curvature operations is the central goal of this work. While a complete framework of point pairs as topological operators is not provided here, we suspect that consideration of various forms and formulations of *boosting rotors* will encourage further study in applying these methods to specific design and engineering problems.

Researchers interested in constructing a full framework should consult [4], in which Dorst and Valkenburg provide a rigorous mathematical treatment of logarithms and exponentials of point pairs and demonstrate that orthogonal and commuting point pairs can generate non-trivial trajectories and "orbits", including torus knots.

The figures in this paper were created using **Versor**, the author's implementation of conformal geometric algebra for graphics synthesis [1].

## 3. Introduction: Tangent Vectors

The simple **boost** rotors  $\mathcal{K}$  in this paper are also called **transversions**. These are **special conformal** operators which can straighten round geometric elements and bend straight ones, while maintaining invariance of local angles.

Symbol	Geometric State	Grade	Algebraic Form
$\alpha$	Scalar	0	$\alpha$
	Vector	1	$\mathbf{a} = \alpha e_1 + \beta e_2 + \gamma e_3$
	Bivector	2	$\mathbf{B} = \mathbf{a} \wedge \mathbf{b}$
	Trivector	3	$\mathbf{I}_3 = \mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c}$
	Point	1	$p = n_o + \mathbf{a} + \frac{1}{2}\mathbf{a}^2 n_\infty$
	Point Pair	2	$\tau = p_a \wedge p_b$
	Circle	3	$\kappa = p_a \wedge p_b \wedge p_c$
	Sphere	4	$\Sigma = p_a \wedge p_b \wedge p_c \wedge p_d$
	Flat Point	2	$\Phi = p \wedge n_\infty$
	Line	3	$\Lambda = p_a \wedge p_b \wedge n_\infty$
	Dual Line	2	$\lambda = \mathbf{B} + dn_\infty$
	Plane	4	$\Pi = p_a \wedge p_b \wedge p_c \wedge n_\infty$
	Dual Plane	1	$\pi = \mathbf{n} + \delta n_\infty$
	Minkowski Plane	2	$E = n_o \wedge n_\infty$
	Direction Vector	2	$\mathbf{t}n_\infty$
	Direction Bivector	3	$\mathbf{B}n_\infty$
	Direction Trivector	4	$\mathbf{I}_3 n_\infty$
	Tangent Vector	2	$n_o \mathbf{t}$
	Tangent Bivector	3	$n_o \mathbf{B}$
	Tangent Trivector	4	$n_o \mathbf{I}_3$
	Rotor	0, 2	$\mathcal{R} = e^{-\frac{\theta}{2}\mathbf{B}} = \cos\frac{\theta}{2} - \sin\frac{\theta}{2}\mathbf{B}$
	Translator	0, 2	$\mathcal{T} = e^{-\frac{d}{2}n_\infty} = 1 - \frac{d}{2}n_\infty$
	Twist	0, 2, 4	$\mathcal{M} = e^{\mathbf{B} + dn_\infty}$
	Dilator	0, 2	$\mathcal{D} = e^{\frac{\lambda}{2}E} = \cosh\frac{\lambda}{2} + \sinh\frac{\lambda}{2}E$
	Boost	0, 2	$\mathcal{K} = e^{-n_o \mathbf{t}} = \cosh(n_o \mathbf{t}) - \sinh(n_o \mathbf{t})$

TABLE 1. Basic elements of 3D conformal geometric algebra and their algebraic constructions as outlined in [2]. Subspaces of the model serve as direct and dual representations of geometric entities such as circles and lines and planes. Bold symbols represent Euclidean elements, with lowercase letters representing 1-blade vectors as is the custom. With  $e_-^2 = -1$  and  $e_+^2 = 1$ , then  $n_o = (e_- + e_+)/2$  and  $n_\infty = (e_- - e_+)$  represent the point at the origin and the point at infinity, respectively.

A component of **inversive geometry**, boosts can be considered a double reflection in two spheres with a common point, much the same way a rotation can

be considered a double reflection in two planes with a common line. Operating at the origin, boosts can also be constructed through concatenation of a sequence of operations: inversion in the unit sphere, followed by a translation, followed by another inversion in the unit sphere. They can be used to realize the *Lorentz group of transformations* and thus to model the symmetries of relativistic physics.<sup>1</sup>

In the conformal model of  $\mathbb{R}^{4,1}$  boost rotors  $\mathcal{K}$  are generated through exponentiation of a **tangent vector**  $t$ . As described by Dorst, Fontijne, and Mann in [2],  $t$  is a tangent vector formed by wedging the origin blade with a Euclidean vector  $\mathbf{v}$ :

$$t = n_o \wedge \mathbf{v} = n_o \mathbf{v} \quad (1)$$

and the exponentiation of this **bivector generator** is expanded to

$$\mathcal{K} = e^{n_o \mathbf{v}} = 1 + n_o \mathbf{v}. \quad (2)$$

The resulting boost (“transversor”)  $\mathcal{K}$  is applied to other elements  $x$  of the geometric algebra using the normal “sandwich” product:

$$x' = \mathcal{K}x\mathcal{K}^{-1} \quad (3)$$

where  $x$  is a geometric element such as a point, circle, line, etc or operator and  $\mathcal{K}^{-1}$  is the *inverse* of  $\mathcal{K}$ :

$$\mathcal{K}^{-1} = \tilde{\mathcal{K}}/(\mathcal{K}\tilde{\mathcal{K}}) \quad (4)$$

which is itself defined using the *reverse*:

$$\tilde{\mathcal{K}} = 1 - n_o \mathbf{v}. \quad (5)$$

The sandwich product is sometimes written as

$$x' = \mathcal{K}[x] \quad (6)$$

when convenient.

In the case that  $x$  is a circle, figure 1 demonstrates what happens as we increase the length of a coplanar tangent vector  $t$ .

In the 5D conformal model of 3D space such transformations are not limited to the 2D plane. Figure 2 shows the result of transforming along a tangent *orthogonal* to the plane of the circle.

---

<sup>1</sup>To support our concept of a curvature operation in homogenous coordinates, this paper uses the notion of a *boost* more liberally than might be found in a physics paper, where *pure boosts* are considered separately from translations and rotations or dilations. Strictly speaking, these *boosted forms* may be more precisely described as *special conformal transformation forms*, and *boosting spinors* likewise *SCT spinors*. However the verb *boost* more accurately describes the mesh modelling technique that is employed, which relates to the active techniques of *lofting* or *skinning* a surface.

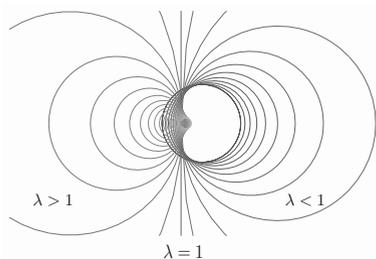


FIGURE 1. Transformations of the form  $\sigma' = \mathcal{K}\sigma\mathcal{K}^{-1}$  with  $\sigma$  representing a unit circle at the origin on the  $e_{12}$  ( $xy$ ) plane and  $\mathcal{K} = e^{\lambda n_o \wedge e_1}$  a boost at the origin applied in the  $e_1$  ( $x$ ) direction for a range of  $\lambda$ . The circle transforms into a line when  $\lambda = 1$ . As  $\lambda$  increases past 1, the unit circle has turned inside-out and reverses orientation. It converges on its own center point as  $\lambda$  approaches infinity.

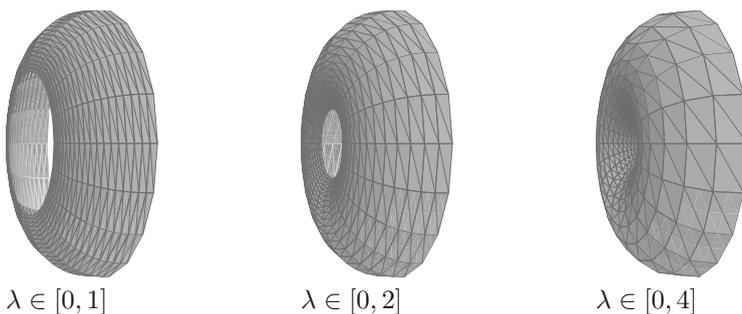


FIGURE 2.  $\sigma' = \mathcal{K}\sigma\mathcal{K}^{-1}$  with  $\sigma$  representing a unit circle at the origin on the  $e_{23}$  ( $yz$ ) plane and  $\mathcal{K} = e^{\lambda n_o \wedge e_1}$  a boost in the  $e_1$  ( $x$ ) direction orthogonal to the circle (i.e. normal to the plane of the circle). Surfaces are created by boosting  $\sigma$  across a range of  $\lambda$  values. From left to right, the range of  $\lambda$  values increases in each of the three surfaces.

### 4. Null Point Pairs

As described by Dorst et al in [2], there is an intimate relationship between tangent vectors and point pairs. Translating a tangent vector away from the origin gives us a point pair with zero radius. We can homogenize the boost transformation in 3D space by translating the tangent vector generator *before* exponentiating it. Figure 3 depicts a negatively curved surface made in this way.

Using translating rotors of the form  $\mathcal{T} = e^{-\frac{\mathbf{v}n_\infty}{2}} = 1 - \frac{\mathbf{v}n_\infty}{2}$  we transform tangent vectors using the sandwich product:

$$\tau = \mathcal{T}t\mathcal{T}^{-1} \tag{7}$$

where the result  $\tau$  is a point pair of zero radius and  $t$  is a tangent vector  $n_o\mathbf{v}$ . Though null, the point pair will still have a variable *weight* associated with it that is determined by the length of the original tangent vector.

The formulation of a *homogenous boost* is therefore

$$\mathcal{K} = e^{\mathcal{T}t\mathcal{T}^{-1}} = e^\tau = 1 + \tau \tag{8}$$

which has been expanded using Dorst and Valkenburg’s rules for the exponentiation of point pairs:

$$e^{-\tau} = \cosh(\tau) - \sinh(\tau) \tag{9}$$

where  $\cosh(\tau) = \cosh(|\tau|)$  is a scalar and

$$\sinh(\tau) = \begin{cases} \frac{\sinh(|\tau|)}{|\tau|}\tau & \text{if } \tau^2 \neq 0 \\ \tau & \text{if } \tau^2 = 0 \end{cases} \tag{10}$$

using the  $l^2$ -norm  $|\tau|$ .

Translated tangents create null point pairs where  $\tau^2 = 0$ . When we interpolate these null point pairs later, the resulting point pairs will usually not be null and we will resort to the definition of  $\sinh(\tau)$  when  $\tau^2 \neq 0$ .

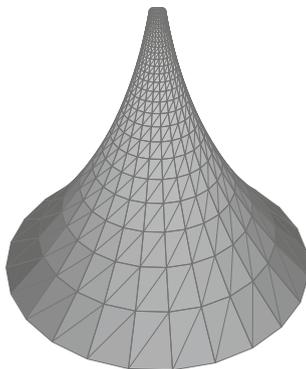


FIGURE 3. A horn formed by boosting a base circle  $C$  along its axis  $\lambda = -n_\infty]C$  by a boost  $\mathcal{K}$ , where  $\mathcal{K} = 1 + k\tau, k \in [0, 1]$ . To generate the boosting spinor  $1 + k\tau$ , we translate the tangent orthogonal to a circle to a point along the axis  $\lambda$  and exponentiate using equations 9 and 10. To generate the mesh, we repeat the transformations across a range of  $k$  (here the range is  $[0, 1]$ ).

### 4.1. A Boost with a Twist: The Hopf Fibration

The canonical *Hopf fibration* can be constructed by a **boost** followed by a **twist**. The fibration maps the 2-sphere to the 3-sphere by taking each point on the 2-sphere to a circle *fiber*, thereby expanding each point by one dimension. If we map the point at the south pole to a circle, then the antipodal point at the north pole is mapped to the axis of that circle. Thus we start by noting that the fibration of a longitudinal line from pole to pole can be generated by a rotor which takes a circle to its own axis. To construct this transformation, we compose a boost which straightens the circle into a line, and a twist which screws that line into the axis. The combined process of boosting and twisting is seen in figure 4d.

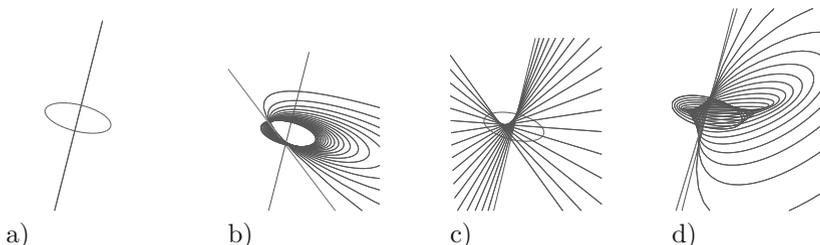


FIGURE 4. a) The initial components of the Hopf fibration are a circle and its axis representing fibers over the south and north poles of the 2-sphere. b) The circle is boosted into a line. c) The line is twisted into the axis. d) Composition of the boost and twist make a fiber bundle over a meridian of the 2-sphere.

Explicitly, to map the point  $p_{\theta\phi}$  with spherical coordinates  $\theta, \phi$  (in radians) on a 2-sphere to a circle fiber of a 3-sphere, we start with a unit circle  $C$  at the south pole (where  $\phi = -\frac{\pi}{2}$ ) and a base axis  $\Lambda = -n_{\infty} \lrcorner \kappa$  at the north pole (where  $\phi = \frac{\pi}{2}$ ). We then define the transversion or boost  $\mathcal{K}_{\theta,\phi}$  which takes  $C$  to a line using equation 2:

$$\mathcal{K}_{\theta,\phi} = 1 + k_{\phi} n_{\circ} \mathbf{v}_{\theta} \tag{11}$$

where  $k_{\phi} = \frac{1}{2} + \frac{\phi}{\pi}$  and scales in the range  $[0, 1]$  and where  $\mathbf{v}_{\theta}$  is a unit vector in the plane of the circle. At  $\phi = \frac{\pi}{2}$ ,  $\mathcal{K}$  takes the circle to a line:

$$\Lambda_{\theta} = \mathcal{K}_{\theta,\frac{\pi}{2}} [C]. \tag{12}$$

The corresponding twist motor  $\mathcal{M}_{\theta,\phi}$  can then be generated by finding the ratio of the initial axis  $\Lambda$  with the line  $\Lambda_{\theta}$ . For detailed analysis on how to find logarithms of motors see for instance [8].

$$\mathcal{M}_{\theta,\phi} = e^{k_{\phi} \log(\frac{\Lambda}{\Lambda_{\theta}})}. \tag{13}$$

The full transformation rotor  $\mathcal{K}^{\mathcal{H}}$  can then be defined:

$$\mathcal{K}_{\theta,\phi}^{\mathcal{H}} = \mathcal{M}_{\theta,\phi} \mathcal{K}_{\theta,\phi}, \quad \theta \in [0, 2\pi], \quad \phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]. \tag{14}$$

Note that as a product of a twist and a boost,  $\mathcal{K}^H$  is not a simple boost composed of just a scalar and bivector, but now also contains the 4-blades  $e_{12}E$ ,  $e_{13}E$ ,  $e_{23}E$ ,  $e_{123}n_o$ , and  $e_{123}n_\infty$ .

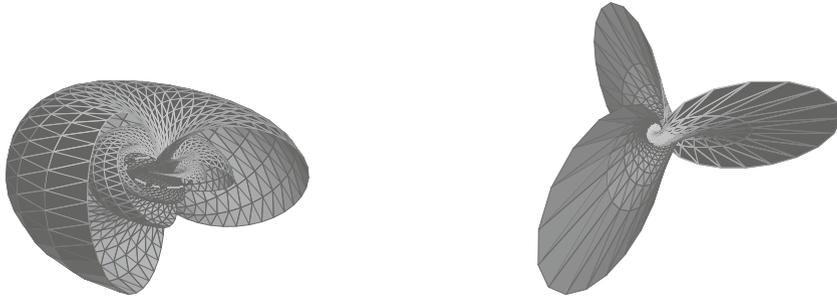


FIGURE 5. *Twistor* meshes representing the 3-sphere Hopf fibration generated by application of the transformation rotor  $\mathcal{K}^{\mathcal{H}}_{\theta,\phi}$  onto an base circle.

The Hopf fibration plays an interesting role in point pair generators: antipodal fibers are orthogonal and commute. This means the we can use them to create the knot orbits described by Dorst and Valkenburg in [4].<sup>2</sup>

### 5. Curvature Control

In subsequent formulations we hope to show the usefulness of treating the boosting rotors as curvature operators. We begin by investigating the use of translated tangents for direct control of curvature at a point  $p$ .

We need to be able to continuously bend a line while fixing one of its points, as depicted in figure 6.

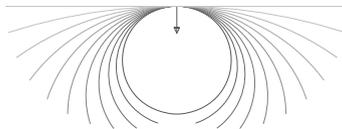


FIGURE 6. A normalized line is bent into a circle. The generator of the transformation is a tangent vector translated to a point on the line.

Given a normalized line  $\Lambda$ , we bend it into a circle of radius  $r$  and curvature  $\kappa = \frac{1}{r}$  at point  $p$  by applying a boosting rotor of the form:

$$\mathcal{K} = 1 - \frac{\kappa\tau}{2} \tag{15}$$

<sup>2</sup>Videos demonstrating this commutation property, which allows us to keep a knot invariant as we smoothly transform the fibers around which it winds, can be viewed online at <https://vimeo.com/wolftype/videos>.

where  $\tau$  is a zero-sized (null) point pair created by first translating a unit tangent vector orthogonal to the line to the point  $p$  on the line, and then weighting it by the target curvature. By rotating the originating tangent vector 180 degrees, we can use the same formula to create a boost that takes a circle with curvature  $\kappa$  and straightens it into a line. The  $p$ -translated and  $\kappa$ -weighted tangent  $\tau$  generates a  $2\kappa$  curvature operator at  $p$ .

**5.1. UV Surface Curvature**

The ability to specify curvature at a point  $p$  with an operator  $\mathcal{K}$  specified in equation 15 suggests that we can build a UV surface patch with a specific curvatures in the neighborhood of  $p$ , one operating in the  $u$  direction and the other in the  $v$  direction.

**5.1.1. Translating tangents along  $u$  and  $v$ .** There are a few ways to build such surfaces with null point pairs. For instance, we can translate two tangents, one along the  $u$  direction and another along the  $v$  direction and then sum them. To each point  $p_{uv}$  on a plane  $\Pi$  we apply a boost

$$\mathcal{K}_{uv} = e^{-(\kappa_u \tau_u + \kappa_v \tau_v)} \tag{16}$$

where  $\tau_u = \mathcal{T}_u[t]$  and  $\tau_v = \mathcal{T}_v[t]$  are the translated tangents and  $\kappa_u, \kappa_v$  are scalar curvature values. Figure 7 demonstrates the results, which allow formation of parabolic, elliptic, umbilic and saddle points.

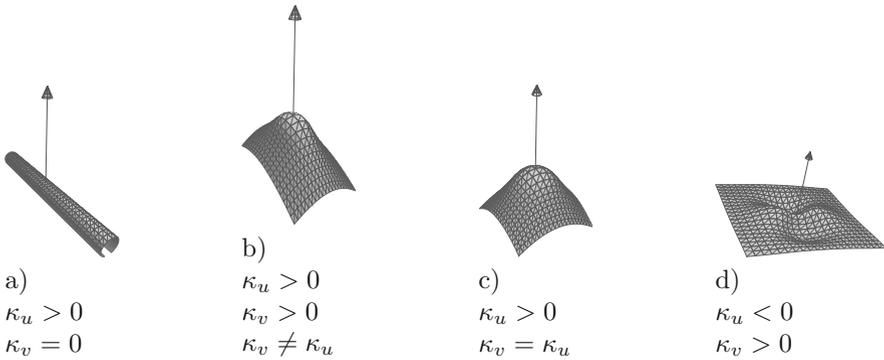


FIGURE 7. Local curvatures defined using equation 16: a) parabolic, b) elliptic, c) umbilic and d) hyperbolic.

**5.1.2. Calculating squared distance to  $u$  and  $v$  osculating circles.** We can also define local curvature based on squared distance to two *osculating* circles at  $p$ .

Consider local curvatures  $\kappa_u$  and  $\kappa_v$  in orthogonal directions  $u$  and  $v$  on a plane surface  $\Pi$ . Using equation 15 we create two circles  $C_u$  and  $C_v$  in the  $u$  and  $v$  directions *orthogonal* to the tangent vector and to each other. We

boost each point  $p_{uv}$  on  $\Pi$  according to the inverse of its squared distance  $d_u^2$  and  $d_v^2$  to these osculating circles.

To calculate this deformation, for each point  $p_{uv}$  on the surface we generate a transforming boost  $\mathcal{K}_{uv}$ ,

$$\mathcal{K}_{uv} = e^{-\left(\frac{\kappa_u}{d_u^2} + \frac{\kappa_v}{d_v^2}\right)\tau} \tag{17}$$

where  $\tau$  is the homogenous tangent  $t$  normal to the plane and squared distances  $d_u^2$  and  $d_v^2$  are between  $p_{uv}$  and the osculating circles  $C_u$  and  $C_v$ . It is particularly convenient to use this distance to the circles themselves, as it creates a natural ‘‘falloff’’ effect.

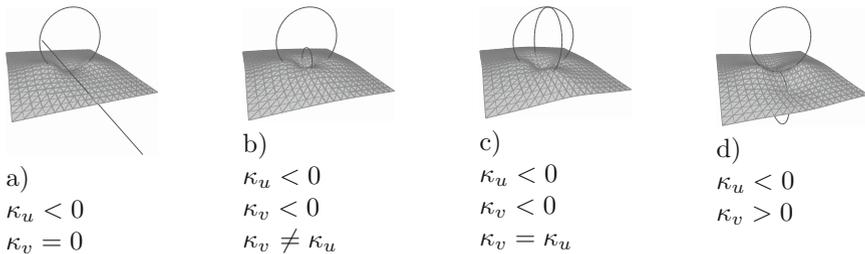


FIGURE 8. Local curvatures defined using equation 17: a) parabolic, b) elliptic, c) umbilic and d) hyperbolic. Distance-based weighting makes it difficult to create purely parabolic deformations, where curvature is exactly zero in one direction.

Squared distances are calculated using the inner product of the point and the dual of the circle.

For  $d_n^2$  :

$$d_n^2 = (p_{uv} \rfloor C_n^*)^2. \tag{18}$$

Note that for points outside the circle, this formulation of  $d^2$  represents a combined measure of both the shortest distance  $d_s^2$  to the perimeter of the circle *and* the curvature of the circle:  $d^2 = (2\kappa + d_s)^2$ .

To transform the point, we first apply the sandwich product and then normalize the result:

$$p'_{uv} = f_n(\mathcal{K}_{uv}[p_{uv}]) \tag{19}$$

where we represent the normalization step as  $f_n$ :

$$f_n(p) : p \mapsto p / (-n_\infty \rfloor p). \tag{20}$$

In an implementation, one can simply divide the result by its  $n_o$  blade to normalize it.

Figure 8 shows the results of such a formulation, which allows us to control curvature in  $u$  and  $v$  directions.

**5.1.3. Generating UV surface patches by adding an additional weighting factor.** Distance-based weighting lends itself to the incorporation of more control points. Integrating four sets of such generating  $u, v$  point pairs allows us to build up surface patches, as demonstrated in figures 9 and 10. For more precise control, the formulation utilizes not only distance from the osculating circles, but also from the location of the null point pair itself:

$$\mathcal{K}_{uv} = e^{-\sum_{i=1}^4 W_i \tau_i} \tag{21}$$

where  $W_i = \frac{1}{d_{p_i}^2} (\frac{\kappa_{ui}}{d_{u_i}^2} + \frac{\kappa_{vi}}{d_{v_i}^2})$  is the weight of the  $i$ th point pair and the new term  $d_{p_i}^2$  is the squared distance from  $p_{uv}$  to the location of the  $i$ th generating point pair. For a point at  $p$  and a point pair at  $x$ :

$$d_p^2 = -2p|x. \tag{22}$$

In an implementation we usually add some small value to  $d_p^2, d_u^2,$  and  $d_v^2$  to prevent dividing by zero, as well as some constant  $k$  by which to multiply  $d_p^2$  to control the “falloff” of the deformation.

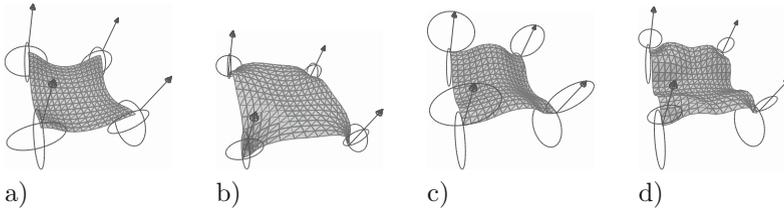


FIGURE 9. Surface patches created through independent control of four  $uv$  curvatures. Arrows represent the translated tangent vectors. Osculating circles are generated in two orthogonal directions using equation 15. Mesh points are deformed according to equation 21 to create a) positive umbilic, b) negative umbilic, c) and d) hyperbolic curvatures.

**5.2. Weighting by Inverse Squared Distance to  $\tau$**

The  $\kappa_u$  and  $\kappa_v$  curvature-controlled meshes above demonstrate our ability to articulate specific curvatures in specific directions at specific locations. In this section we examine the effects of weighting *only* by squared distance to the the point pair. It is a simpler algorithm that does not use orthogonal osculating circles and yet still creates alluring results.

An  $n$ -sized field of point pairs is used to warp a pre-existing mesh. The point pair generator to be applied to a point  $p$  is a sum of  $n$  distance-weighted point pairs:

$$\tau_p = \sum_i^n \frac{1}{d_p^2} \tau_{ix} \tag{23}$$

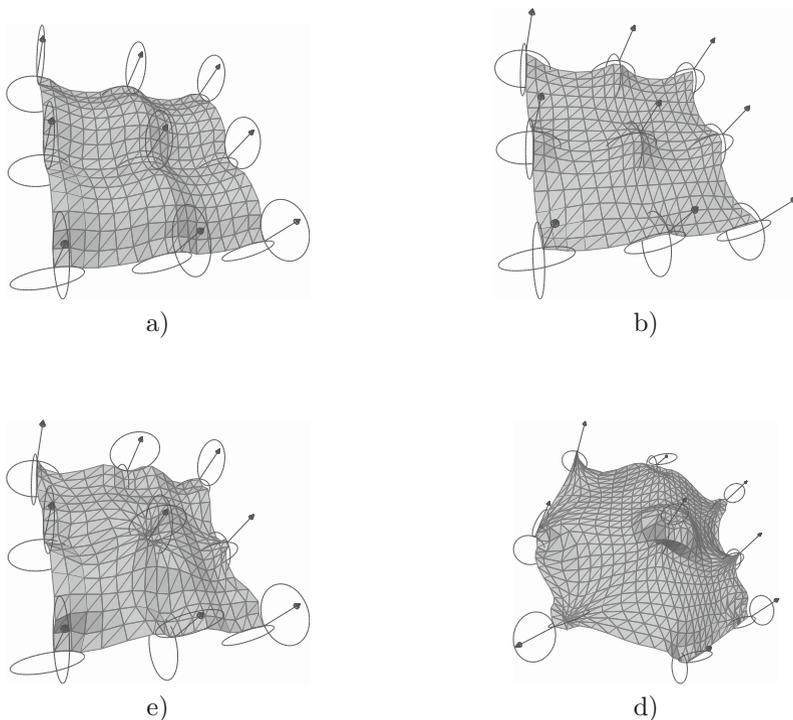


FIGURE 10. Surfaces here are built from addition of multiple patches from figure 9. a) Hyperbolic saddle points, b) umbilic points. In c) we apply an assortment of curvature operators. In d) the tangents point in various directions.

with  $\tau_{i,x}$  a null point pair at  $x$  and the denominator is the squared distance function  $d_p^2 = -2p|x$ . Figure 11 shows various boosted forms created using this method.

## 6. Linear Interpolation

For comparison to the squared distance-based methods outlined above, let us explore the effects of an even simpler technique. Rather than weighting based on squared distances  $d_u^2$  and  $d_v^2$  to two osculating circles or a squared distance  $d_p^2$  to a null point pair location, we explore the effects of linear and bilinear interpolation. We will not distinguish between  $u$  and  $v$  curvatures and instead start with something quite basic: we define a line contour by interpolating across two null point pairs.

Two null point pairs  $\tau_1$  and  $\tau_2$  are linearly interpolated:

$$\tau_t = (1 - t)\tau_1 + t\tau_2, \quad t \in [0, 1]. \quad (24)$$

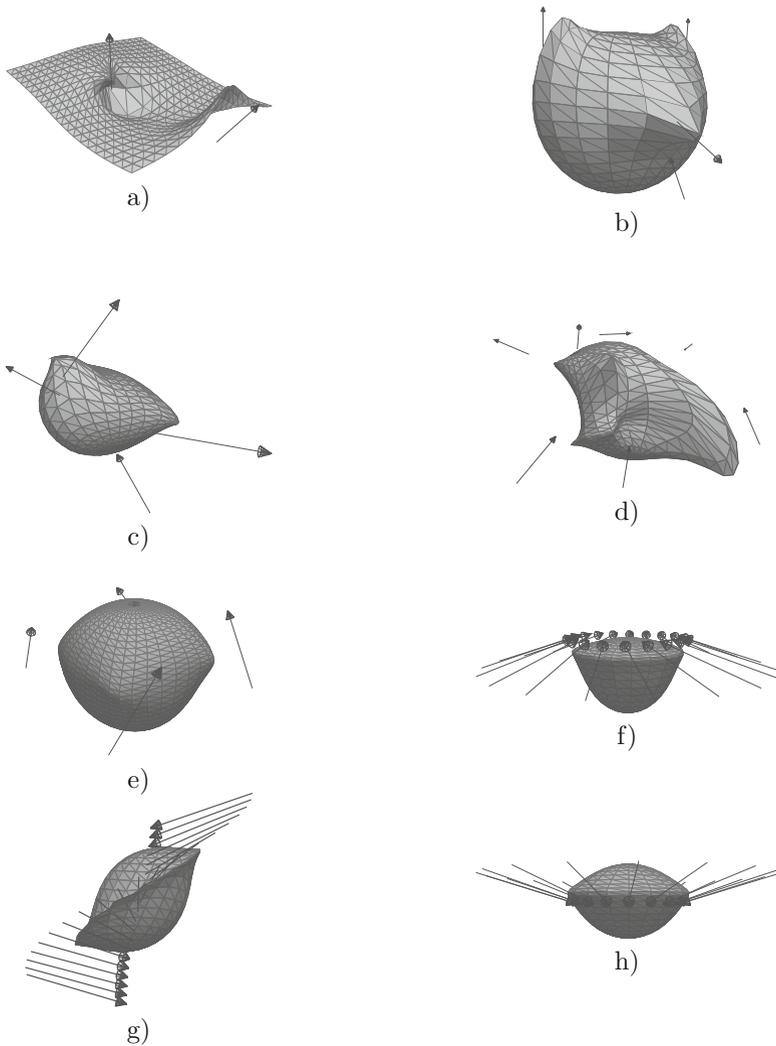


FIGURE 11. Warping of a plane (a) and a sphere (b-h) mesh by integrating multiple point pair generators located near the mesh. Arrows once again represent the null point pair generators. To transform each point, we use equation 23 where each generator is weighted based on the inverse of its squared distance to the original mesh point.

The resulting point pair,  $\tau_t$ , is used to generate a boosting rotor  $\mathcal{K}_t = e^{-\tau_t/2}$ . This rotor is applied to points  $p_t$  on the straight line interval connecting the two point pairs.

We can create the surfaces of figure 12 by using four null point pair generators and creating affine combinations of them using bilinear interpolation.

$$\tau_{st} = (1 - s)((1 - t)\tau_1 + t\tau_2) + s((1 - t\tau_3) + t\tau_4), \quad t \in [0, 1], \quad s \in [0, 1]. \quad (25)$$

A point on a surface is evaluated as in equation 19:

$$p'_{st} = f_n(\mathcal{K}_{st}[p_{st}]) \quad (26)$$

where  $\mathcal{K}_{st} = e^{-\frac{\tau_{st}}{2}}$  is the boosting rotor evaluated at  $s, t$  using bilinear interpolation,  $p_{st}$  is a point on a grid and  $f_n(p)$  is our normalization function from equation 20.

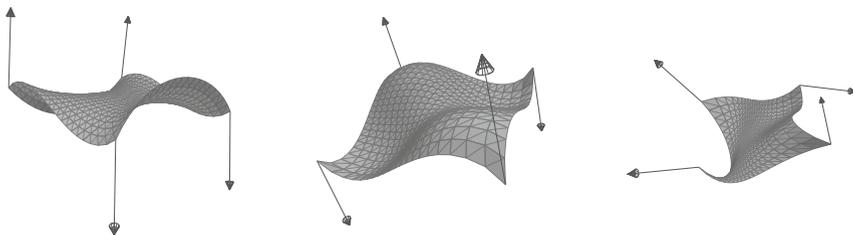


FIGURE 12. Surface patches generated by four boosts. Point pairs are first created by translating tangents to the four corners of a grid. At each vertex of the grid, rotors are built by bilinear interpolation of the corner point pairs. Boosting spinors are then used to warp that point on the grid.

The linear approach to surface generation is a direct extrapolation of the method using *twist fields* described by Wareham, Cameron and Lasenby in [7]. Both are parametric methods offering intriguing alternatives to nurbs-based modelling, however we find that the twist-field method leads to more predictable results. While our results maintain a relationship between the orientation of the generating point pairs and the curvature of the contour, the precise relationship is counterintuitive. Until we can implement an absolute transformation from a tangent vector at the origin that can be interpolated and exponentiated, we prefer to use the more easily interpretable weight-based methods outlined above.

## 7. A Note on Exponentiating the Sum of Point Pair Generators

Individually, each null point pair generates a *dipole* warp field around it, as shown in figure 13. Given two or more point pair generators, we have seen that we can sum these generators before exponentiating to manipulate curvature across a surface. Thus we have been able to define a field of dipole boosting forces.

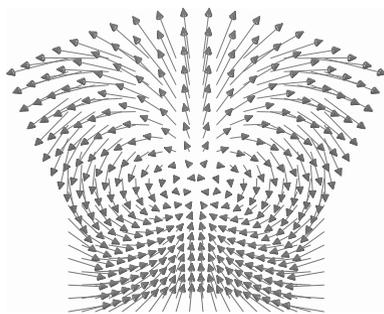


FIGURE 13. A dipole field is created by a boost using equations 27 and 28.

We must take a moment to distinguish, however, the difference between this method of summing up all the point pairs before generating a single transformation, and the typical vector algebra approach to modelling forces, where one generates a transformation for *each* generator and then sums up the displacements. This latter approach is commonly used in modelling electromagnetic fields, for instance. Since we are primarily interested in manipulating *shapes* intuitively and consistently, the first method works well: it is computationally efficient, and provides consistent, controllable results.

To emphasize the difference between these methods, we take a look at each as they act upon a field of points. A boost generated from a single null point pair creates a dipole field around it, visualized in figure 13. At each point  $p_{uv}$  in a vector field, we calculate the Euclidean displacement vector

$$\mathbf{v}_{uv} = \mathbf{p}'_{uv} - \mathbf{p}_{uv} \tag{27}$$

where  $\mathbf{p}'_{uv} = f_n(\mathcal{K}_{uv}[p_{uv}])$  and bold  $\mathbf{p}$  signifies use of the Euclidean vector component and

$$\mathcal{K}_{uv} = e^{-\frac{\tau}{d^2}} \tag{28}$$

with  $d^2$  the squared distance between the translated tangent  $\tau$  and the point  $p_{uv}$ .

Figure 14 demonstrates that when calculating the effect of multiple dipole generating boosts in a field, the two methods are not equivalent. Thus while boosts generate dipole fields individually, their sums do not appear to generate accurate models of physical forces.

It is also important to note that in using the interpolation method of equation 21 we are *not* creating suitable bivector splits as outlined in [4]. Typically in our case:

$$e^{\tau_1 + \tau_2} \neq e^{\tau_1} e^{\tau_2}$$

$$\tau_1 \tau_2 \neq \tau_2 \tau_1.$$

This is because null point pairs *only* commute if they are located at the same point in space, and we are explicitly spreading them across a field. Since we treat  $\tau = \tau_1 + \tau_2$  as an affinely interpolated 2-blade, the exponent is

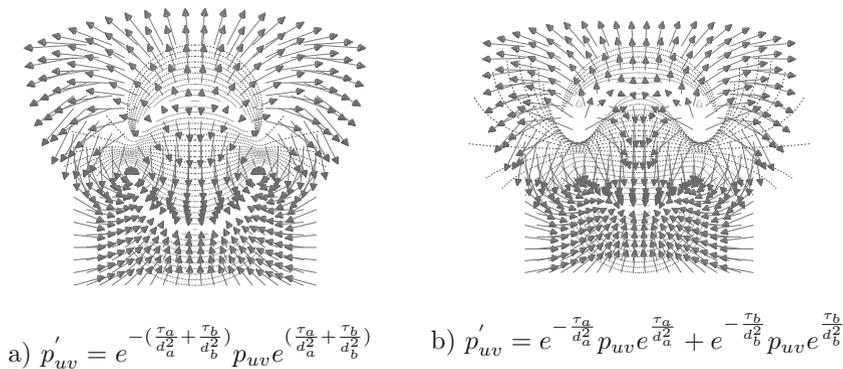


FIGURE 14. In each image, two dipole generating point pairs  $\tau_a$  and  $\tau_b$  act on a field of points  $p$ . We draw a vector field at each point  $p_{uv}$  defined by  $v_{uv} = p'_{uv} - p_{uv}$ . The equipotential (dotted) lines of the vector field are also drawn. Two integration methods are examined. a) Summing generators weighted by their inverse squared distance to each point, and then creating a single rotor displacement. b) Calculating displacements for each generator and then summing the displacements. Method b) is standard in vector field topology. Method a) is the simplified approach used in this paper. The methods are clearly not equivalent, but our implementation is cheaper to compute and remains intuitive to control.

not always null in this case, and therefore equation 8 does not hold, since it only describes null point pair exponentials. We therefore calculate our rotor using equation 9, solving  $\sinh(\tau)$  using the rules for imaginary, real, and null  $\tau$  defined in [4] and repeated here in equation 10.

### 8. Conclusions and Future Work

CGA offers the graphics researcher the opportunity to discover and experiment with powerful methods for the synthesis of forms. In this paper we have explored some synthesis techniques through bending and warping using boosting rotors generated by point pairs. We have seen that such boosting operations can be used to generate the Hopf fibration. We have described specific formulations for transforming curvature at an arbitrary point by generating osculating circles, and began to establish a protocol for parametric design

through curvature control. By interpolating between point pair generators in a field based on various squared distance and linear metrics we were able to propose a novel design technique for the generation of surfaces. While a more rigorous treatment is necessary, the simple fact that we were able to generate expressive forms through simple extrapolation of methods investigated in the references lends credence to the inventive powers of geometric algebra.

It is hoped that such *formal* investigations will provoke more explorations into curvature operations with tangent vectors. For instance, one could use the commutator product of a point with a point pair to establish a curvature differential operator, or develop an analytical technique that decomposes arbitrary forms into a set of weighted basis boosts. The modelling of more complicated topological operations and morphogenetic *catastrophes* should be explored, especially those with biological significance, such as invagination and intussusceptions. A formulation of *implicit* surface design using boosting operators is also desirable. Additionally, we should investigate whether the methods outlined in this paper can be used to directly model actual physical forces; pneumatic pressures or special cases of electromagnetic forces perhaps. Ultimately, such explorations could help establish a new boost-based framework for artistic and scientific modelling.

### Acknowledgements

The author would like to thank the many anonymous reviewers for their invaluable criticism and corrections, as well as the AGACSE2012 organizing committee in La Rochelle, the Media Art and Technology Program at UCSB, and the Deutsch Foundation for its generous support.

### References

- [1] P. Colapinto, *Versor: Spatial Computing with Conformal Geometric Algebra*. Masters Thesis, University of California at Santa Barbara. Available via <http://versor.mat.ucsb.edu> (2011) with video demonstrations available at <http://www.vimeo.com/wolftype/videos>.
- [2] L. Dorst, D. Fontijne, S. Mann, *Geometric Algebra for Computer Science*. Morgan Kaufmann, San Francisco (2007).
- [3] L. Dorst, *The Representation of Rigid Body Motions in the Conformal Model of Geometric Algebra*. In: B. Rosenhahn, R. Klette, D. Metaxas (Eds.), *Human Motion – Understanding, Modelling, Capture, and Animation*, Springer Netherlands (2008), pp. 507–529.
- [4] L. Dorst and R. Valkenburg, *Square Root and Logarithm of Rotors in 3D Conformal Geometric Algebra using Polar Decomposition*. In: L. Dorst and J. Lasenby (Eds.), *Guide to Geometric Algebra in Practice*, Springer London (2011), pp. 25–46.
- [5] A. Lasenby, *Recent Applications of Conformal Geometric Algebra*, In: H. Li, P. J. Olver and G. Sommer (Eds.), *Computer Algebra and Geometric Algebra with Applications*, Lecture Notes in Computer Science Volume 3519, Springer Berlin Heidelberg (2005), pp. 298–328.

- [6] H. Li, D. Hestenes, and A. Rockwood, *Generalized Homogeneous Coordinates for Computational Geometry*. In: G. Sommer (Ed.), *Geometric Computing with Clifford Algebra*, Springer Berlin Heidelberg (2001), pp. 25–58.
- [7] R. Wareham, J. Cameron, A. Lasenby, *Applications of Conformal Geometric Algebra in Computer Vision and Graphics*. In: H. Li, P. J. Olver and G. Sommer (Eds.), *Computer Algebra and Geometric Algebra with Applications*, Lecture Notes in Computer Science Volume 3519, Springer Berlin Heidelberg (2005), pp. 329–349.
- [8] R. Wareham, J. Lasenby, *Rigid Body and Pose Interpolation using Geometric Algebra*. Submitted to *ACM Transactions on Graphics*, (2004).

Pablo Colapinto  
University of California at Santa Barbara  
Media Arts and Technology Program  
USA  
e-mail: [pablocolapinto@gmail.com](mailto:pablocolapinto@gmail.com)

Received: December 28, 2012.

Accepted: September 17, 2013.