Applications of Conformal Geometric Algebra in Mesh Deformation

Mauricio Cele López Belón
Huddle Group S.A.
Buenos Aires, Argentina
mauricio.lopez@huddle.com.ar

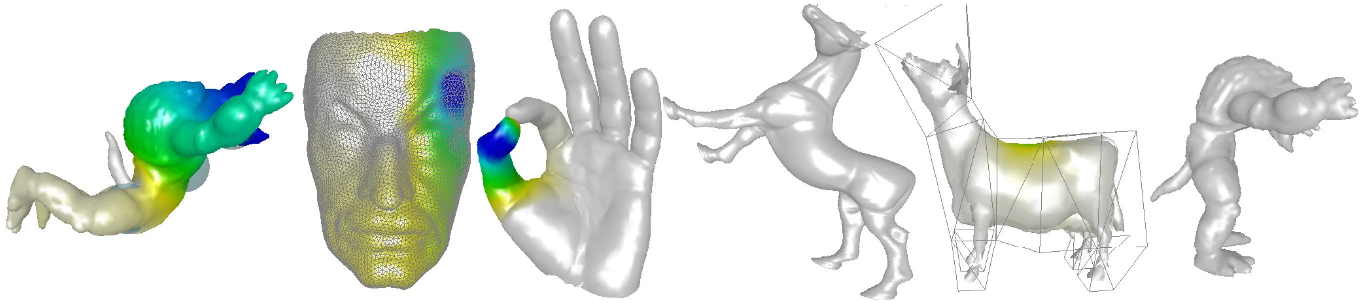


Fig. 1. Deformed shapes obtained using several methods extended by Conformal Geometric Algebra: Spline aligned deformation (left armadillo), generalized barycentric coordinates (max-face and hand), Free-Form Deformation (horse), Harmonic Coordinates (caw) and As-Rigid-As-Possible Surface Modeling (right armadillo)

***Abstract*—We illustrate the suitability of Conformal Geometric Algebra for representing deformable mesh models. State-of-the-art modeling tools allow the user to deform $3D$ models (or region of interest) by selecting sets of points on the surface, called *handles*, and move them freely. The deformed surface should look "naturally" stretched and bent. Mesh representations based on Conformal Geometric Algebra extend, quite naturally, the existing deformable mesh representations by introducing rigid-body-motion handles, a.k.a *motor handles*, instead of just translation handles. We show how these mesh representations conduct to a fast and easy formulation for the Spline-aligned deformation and a formulation for linear surface deformation based on generalized barycentric coordinates. Also, we reformulate the Free-Form Deformation (FFD), Harmonic Coordinates (HC) and As-Rigid-As-Possible (ARAP) Surface Modeling into the Conformal Geometric Algebra framework and discuss the advantages of these reformulations.**

***Keywords*-I.3.5.d Computer Graphics; Computational Geometry and Object Modeling; geometric algorithms, languages, and systems**

I. Introduction

Modern day graphic systems are an amalgam of matrix, vector, and tensor algebras. These algebras run fast in computers, but the intricacies of the algebraic representation make it difficult to do the geometric insight. Within the last decade, Geometric Algebra has emerged as a powerful alternative to the classical matrix, vector and tensor algebras as a comprehensive conceptual language and computational system for computer science. Great research has been done demonstrating its powers of synthesis, especially with regards to control systems generated by twist groups and motor algebras ([1], [2]), but little research focuses on their application to surface modeling and animation. Although Geometric Algebra has been used for doing computer graphics ([3], [4]), its use for mesh deformation remains a relatively esoteric exercise. As a result, many of its formal characteristics have yet to be explored. We aim for illustrating the suitability of the conformal model of Geometric Algebra for representing deformable mesh models and encourage its use in geometric modeling applications.

In Conformal Geometric Algebra (CGA), rigid body motions can be represented as compact multivectors called *Motors*. Unlike homogeneous matrices, motors are *linear* transformations. CGA motors can be efficiently interpolated for generating continuously deformed motions, and that is one of the main features that we exploit in this work. On the other hand, the interpolation of matrices is difficult and expensive. The logarithm of a matrix can be defined but is not elementary and not in closed form. A straightforward way to compute it is to take the eigenvalue decomposition of the rigid body motion matrix in the homogeneous coordinate framework, and take the Nth root of the diagonal matrix. Such numerical techniques makes the matrix logarithm expensive to compute and hard to analyze.

A. Contributions

- A Conformal Geometric Algebra representation of deformable meshes that leads to an efficient formulation for spline-aligned deformations. We avoid the complex mathematical formulations related to such deformations. Instead we deal with a motor interpolation problem which has a simple and elegant formulation. Deformations produced are predictable and visually pleasant and they are computed at interactive rates for large models (over a hundred thousand vertices) without even using the power of GPU.

- A linear deformation method based on generalized barycentric coordinates. Motors are used for representing general surface deformations imposing positional and rotational constraints. Our method computes a local transformation for each vertex by interpolating user-defined key motors using the generalized barycentric coordinates computed on the surface. This method is also capable of deforming large meshes at interactive rates.
- Reformulation of popular methods such as FFD ([5]), Harmonic Coordinates ([6]) and ARAP Surface Modeling ([7]) into the Conformal Geometric Algebra. These reformulations extend the capabilities of the methods and leads to improvements in their quality or efficiency.

B. Related work

There have been many approaches for mesh modeling and editing. Each approach uses a different deformable mesh representation for the formulation and actual computing of the deformations. Space deformation techniques typically need to embed a surface mesh inside a volume geometry. The surface is typically represented as a function of its barycentric coordinates in the volume space. The deformation of the space inside the volume is transferred to the embedded surface. FFD techniques ([5]) need to embed a surface mesh inside a volume lattice. Other techniques have replaced the lattice with more general volumes ([3], [8], [9]). Among these techniques, the *cage*-based methods provide the ability to use unstructured meshes for deforming the subspaces inside them ([6], [10]–[12], [12]–[14]). In sections V and VI we show how FFD and Harmonic Coordinates methods ([5], [6]) can be easily extended by reformulating them in the CGA framework.

Spline-aligned methods are in the same spirit as the free-form deformation, but instead of using a volume to deform the space they use a 3D spline curve. A surface is represented as a spline function of a scalar field associated to the surface, the curve is defined by specifying where a few vertices of the mesh should deform to, and defining a set of constraints to a linear system ([15]–[18]). In section III we use the CGA framework to formulate a Spline-aligned method in which the surface is represented as a spline function acting on motors. Space deformation is directly transferred to the mesh by applying the interpolated motors.

Skinning techniques ([19], [20]) are probably the most popular mesh deformation techniques. Kavan *et. al.* ([4]) uses the dual quaternion framework for improving the interpolation of positional and directional quantities. They show that by reformulating the geometric skinning in the motor algebra, instead of linear matrix algebra, the quality of joint transformations blending can be improved, removing artifacts from model skin without increasing performance costs.

Gradient domain deformation techniques are characterized by applying modifications to differential rather than spatial coordinates of the models. Such methods try to preserve some differential properties of the mesh, mostly manipulating the differential coordinates of the undeformed mesh (i.e., gradient fields, Laplacian coordinates or the first/second fundamental forms of a surface) and then reconstructing the deformed mesh from the modified differential coordinates ([21]–[25]). The so-called As-Rigid-As-Possible (ARAP) deformation methods tend to produce physically plausible results at relatively low cost by solving a non-linear least squares problem ([7], [26], [27]). In section VII we show how to recast the ARAP method described in [7] into Geometric Algebra producing a formulation with a better geometric interpretation. Recent surveys on Gradient domain deformations can be found in [28] and [29].

In [30] a conformal deformation method, based on a quaternionic Dirac operator, is presented. The authors look for the best quasi-conformal deformation for a given surface i.e., a deformation that preserve angles on the surface as much as possible. The authors uses discrete differential geometry to setup a quaternion-based eigenvalue problem that leads to the desired solution. In contrast, the techniques discussed in this paper are not enforcing quasi-conformality, i.e., our usage of CGA motors does not automatically enforce globally quasi-conformal deformations.

II. Mathematical Background

A thorough introduction to CGA is far beyond the scope of this paper, for foundations on the subject with emphasis in computer science and computer graphics we refer the reader to [31], [32]. Here we present a brief introduction to the basics.

A. Universal Geometric Algebra

A geometric algebra is constructed over a *real vector space* $\mathbb{R}^{r,s}$ with dimension $r + s = n$, which means r basis vectors have a positive square, and s have a negative square. The associative *geometric product* is defined by the simple rule that the square of any vector is a scalar $aa = a^2 \in \mathbb{R}$. The vector a is said to be *null* if it squares to zero $a^2 = 0$. From the vector space $\mathbb{R}^{r,s}$, the geometric product generates the real geometric algebra $\mathbb{G}^{r,s}$ with elements $\{A, R, M, ...\}$ called multivectors.

For a pair of vectors, a symmetric inner product $a \cdot b = b \cdot a$ and antisymmetric outer product $a \wedge b = -b \wedge a$ can be defined implicitly by the geometric product $ab = a \cdot b + a \wedge b$ and $ba = b \cdot a + b \wedge a$. It is easy to prove that $a \cdot b = \frac{1}{2}(ab + ba)$ is scalar valued, while the quantity $a \wedge b = \frac{1}{2}(ab - ba)$, called a *bivector* or 2-vector, is a new algebraic entity that can be visualized as the two-dimensional analogue of a direction, that is, a *planar direction*.

The outer product can be generalized iteratively to define *k-vectors* by $a \wedge A_k = \frac{1}{2}(aA_k + (-1)^k A_k a)$ which generates a $(k + 1)$-vector from k-vector A_k. It follows that the outer product of k-vectors is the completely antisymmetric part of their geometric product: $a_1 \wedge a_2 \wedge ... \wedge a_k = \langle a_1 a_2 ... a_k \rangle_k$ where the angle bracket means k-vector part, and k is its *grade*. The term grade is used to refer to the number of vectors in any exterior product. This product vanishes if and only if the vectors are linearly dependent. Consequently, the maximal grade for nonzero k-vectors is $k = n$. It follows that every multivector A can be expanded into its k-vector parts and the entire algebra can be decomposed into k-vector subspaces:

$$\mathbb{G}^{r,s} = \sum_{k=0}^{n} \mathbb{G}_k^{r,s} = \{A = \sum_{k=0}^{n} \langle A \rangle_k\}$$

This is called a *grading* of the algebra.

Reversing the order of multiplication is called *reversion*, as expressed by $(a_1 a_2 ... a_k)\tilde{} = a_k ... a_2 a_1$ and $(a_1 \wedge a_2 \wedge ... \wedge a_k)\tilde{} = a_k \wedge ... \wedge a_2 \wedge a_1$, and the reverse of an arbitrary multivector is defined by $\tilde{A} = \sum_{k=0}^{n} \langle \tilde{A} \rangle_k$.

B. Conformal Geometric Algebra

The conformal group is the group of transformations that preserve angles. These include the rigid transformations. The conformal group on $\mathbb{R}^3$ has a natural representation in terms of rotations in a 5D space, with signature $\mathbb{R}^{4,1}$. So, in the same way that projective transformations are linearized by working in a 4D homogeneous space, conformal transformations are linearized in a 5D space. The conformal vector space poses five vectors $\{o, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \infty\}$, i.e., the euclidean basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ plus the Minkowski basis $\{o, \infty\}$. Minkowski basis vectors are null vectors $o^2 = \infty^2 = 0$ not orthogonal between them $o \cdot \infty = -1$ but are still orthogonal to the Euclidean basis vectors. The o vector can be interpreted as the point at the origin and the ∞ as the point at infinity. A general null vector p can be constructed given a Euclidean vector $\mathbf{p}$ as $p = o + \mathbf{p} + \frac{1}{2}\|\mathbf{p}\|^2\infty$. This null vector p can be interpreted as a Euclidean *point*. The existence of null vectors is guaranteed by the fact that the conformal vector space has a mixed signature. The main advantage of basing the description in a conformal setting is that *distance* is encoded simply. The Euclidean distance of two points p and q is defined as its inner product $p \cdot q = -\frac{1}{2}\|\mathbf{p} - \mathbf{q}\|^2$.

1) CGA Motors: Rigid body motions in the Euclidean space $\mathbb{R}^3$ are represented by orthogonal transformations in $\mathbb{G}^{4,1}$. A motor M is an even-grade multivector satisfying $M\tilde{M} = 1$ and is generated by reflections in well-chosen planes. Pure translation $T_\mathbf{v}$ over a vector $\mathbf{v}$ is generated by double reflection in two *parallel* planes separated by a vector $\mathbf{v}/2$ and pure rotation $R_{\mathbf{B}\theta}$ is generated by double reflection in two *incident* planes at the origin with a relative angle of $\theta/2$ between them. A motor can be represented as $M = T_\mathbf{v}\ R_{\mathbf{B}\theta}$. Motors can transform any geometric entity using the geometric product as $A' = M\ A\ \tilde{M}$. Every motor can be written in terms of the exponential of a bivector as $M = e^B$. The bivector B is the generator of the motion, and is an element of the Lie algebra of the conformal group.

2) Motor Interpolation: A motor can be applied gradually to geometric entities by splitting it in n equal parts through $M^{1/n} = e^{B/n}$ and then applying them in n time steps. A simple way to get the *linear* interpolation between two motors $M_1 = e^{B_1}$ and $M_2 = e^{B_2}$ is $M_\alpha = e^{(1-\alpha)B_1 + \alpha B_2}$. This formulation requires to take the motor logarithms $B_1 = \log(M_1)$ and $B_2 = \log(M_2)$. An expression for the motor logarithm may be found in [31].

III. Spline Aligned Deformation

We show a generic way to interpolate conformal motors through polynomial curves and spline curves. Interpolated motors are used to deform the 3D space and thus, the meshes parameterized in that space. Resulting deformations are known as *spline aligned* deformations or *spline path* deformations.

Let the spline function $S(\lambda) \in \mathbb{G}^{4,1}$ be defined as the convex combination of the key motors $\mathcal{R} = \{R_j\}_{j=1}^{n}$ and let the parameter $\lambda(\mathbf{p}_i)$ be the real valued function $\lambda : \mathbb{R}^3 \mapsto \mathbb{R}$ such that $0 \leq \lambda_i \leq 1$.

$$S(\lambda_i) = \exp\left(\sum_{j=1}^{n} W_j(\lambda_i) \log(R_j)\right) \tag{1}$$

The range of the basis function $W_j(\lambda_i)$ is a set of n scalars w_j^i forming a partition of unity $\sum w_j^i = 1$. Several schemes for basis functions are explored including linear, quadratic, cubic Bézier and cubic B-Splines. The spline function $S(\lambda)$ can be viewed as producing a motor M_i for a given point $\mathbf{p}_i \in \mathbb{R}^3$ in the sense that $M_i = S(\lambda(\mathbf{p}_i))$. In 3D space we can find a real scalar field of the form $\lambda : \mathbb{R}^3 \mapsto \mathbb{R}$ for a mesh $\mathcal{M}$ (or region of interest) by solving the discrete Laplace equation $\Delta\lambda = 0$ with Dirichlet boundary conditions as described in section VIII. Then, the function S will transform the conformal space, constraining it to follow the curved path and the orientation of the combined motors.

Given a surface represented by a mesh $\mathcal{M} = \{\mathcal{P}, \mathcal{G}\}$ where $\mathcal{P}$ is the set of m conformal points $\mathcal{P} = \{p_i\}_{i=1}^{m}$ and $\mathcal{G}$ is the connectivity graph, we define the deformation function f of the form $f : \mathcal{M} \mapsto \mathcal{M}$ which maps the mesh $\mathcal{M}$ into a deformed version $\mathcal{M}'' = \{p'' = f(p) \mid p \in \mathcal{P},\ p'' \in \mathcal{P}''\}$ as follows:

$$f(p_i) = M_i\ p_i\ \tilde{M}_i \tag{2}$$

where $M_i = S(\lambda(\mathbf{p}_i))$ is called the *deforming* motor.

A. Treatment of Handles

Mesh editing operations are usually specified through the manipulation of *handles* that allows the user to deform the 3D models (or region of interest) by selecting a set of points on the surface and move or rotate them. We denote handles as $\mathcal{H}_i$. By convention, all vertices on $\mathcal{H}_1$ have the weights $\lambda(\mathbf{p}_i) = 0, \forall \mathbf{p}_i \in \mathcal{H}_1$ and conversely all vertices on $\mathcal{H}_n$ have the weights $\lambda(\mathbf{p}_i) = 1, \forall \mathbf{p}_i \in \mathcal{H}_n$. So that (1) reduces to $S(0) = R_1$ and $S(1) = R_n$ for vertices on handles $\mathcal{H}_1$ and $\mathcal{H}_n$. The rest of key motors also contributes to deform the mesh, but the vertices on their handles are not transformed "rigidly".

In order to *bind* the "resting-pose" mesh $\mathcal{M}$ to the key motors, we compute the points p'_i using the key motors defined for the undeformed mesh:

$$p'_i = \tilde{M}_i\ p_i\ M_i$$

The original points p_i are not needed anymore. Having the key motors $\mathcal{R}$ and the scalar weights w_j^i we are able to compute the motors M_i at any time, so the points p_i' are what is needed for reconstructing the mesh by $p_i = M_i\, p_i'\, \tilde{M}_i$. Mesh deformation is achieved by modifying the key motors using the *handle* metaphor.

B. Spline Path Deformation

Linear interpolation is the simplest case for path deformation, though compelling deformations can be achieved even with this simple approach (Fig. 2). As expected, the surface is stretched (i.e., pure translation applied) or bent uniformly along a circular path. A simple extension of the linear case is to have three conformal motors R_1, R_2 and R_3 bound to the mesh and interpolate them using (1) with the quadratic Lagrange Polynomial weights $w_1^i = 2\lambda_i^2 - 3\lambda_i + 1$, $w_2^i = -4\lambda_i^2 + 4\lambda_i$ and $w_3^i = 2\lambda_i^2 - \lambda_i$. The paths are now quadratic curves passing through all the key motors and relatively complex poses can be produced with this method (Fig. 3).

This is straightforward to generalize further the previous cases to support general spline path deformation. Having n conformal motors $\mathcal{R} = \{R_j\}_{j=1}^n$ bound to the mesh, the spline path deformation is obtained by interpolating them using (1) with suitable spline weights, for instance the B-Spline weights $w_j^i = \mathcal{N}_{j,d}(\lambda_i)$ (where d is the degree of the B-Spline).

In terms of flexibility this method is much more powerful than linear and quadratic methods as can be seen in Fig. 4.

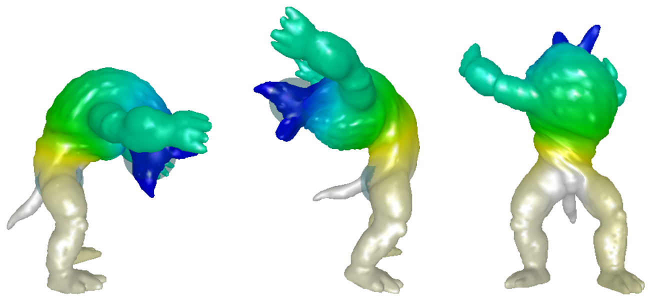


Fig. 2. Linear weights $w_1^i = 1 - \lambda_i$, $w_2^i = \lambda_i$

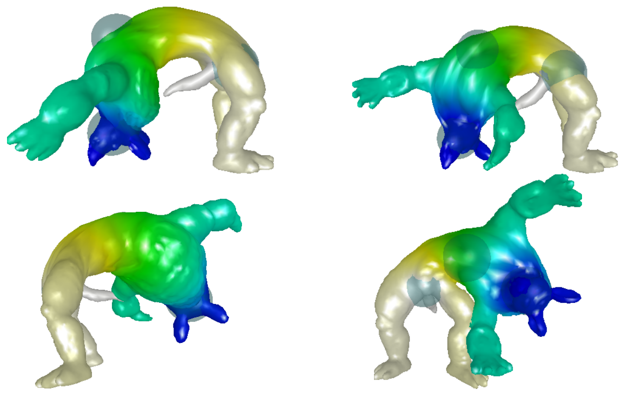


Fig. 3. Lagrange Polynomial weights

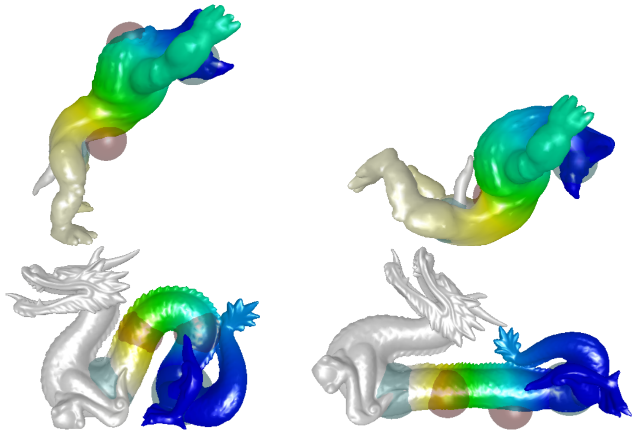


Fig. 4. Cubic B-Spline weights

C. Path Deformation through B-Spline Fitting

We show how the set of key motors $\{Q_i\}_{i=0}^n$ can be fitted by a cubic B-Spline deformer function. Our goal is to determine the set of motors $\{R_j\}_{j=0}^n$ such that

$$Q_i = \exp\left(\sum_{j=0}^{n} \mathcal{N}_j(\lambda_i)\log(R_j)\right) \tag{3}$$

where $\mathcal{N}_j(\lambda_i)$ are the cubic B-Spline weights. This problem can be solved using the well known linear least squares fitting as follows. Lets rewrite (3) in matrix form:

$$\mathrm{Q} = \mathrm{A\ R} \tag{4}$$

where the column vector Q is $\{\log(Q_i)\}_{i=0}^n$, the column vector R is $\{\log(R_j)\}_{j=0}^m$ and the matrix $\mathrm{A}_{ij} = \mathcal{N}_j(\lambda_i)$. In general, the matrix A is of dimensions $n+1 \times m+1$, where $n > m$ since we want to solve the linear system in the least squares sense. The best solution is obtained by solving the LLS normal equations:

$$\mathrm{A}^T\ \mathrm{Q} = (\mathrm{A}^T\ \mathrm{A})\ \mathrm{R} \tag{5}$$

The matrix $(\mathrm{A}^T\ \mathrm{A})$ can be ill-conditioned depending on the chosen *knot* vector. In that case the system can be solved by SVD, otherwise it can be prefactored by Cholesky method. The system must be solved six times for different right-hand-sides, since the bivector basis for the logarithm of a motor has six basis-bivector components $\{\mathbf{e}_{12}, \mathbf{e}_{31}, \mathbf{e}_{23}, \mathbf{e}_{1\infty}, \mathbf{e}_{2\infty}, \mathbf{e}_{3\infty}\}$.

As an application example we have implemented the Inverse Kinematic (IK) solver FABRIK ([33]) that control a single kinetic chain. The joints in the chain are the key motors $\{Q_i\}_{i=0}^n$ which in turn controls the path deformation of the female model. Results can be seen in Fig. 5.

IV. Deformation Based on Generalized Barycentric Coordinates

We can adapt the path deformation method described in section III to achieve mesh deformation with multiple hard handles. The idea is to bind each motor R_j with a set of vertices $\mathcal{H}_j$, so that $\lambda_j(\mathbf{p}_i) = 1, \forall \mathbf{p}_i \in \mathcal{H}_j$ and gradually

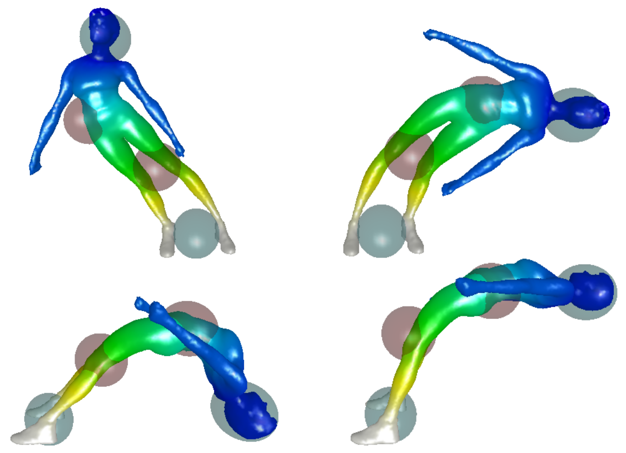


Fig. 5. B-Spline Fitting and Inverse Kinematics

decrease the influence of motor R_j in the vertices outside the handle H_j so that $\lambda_j(\mathbf{p}_i) = 0, \forall \mathbf{p}_i \in \bigcup_k^n \mathcal{H}_k, \forall k \neq j$. Doing the same for all key motors we end with n scalar fields λ_j, so that $M_i = \exp(\sum_{j=1}^n \lambda_j^i \log(R_j))$. Each scalar field λ_j is computed using the discrete Laplace equation with boundary conditions set to 1 for vertices on H_j and set to 0 for vertices on all the other handles (see section VIII). Laplace equation will smoothly interpolate the boundary conditions throughout the rest of mesh vertices.

The end result is that each vertex p_i of the mesh $\mathcal{M}$ has associated a set of scalars $\{\lambda_j\}_{j=1}^n$. As these scalar fields are harmonic functions, and due to the nature of the boundary conditions used for their computation, the sum of the scalar values associated with each vertex is equal to one i.e., $\sum_j^n \lambda_j^i = 1$. In effect, the scalars $\{\lambda_j\}_{j=1}^n$ are the barycentric coordinates of the vertices, also known as *Harmonic Coordinates*, with respect to the motors $\{R_j\}_{j=1}^n$. These Harmonic Coordinates are in fact the same as the ones described in [6] but are computed at the surface level, not in 3D space as in [6] which are more involved to compute.

Fine detail deformation can be achieved as shown in Fig. 6. Four motor handles are used for changing the facial expressions of the Max-face model. The method can also be used for global deformation as shown in the bottom row of Fig. 6.

V. Free Form Deformation

The FFD method need the definition of a lattice of control points $P_{u,v,k}$. An object embedded within the lattice is then deformed by defining the following mapping from the lattice to the object.

$$\mathbf{p}_i = \sum_{u=1}^n \sum_{v=1}^m \sum_{k=1}^l \mathcal{B}_u(\lambda_1^i)\mathcal{B}_v(\lambda_2^i)\mathcal{B}_k(\lambda_3^i)\mathbf{P}_{u,v,k} \tag{6}$$

where $\mathcal{B}_u$, $\mathcal{B}_v$ and $\mathcal{B}_k$ are Bézier basis functions and $\{\lambda_1^i, \lambda_2^i, \lambda_3^i\}$ are the local coordinates of the point $\mathbf{p}_i$ in the lattice space. The user thus deals with a level of detail dictated by the density of the control lattice. Interactions of the user are basically to specify new positions for the control

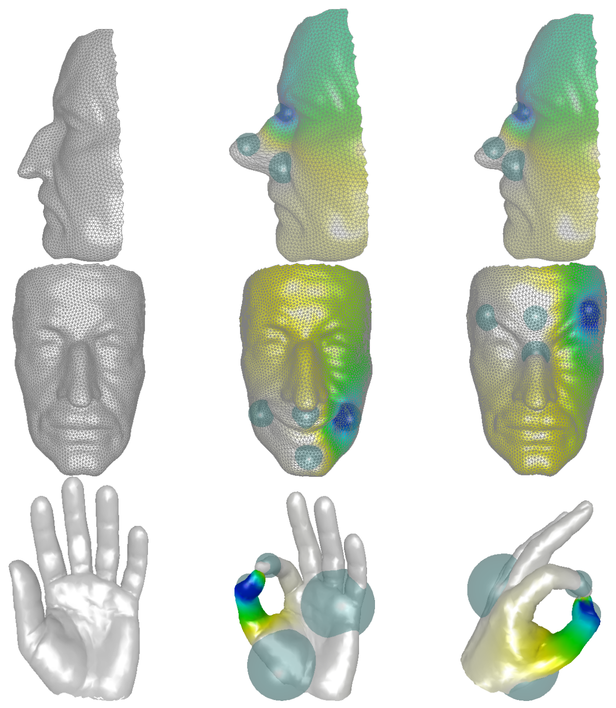


Fig. 6. Top and Middle rows: motor handles deform Max's face. Bottom row: motor handles deform the hand's fingers.

points. Rotational information is not taken into account by the deformation system, so in some cases it is very difficult to produce convincing deformations when they involve large rotations of the lattice. We show that this limitation can be overcome easily by introducing conformal motors into the system.

The control points $\mathbf{P}_{u,v,k}$ of the 3D lattice are replaced by conformal motors $R_{u,v,k}$ which are bound to the mesh vertices p_i by means of the motor M_i so that the mesh vertices can be expressed in terms of the motors as $p_i = M_i \; o \; \tilde{M}_i$, where o is the conformal *point at the origin* and the motor M_i is defined as:

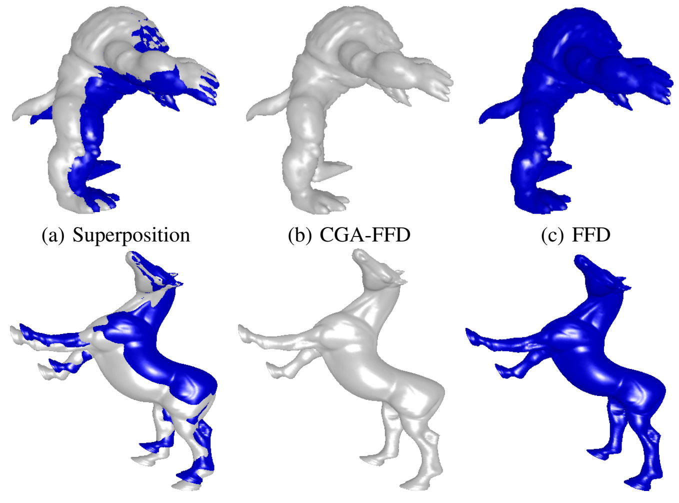
$$M_i = \exp\left(\sum_{u=1}^n \sum_{v=1}^m \sum_{k=1}^l w_{u,v,k}^i \log(R_{u,v,k})\right) \tag{7}$$

$$w_{u,v,k}^i = \mathcal{B}_u(\lambda_1^i)\mathcal{B}_v(\lambda_2^i)\mathcal{B}_k(\lambda_3^i) \tag{8}$$

Conformal motors allow us to extend the FFD handles from simple positional constraints to motions that deform the space beyond translations, including rotations and screws. Two immediate improvements resulting from this adaptation are the better bending and twisting of meshes. We can see the comparison of our technique with the classic FFD in Fig. 7.

VI. Harmonic Coordinates

The HC method consists of binding a cage to the mesh $\mathcal{M}$ so that Euclidean vertices $\mathbf{p}_i$ can be expressed in terms of the cage vertices $\mathbf{P}_j$ so that $\mathbf{p}_i = \sum_{j=1}^n \lambda_j^i \mathbf{P}_j$. So, each vertex $\mathbf{p}_i$ has associated a set of scalars $\{\lambda_j\}_{j=1}^n$. The scalars are the generalized barycentric coordinates of the vertex $\mathbf{p}_i$ since


(a) Superposition (b) CGA-FFD (c) FFD

(d) Superposition (e) CGA-FFD (f) FFD

Fig. 7. Comparison of CGA-based FFD and the classic FFD methods. The deformed lattice is the same for each comparison, though some CGA-based handles are rotated in order to align the rotation field to follow the deformed lattice.

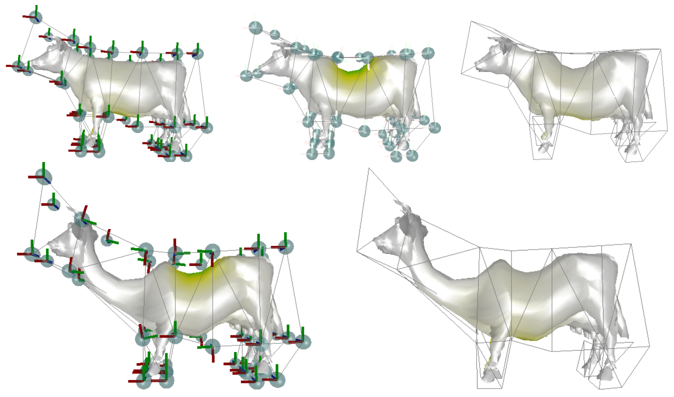


Fig. 8. First row: cow's back is deformed by rotating the motor handles (no translation applied). Bottom row: cow's shape is turned into a camel shape by rotating several handles and translating the handles around the neck.

they are positive and form a partition of unity $\sum_j^n \lambda_j^i = 1$ and are interpolants with respect to the cage vertices. When the barycentric coordinates are harmonic functions, they are known as *Harmonic Coordinates*.

As occurs with the FFD method, the interactions of the user are basically specifying new poses for the cages. Rotational information is not taken into account by the deformation system, so it is hard to produce complex deformations such as the ones shown in Fig. 8. We show that this limitation can be easily overcome by introducing conformal motors into the system.

The idea is pretty similar to the FFD case, the cage vertices $\mathbf{P}_j$ are replaced by conformal motors R_j which are bound to the conformal vertices p_i by means of the motor M_i so that the mesh vertices can be expressed in terms of the motors as $p_i = M_i \; o \; \tilde{M}_i$, where o is the conformal *point at the origin* and the motor M_i is defined as $M_i = \exp(\sum_{j=1}^n \lambda_j^i \log(R_j))$.

The usage of motor handles allows us to deform the space beyond translations as can be seen in Fig. 8. The handle positions don't need to be modified in order to deform the space. Complex deformations that are hard to achieve with the positional handles can be achieved easily using the motor handles.

VII. As-Rigid-As-Possible Surface Modeling

The main idea of this method is breaking the surface into overlapping cells $\mathcal{C}_i$ and seek for keeping the cells transformations as rigid as possible in the least squares sense. Overlap of the cells is necessary to avoid surface stretching or shearing at the boundary of the cells. The cell rigidity measure is:

$$E(\mathcal{C}_i, \mathcal{C}_i') = \sum_{j \in \mathcal{N}(i)} w_{ij} \|(\mathbf{p}_i' - \mathbf{p}_j') - \mathrm{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2 \qquad (9)$$

where $\mathbf{p}_i$ are the original mesh vertices, $\mathbf{p}_i'$ are the deformed vertices, $\mathbf{p}_j \in \mathcal{N}(i)$ are the incident one-ring neighbors of $\mathbf{p}_i$ (similarly for $\mathbf{p}_j'$) and the weights w_{ij} are the cotangent weights. The matrix R_i is the best rigid transformation, in the least squares sense, relating the original and the deformed vertices.

So, the best rigid deformation is obtained by solving the following optimization problem:

$$arg \min_{\mathbf{p}', \mathrm{R}} \sum_1^m w_i E(\mathcal{C}_i, \mathcal{C}_i') \qquad (10)$$

This is a non-linear optimization problem that can be efficiently solved by a simple iterative method that solves two linear least squares sub-problems on each iteration.

We show how to reformulate this method to their more encompassing Geometric Algebra counterpart. Our aim is producing a coordinate-free formulation with a better geometric interpretation and update the use of Geometric Algebra to the non-linear variational deformation methods. To this goal we have to express the rigid energy in terms of Euclidean rotors $M_i = e^{-\frac{1}{2}\mathbf{B}_i \theta_i}$ where $\mathbf{B}_i$ is a Euclidean bivector:

$$E(\mathcal{C}_i, \mathcal{C}_i') = \sum_{j \in \mathcal{N}(i)} w_{ij} \|\mathbf{v}_{ij}' - M_i \mathbf{v}_{ij} \tilde{M}_i\|^2 \qquad (11)$$

where $\mathbf{v}_{ij} = \mathbf{p}_i - \mathbf{p}_j$ and $\mathbf{v}_{ij}' = \mathbf{p}_i' - \mathbf{p}_j'$. Following [34], by differentiating (11) w.r.t. M_i, we get the optimality condition $\sum_{j \in \mathcal{N}(i)} w_{ij}(M_i \mathbf{v}_{ij} \tilde{M}_i) \wedge \mathbf{v}_{ij}' = 0$, which makes sense since we want the M_i which makes $\mathbf{v}_{ij}$ "most parallel" to $\mathbf{v}_{ij}'$. That is equivalent to finding the rotor M_i that *maximizes* the following energy $\sum_{j \in \mathcal{N}(i)} w_{ij}(M_i \mathbf{v}_{ij} \tilde{M}_i) \cdot \mathbf{v}_{ij}'$. The solution for M_i utilizes the linear algebra framework of quaternions in which the optimal rotor M_i is the eigenvector associated with the largest positive eigenvalue of a matrix P ([35]), which is constructed from the elements of the covariance matrix $\mathrm{S} = \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{v}_{ij} \mathbf{v}'^T_{ij}$, so that

$$\mathrm{P} = \begin{bmatrix} \mathrm{S}_a & \mathrm{S}_{12} - \mathrm{S}_{21} & \mathrm{S}_{20} - \mathrm{S}_{02} & \mathrm{S}_{01} - \mathrm{S}_{10} \\ \mathrm{S}_{12} - \mathrm{S}_{21} & \mathrm{S}_b & \mathrm{S}_{01} + \mathrm{S}_{10} & \mathrm{S}_{20} + \mathrm{S}_{02} \\ \mathrm{S}_{20} - \mathrm{S}_{02} & \mathrm{S}_{01} + \mathrm{S}_{10} & \mathrm{S}_c & \mathrm{S}_{12} + \mathrm{S}_{21} \\ \mathrm{S}_{01} - \mathrm{S}_{10} & \mathrm{S}_{20} + \mathrm{S}_{02} & \mathrm{S}_{12} + \mathrm{S}_{21} & \mathrm{S}_d \end{bmatrix}$$

$$\mathrm{S}_a = \mathrm{S}_{00} + \mathrm{S}_{11} + \mathrm{S}_{22} \qquad \mathrm{S}_b = \mathrm{S}_{00} - \mathrm{S}_{11} - \mathrm{S}_{22}$$

$$\mathrm{S}_c = -\mathrm{S}_{00} + \mathrm{S}_{11} - \mathrm{S}_{22} \qquad \mathrm{S}_d = -\mathrm{S}_{00} - \mathrm{S}_{11} + \mathrm{S}_{22}$$

Since the matrix P is symmetric the eigenvectors are real. The eigenvector with the largest eigenvalue is the quaternion we want $q = q_0 + iq_x + jq_y + kq_z$.

$$M_i = \frac{1}{\|q\|}(q_0 - q_z e_{12} - q_x e_{23} - q_y e_{31})$$

The second step is computing the optimal vertices $\mathbf{p}'_i$ that minimizes the rigidity energy (11) in the least squares sense. This can be achieved by taking the partial derivatives of (11) w.r.t. $\mathbf{p}'_i$ and equating the result to zero, which results in:

$$\sum_{j \in \mathcal{N}(i)} w_{ij}\mathbf{v}'_{ij} = \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2}(M_i \mathbf{v}_{ij} \tilde{M}_i + M_j \mathbf{v}_{ij} \tilde{M}_j) \qquad (12)$$

The minimum is attained when the total change on deformed edges $\mathbf{v}'_{ij}$ is equal to the change on the sets of rigid edges generated by the rigid motions M_i and M_j. Equation (12) can be expressed in matrix form as $\Delta \mathrm{P}' = \mathrm{C}$, where Δ is the symmetric Laplace-Beltrami operator, P' is the column of target positions and C is the right hand side of (12). That is, a Poisson equation. *Handle* constraints of the form $\mathbf{p}'_i = \mathbf{c}_i$ are incorporated into the system by means of substituting the corresponding variables i.e., erasing respective rows and columns from Δ and updating the right-hand side with the values $\mathbf{c}_i$. The system is then solved in the least squares sense:

$$(\Delta^T \Delta)\ \mathrm{P}' = \Delta^T\ \mathrm{C} \qquad (13)$$

The sparse symmetric-matrix $\Delta^T \Delta$ can be pre-factored with Cholesky decomposition. Mesh deformations are achieved by repositioning the constrained vertices $\mathbf{p}'_i = \mathbf{c}_i$, solving the linear subproblem for rotors M_i, updating the C column and solving the LLS system for $\mathbf{p}'_i$. Having a good initial guess, the convergence is typically achieved in less than 10 iterations (three to four iterations already provides compelling results).

The Euclidean rotors are as efficient as quaternions (indeed they are quite the same). Rotors requires less storage than rotation matrices (just four numbers) and operations such as scalar multiplication, composition of transformations, and addition are more efficient than matrices. The Fig. 9 shows that our results are exactly the same as the results produced by the formulation in [7].

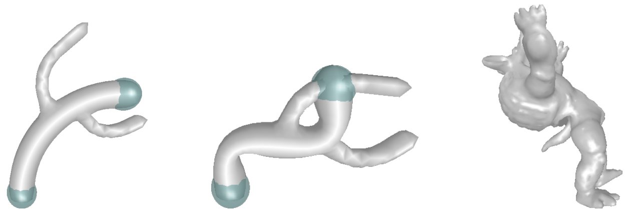


Fig. 9. Deformations obtained minimizing the GA-based ARAP energy.

VIII. Harmonic Field Computation

We define the harmonic mapping $\lambda(p)$ of the form $\lambda : \mathcal{M} \mapsto \mathbb{R}$, as a solution of the Laplace's equation $\Delta\lambda = 0$, subject to Dirichlet boundary constraints, where Δ represents the Laplacian operator (∇^2).

In the discrete setup, the Laplace operator Δ turns into the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ w.r.t the mesh $\mathcal{M}$. Let the user define n sets of *fixed* vertices $\mathcal{F}_i$ and m sets of *handle* vertices $\mathcal{H}_j$. Dirichlet boundary conditions for $\lambda(p)$ with $p \in \mathcal{F}_i, \forall i$ are set to 0, and for $\lambda(p)$ with $p \in \mathcal{H}_j, \forall j$ are set to 1. Having that, the finite differences discretization of the Laplace-Beltrami equation is as follows:

$$\Delta_{\mathcal{M}}\lambda(p_i) = w_i \sum_{p_j \in \mathcal{N}_1(p_i)} w_{ij}(\lambda(p_j) - \lambda(p_i)) \qquad (14)$$

where $p_j \in \mathcal{N}_1(p_i)$ are the incident one-ring neighbors of p_i. The normalization weights $w_i = \frac{1}{\sum_j w_{ij}}$ and the edge weights $w_{ij} = w_{ji}$ are the Mean-Value weights ([36]). Mean-Value weights are always positive, no matter the triangle quality, and therefore always produce smooth scalar fields.

Equation (14) can be expressed as a linear matrix system $L_{\mathcal{M}}\lambda = 0$ where $L_{\mathcal{M}}$ is sparse and positive semi-definite. This system can be efficiently solved for large meshes by sparse direct solvers like the **SuperLU** package ([37]).

IX. Implementation details

All the deformation methods described in this paper have been implemented using the Gaigen2 Geometric Algebra package ([38]). Gaigen2 is a C++ library designed to be as efficient as 4D linear matrix algebra (i.e., homogeneous coordinates). We have tested our implementations on meshes with increasing complexity and all have ran at interactive frame rates without using the power of the GPU.

X. Conclusion and Future Work

This paper investigates the emerging field of Conformal Geometric Algebra as a new tool for mesh deformation. The ability of expressing rigid-body transformations in a linear space is immensely useful when attempting to smoothly interpolate positions and orientations, since the space of rotation matrices is not linear. For this, geometric algebra can be regarded as a technique related to the use of quaternions and dual quaternions. The sub-algebra of Euclidean rotors is isomorphic to quaternions and the sub-algebra of conformal motors is isomorphic to dual quaternions. However, quaternions and dual quaternions can only be applied to vectors (i.e., pure quaternions). On the other hand, Euclidean rotors and conformal motors can be applied to any geometric entity that can be expressed in the algebra.

We have demonstrated that Conformal Geometric Algebra is a powerful tool for representing deformable meshes. The effort needed for integrating it into the existing methods is minimal and it pays off. We have demonstrated how the CGA-based mesh representations conduct to a fast and easy formulation for the Spline-aligned deformation and a formulation for linear surface deformation based on generalized barycentric coordinates. Also, we have reformulated the Free-Form Deformation (FFD), Harmonic Coordinates (HC) and As-Rigid-As-Possible (ARAP) Surface Modeling techniques into the CGA framework and discussed the advantages of these reformulations.

There is a lot of room for future work. In particular the development of appropriate numerical techniques for motor estimation completely within the framework of geometric algebra as well as having an appropriate discretization theory for Geometric Calculus. We believe that these tools will have an enormous impact in this field.

References

[1] G. Sommer, B. Rosenhahn, and C. Perwass, "The twist representation of free-form objects," in *Geometric Properties from Incomplete Data*, ser. Computational Imaging and Vision, R. Klette, R. Kozera, L. Noakes, and J. Weickert, Eds., vol. 31. Dagstuhl, Wadern, Germany: Springer, 2006, pp. 3–22.

[2] E. Bayro-Corrochano, *Geometric Computing: For Wavelet Transforms, Robot Vision, Learning, Control and Action*. London: Springer Verlag, 2010.

[3] R. Wareham, J. Cameron, and J. Lasenby, "Applications of conformal geometric algebra in computer vision and graphics," in *Proceedings of the 6th international conference on Computer Algebra and Geometric Algebra with Applications*, ser. IWMM'04/GIAE'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 329–349.

[4] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan, "Geometric skinning with approximate dual quaternion blending," *ACM Trans. Graph.*, vol. 27, no. 4, pp. 105:1–105:23, Nov. 2008.

[5] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '86. New York, NY, USA: ACM, 1986, pp. 151–160.

[6] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki, "Harmonic coordinates for character articulation," in *ACM SIGGRAPH 2007 papers*, ser. SIGGRAPH '07. New York, NY, USA: ACM, 2007.

[7] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Proceedings of the fifth Eurographics symposium on Geometry processing*, ser. SGP '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 109–116.

[8] M. Botsch and L. Kobbelt, "Real-time shape editing using radial basis functions," in *Computer Graphics Forum*, 2005, pp. 611–621.

[9] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt, "Primo: coupled prisms for intuitive surface modeling," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, ser. SGP '06. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 11–20.

[10] M. S. Floater, G. Kós, and M. Reimers, "Mean value coordinates in 3d," *Comput. Aided Geom. Des.*, vol. 22, no. 7, pp. 623–631, Oct. 2005.

[11] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes," in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005, pp. 561–566.

[12] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum, "Subspace gradient domain mesh deformation," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006, pp. 1126–1134.

[13] Y. Lipman, D. Levin, and D. Cohen-Or, "Green coordinates," in *ACM SIGGRAPH 2008 papers*, ser. SIGGRAPH '08. New York, NY, USA: ACM, 2008, pp. 78:1–78:10.

[14] M. Ben-Chen, O. Weber, and C. Gotsman, "Variational harmonic maps for space deformation," in *ACM SIGGRAPH 2009 papers*, ser. SIGGRAPH '09. New York, NY, USA: ACM, 2009, pp. 34:1–34:11.

[15] K. Singh and E. Fiume, "Wires: a geometric deformation technique," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '98. New York, NY, USA: ACM, 1998, pp. 405–414.

[16] S. Forstmann, J. Ohya, A. Krohn-Grimberghe, and R. McDougall, "Deformation styles for spline-based skeletal animation," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ser. SCA '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 141–150.

[17] W. v. Funck, H. Theisel, and H.-P. Seidel, "Explicit control of vector field based shape deformations," in *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, ser. PG '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 291–300.

[18] I. Llamas, A. Powell, J. Rossignac, and C. D. Shaw, "Bender: a virtual ribbon for deforming 3d shapes in biomedical and styling applications," in *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, ser. SPM '05. New York, NY, USA: ACM, 2005, pp. 89–99.

[19] A. Mohr, L. Tokheim, and M. Gleicher, "Direct manipulation of interactive character skins," in *Proceedings of the 2003 symposium on Interactive 3D graphics*, ser. I3D '03. New York, NY, USA: ACM, 2003, pp. 27–30.

[20] P. G. Kry, D. L. James, and D. K. Pai, "Eigenskin: Real time large deformation character skinning in hardware," in *In ACM SIGGRAPH Symposium on Computer Animation*. ACM Press, 2002, pp. 153–159.

[21] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, "Mesh editing with poisson-based gradient field manipulation," in *ACM SIGGRAPH 2004 Papers*, ser. SIGGRAPH '04. New York, NY, USA: ACM, 2004, pp. 644–651.

[22] R. Zayer, C. Rssl, Z. Karni, and H. peter Seidel, "Harmonic guidance for surface deformation," in *In Proc. of Eurographics 05*, 2005, pp. 601–609.

[23] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or, "Linear rotation-invariant coordinates for meshes," in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005, pp. 479–487.

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

[25] H. Fu, O. K.-C. Au, and C.-L. Tai, "Effective derivation of similarity transformations for implicit Laplacian mesh editing," *Computer Graphics Forum*, vol. 26, no. 1, pp. 34–45, 2007.

[26] N. Paries, P. Degener, and R. Klein, "Simple and efficient mesh editing with consistent local frames," in *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*. IEEE Computer Society, 2007, pp. 461–464.

[27] W. Xu, K. Zhou, Y. Yu, Q. Tan, Q. Peng, and B. Guo, "Gradient domain editing of deforming mesh sequences," in *ACM SIGGRAPH 2007 papers*, ser. SIGGRAPH '07. New York, NY, USA: ACM, 2007.

[28] M. Botsch and O. Sorkine, "On linear variational surface deformation methods," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 213–230, Jan. 2008.

[29] W.-W. Xu and K. Zhou, "Gradient domain mesh deformation: a survey," *J. Comput. Sci. Technol.*, vol. 24, no. 1, pp. 6–18, Jan. 2009.

[30] K. Crane, U. Pinkall, and P. Schröder, "Spin transformations of discrete surfaces," in *ACM SIGGRAPH 2011 papers*, ser. SIGGRAPH '11. New York, NY, USA: ACM, 2011, pp. 104:1–104:10.

[31] L. Dorst, D. Fontijne, and S. Mann, *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry (The Morgan Kaufmann Series in Computer Graphics)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[32] C. Doran and A. Lasenby, *Geometric Algebra for Physicists*. Cambridge University Press, 2003.

[33] L. Dorst and J. Lasenby, Eds., *Guide to Geometric Algebra in Practice*. Springer, 2011.

[34] J. Lasenby, W. J. Fitzgerald, A. N. Lasenby, and C. J. L. Doran, "New geometric methods for computer vision: An application to structure and motion estimation," *Int. J. Comput. Vision*, vol. 26, no. 3, pp. 191–213, Feb. 1998.

[35] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[36] M. S. Floater, "Mean value coordinates," *Computer Aided Geometric Design*, vol. 20, p. 2003, 2003.

[37] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, "A supernodal approach to sparse partial pivoting," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, pp. 720–755, 1999.

[38] D. Fontijne, "Efficient implementation of geometric algebra," Ph.D. dissertation, University of Amsterdam, 2007.