Linear-Time Estimation of Smooth Rotations in ARAP Surface Deformation

Mauricio Cele Lopez Belon

Abstract. In recent years the As-Rigid-As-Possible with Smooth Rotations (SR-ARAP [2]) technique has gained popularity in applications where an isommetric-type of surface mapping is needed. The advantage of SR-ARAP is the quality of deformation results which is comparable to more costly volumetric techniques operating on tetrahedral meshes. However, the main drawback of this technique is its performance. The SR-ARAP relies on local/global optimization approach to minimize the non-linear least squares energy. The power of this technique resides on the local step. The local step estimates the best local rotation of a small region of surface, or cell, with respect of its neighboring cells so a local change in one cell's rotation affect the neighboring cell's rotations and viceversa. The local step requires a global convergence of rotation changes. Currently the local step is solved in an iterative fashion, where the number of iterations needed to reach convergence can be prohibitively large and so, in practice, only a fixed number of iterations is possible. This trade-off is, in some sense, defeating the goal of SR-ARAP. We propose a linear-time closed-form solution for estimating the codependent rotations of the local step by solving a sparse linear system of equations. Our method is more efficient than state-of-the-art since no iterations are needed and optimized sparse linear solvers can be leveraged to solve this step in linear time. It is also more accurate since it is a closed-form solution. We apply our method to generate interactive surface deformation, we also show how a multirresolution optimization can be applied to achieve real-time animation of large surfaces.

Mathematics Subject Classification (2010). Parallel algorithms 68W10; Clifford algebras, spinors 15A66.

Keywords. Geometric Algebra, Quaternion Estimation, Mesh Deformation.

1. Introduction

Quatily of surface deformation aim to preserve local properties of shapes as much as possible, removing distortion in the form of shear and stretch. As-Rigid-As-Possible with Smooth Rotations (SR-ARAP) aims to create a surface mapping that minimize the deviation from rigid behavior on the local scale. It also makes that rigid transformations vary smoothly on local neighborhoods distributing the distortions uniformly over the surface. The resulting deformations not only preserves the shape at small scale but also at large scale significantly increasing the quality of the final result.

There are two main disadvangates on the current SR-ARAP method. First is the local step, which looks for best rotations of corresponding surface cells while keeping neighbor rotations similar to each other. It needs to reach global convergence for the whole mesh. Currently the local step is solved with a relaxation method i.e., optimizing rotations for individual cells keeping cell's neighbor rotations as constant and also optimizing neighbor rotations in the same manner repeating that process until convergence. Although the cost per relaxation iteration can be reduced considerably by avoiding SVD calculations ([6]) usually a fixed number of iterations is used in order to bound the solver time. That compromise weaks the purpose of SR-ARAP i.e, the uniform distribution of distortions over the surface, since the reduced number of iterations (usually two or three) might be not enough to reach optimal rotations across the surface. Second is the high computational cost, that increases drastically when the number of vertices grows. The SR-ARAP is a non-linear technique that requires several iterations of local/global optmimization to converge.

In this paper we address both drawbacks of SR-ARAP, first we propose a method to solve the local step as a single linear system of equations, avoiding the need of iteratively solve a series of SVD problems which are codependant to each other. Second we propose a simple multiresolution method based on solving the non-linear system on a simplified surface mesh first and then use harmonic interpolation of rotations to transfer the optimized rotations to the full resolution mesh, leveraging the Laplacian matrix needed to solve the global step of the optimization.

To find the best rotations matching two sets of vectors in a global manner we use the mathematical framework of Geometric Algebra (GA) since the treatment of rotations is simpler than quaternions framework. Quaternions are isomorphic to GA rotors so implementation of this method in terms of quaternions is trivial.

2. Related Work

3. Geometric Algebra $\mathbb{G}_3$

A thorough introduction to Geometric Algebra is out of the scope of this paper (we refer the reader to [1] for an in-depth introduction). In this section we shortly introduce the relevant GA elements relevant to our work.

The geometric algebra $\mathbb{G}_{3,0,0}$ is constructed over a real vector space $\mathbb{R}^3$, with basis vectors $\{e_1, e_2, e_3\}$. The associative geometric product is defined on vectors so that the square of any vector a is a scalar $aa = a^2 \in \mathbb{R}$ and the geometric product of two vectors a and b is $ab = a \cdot b + a \wedge b$ and $ba = b \cdot a - a \wedge b$. From the vector space $\mathbb{R}^3$, the geometric product generates the geometric algebra $\mathbb{G}_{3,0,0}$ with elements $\{X, R, A...\}$ called multivectors.

For a pair of vectors a and b, 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. It is easy to prove that $a \cdot b = \frac{1}{2}(ab + ba)$ is a scalar, 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 i.e., a planar direction. Similar to vectors, bivectors can be decomposed in a bivector basis $\{e_{12}, e_{13}, e_{23}\}$ where $e_{ij} = e_i \wedge e_j$.

The outer product of three vectors $a \wedge b \wedge c$ generates a 3-vector also known as the pseudoscalar, because the trivector basis consist of single element $e_{123} = e_1 \wedge e_2 \wedge e_3$. Similarly, the scalars are regarded as 0-vectors whose basis is the number 1. 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 3. It follows that every multivector X can be expanded into its k-vector parts and the entire algebra can be decomposed into k-vector subspaces:

$$\mathbb{G}_3 = \sum_{k=0}^{3} \mathbb{G}_3^k = \{X = \sum_{k=0}^{3} \langle X \rangle_k\}$$

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 $\tilde{X} = \sum_{k=0}^{3} \langle \tilde{X} \rangle_k$.

3.1. Rotors

Rotations are even grade multivectors known as rotors. We denote the subalgebra of rotors as $\mathbb{G}_3^+$. A rotor R can be generated as the geometric product of an even number of vectors. A reflection of any k-vector X in a plane with normal n is expressed as the sandwitch product $(-1)^k nXn$. The most basic rotor R is defined as the product of two unit vectors a and b with angle of $\frac{\theta}{2}$.

The rotation plane is the bivector $B = \frac{a \wedge b}{\|a \wedge b\|}$.

$$ab = a \cdot b + a \wedge b = \cos\left(\frac{\theta}{2}\right) + B \sin\left(\frac{\theta}{2}\right). \tag{3.1}$$

Rotors act on all k-vectors using the sandwitch product $X' = RX\tilde{R}$, where $\tilde{R}$ is the reverse of R and can be obtained by reversing the order of all the products of vectors.

3.2. Bivector products

We define the commutator product of two bivectors A and B as $A \times B = \frac{1}{2}(AB - BA)$. The commutator product of bivectors in $\mathbb{G}_3$ can be interpreted as a cross-product of bivectors i.e., the resulting bivector $C = A \times B$ is orthogonal to both A and B. The commutator product allow us to define the geometric product of two bivectors as $AB = A \cdot B + A \times B$. The inner product of bivectors differs from the inner product of vectors on the sign, since the square of bivectors is negative, the inner product of bivectors is a negative scalar e.g., $(ae_{12} + be_{13} + ce_{23}) \cdot (de_{12} + ee_{13} + fe_{23}) = -ad - be - cf$.

3.3. Quaternions

A quaternion $Q = w + \vec{v}$ consists of two parts: a scalar part w and vector part $\vec{v}$ which denotes the axis of rotation. The vector part $\vec{v}$ is defined in a basis of complex vectors $\{i, j, k\}$ that squares to -1 and anticommute i.e., $i^2 = j^2 = k^2 = -1$ and $ijk = -1$. In geometric algebra they corresponds to bivectors:

$$\begin{gathered} i = e_{23} \quad j = e_{13} \quad k = e_{12} \\ ijk = e_{23}e_{13}e_{12} = -1 \end{gathered} \tag{3.2}$$

Rotors can easily be transformed to quaternions and vice versa. A rotor $R = w + B$ corresponds with a quaternion $Q = w + \vec{v}$, where $B = \alpha e_{12} + \beta e_{13} + \gamma e_{23}$ and $\vec{v} = \gamma i + \beta j + \alpha k$.

4. As-Rigid-As-Possible Shape Deformation with Smooth Rotations

Given two meshes Q and P consisting of vertices q_i and p_i respectively, and directed edges $p_{ij} = p_j - p_i$ and $q_{ij} = q_j - q_i$, the discrete As-Rigid-As-Possible with Smooth Rotations (SR-ARAP) energy is defined as:

$$E(Q,P) = \sum_{i=1}^{m} \Big(\sum_{j \in N(i)} c_{ij} \|R_i(q_j - q_i)\tilde{R}_i - (p_j - p_i)\|^2 + \alpha A \sum_{j \in N(i)} w_{ij} \|R_j - R_i\|^2 \Big)$$

where $R_1, ..., R_m \in \mathbb{G}_3^+$ are local rotors, $N(i)$ denote the set of 1-ring neighbors of vextext at position i, c_{ij} are weighting coefficients such that $\sum_j^n c_{ij} = 1$, typically the familiar cotangent weights, w_{ij} are positive weighting coefficients such that $\sum_j^n w_{ij} = 1$, A is the mesh area used to make the energy scale invariant and α is a positive scalar parameter.

The main idea of this method is breaking the surface into overlapping cells 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 first term is a membrane energy which penalizes stretching and shearing of a cell and the second term is the bending energy which penalizes the difference between a cell's rotation and the rotations of its neighboring cells. The objective of the membrane term is to lower the distortion of a cell by keeping the map differential close to rigid. The objective of the bending term is to keep the variation in the rotations in a cell neighborhood low, such that the neighborhood would transform as a unit, as much as possible.

The vertices of mesh Q are in original position while vertices of mesh P are the deformed vertices and the rotor R_i is the best rigid transformation, in the least squares sense, relating the original and the deformed vertices. This is a non-linear optimization problem that is tipically solved by a iterative local/global method that solves two linear sub-problems on each iteration.

The first step, so called local step, is to consider the vertices of P constant and obtain the best rigid transformation R_i for each cell. The second step, so called global step, is to consider the rotations R_i constant and computing the optimal deformed vertices p_i in the least squares sense.

5. Global Step

The global step is computing the optimal vertices $\{p_i\} \in P$.

$$E(P) = \min_{p_1,\ldots,p_m \in \mathbb{R}^3} \sum_{i=1}^{m} \Big(\sum_{j \in N(i)} c_{ij} \| R_i q_{ij} \tilde{R}_i - p_{ij} \|^2 + \alpha A \sum_{j \in N(i)} w_{ij} \| R_i - R_j \|^2 \Big)$$

Taking the partial derivatives of $E(P)$ w.r.t. p_i and equating the result to zero lead us to obtain the linear system of Equation (4.1) ([3]):

$$\sum_{j \in N(i)} c_{ij}(p_j - p_i) = \sum_{j \in N(i)} \frac{c_{ij}}{2} (R_i q_{ij} \tilde{R}_i + R_j q_{ij} \tilde{R}_j) \tag{5.1}$$

which can be expressed in matrix form as L P = C, where L is the symmetric Laplace-Beltrami operator, P is the column of target positions and C is the right hand side of Equation (4.1). Constraints of the form $p_i = p_i^{const}$ are incorporated into the system by substituting the corresponding variables i.e., erasing respective rows and columns from L and updating the right-hand side with the values c_i. The system is then solved in the least squares sense:

$$(\mathrm{L}^T \mathrm{L})\ \mathrm{P} = \mathrm{L}^T\ \mathrm{C} \tag{5.2}$$

6. Local Step

As described in a previous sections, the rotations matching cell's are codependent to each other over the whole mesh. The current approach is to optimize the rotations with a relaxation method in which the rotation of each cell independently, while keeping the neighbor rotations fixed, repeating it until convergence. At least two relaxation iterations should be done per each global iteration. In this section we show how the local step can be solved in closed form as a single sparse linear system eliminating entirely the need to relaxation iterations.

6.1. Closed Form of Local Step

We attempt to minimize the following energy function:

$$E(R) = \min_{R_1,\ldots,R_m \in \mathbb{G}^3} \sum_{i=1}^{m} (\sum_{j \in N(i)} c_{ij} \| R_i q_{ij} \tilde{R}_i - p_{ij} \|^2 \tag{6.1}$$

$$+ \alpha A \sum_{j \in N(i)} w_{ij} \| R_i - R_j \|^2) \tag{6.2}$$

The energy to minimize in the neighborhood of point p_i is the SR-ARAP energy $E_i(R_i)$:

$$E_i(R_i) = \sum_{j \in N(i)} c_{ij} \| R_i q_{ij} \tilde{R}_i - p_{ij} \|^2 + \alpha A \sum_{j \in N(i)} w_{ij} \| R_j - R_i \|^2$$

Notice that the first term $\sum_{j \in N(i)} c_{ij} \| R_i q_{ij} \tilde{R}_i - p_{ij} \|^2$ can be written in matrix language as the quadratic form $R_i^T M_i R_i$ where M_i is a 4×4 matrix constructed from vectors q_{ij} and p_{ji} (see Section 7):

$$E_i(R_i) = R_i^T M_i R_i + \alpha A \sum_{j \in N(i)} w_{ij} \| R_j - R_i \|^2 \tag{6.3}$$

$$\tag{6.4}$$

Differentiating with respect to R_i we get:

$$\frac{\partial E_i(R_i)}{\partial R_i} = 2 M_i R_i + 2 \alpha A \sum_{j \in N(i)} w_{ij} (R_i - R_j) \tag{6.5}$$

$$= M_i R_i + \alpha A R_i - \alpha A \sum_{j \in N(i)} w_{ij} R_j \tag{6.6}$$

Setting the partial derivatives to zero $\frac{\partial E_i(R_i)}{\partial R_i} = 0$

$$(M_i + \alpha A I) R_i = \alpha A \sum_{j \in N(i)} w_{ij} R_j \tag{6.7}$$

$$\tag{6.8}$$

Which is a linear system of equations. The system 5.7 can be solved for all rotors in closed form. Writting 5.7 in matrix form we get:

$$\mathrm{M\ R} = \mathrm{W\ R} \tag{6.9}$$

$$(\mathrm{M} - \mathrm{W})\mathrm{R} = 0 \tag{6.10}$$

Where M is a symmetric sparse matrix with dimensions $4n \times 4n$ which is stacking M_i at its diagonal, W is a discrete Laplacian matrix with dimensions $4n \times 4n$ derived from RHS of 5.7 i.e., $\sum_{j \in N(i)} \alpha A w_{ij} R_j$ holding the coefficients $\alpha A w_{ij}$. R is column matrix of dimensions $1 \times 4n$ stacking the coefficients of rotors R_i.

The linear system can be solved my imposing rotor constraints R_i^{const} which corresponds to best rotation of constrained points p_i^{const}:

$$\begin{aligned} (\mathrm{M} - \mathrm{W})\mathrm{R} &= 0 \\ s.t.\ R_i &= R_i^{const} \end{aligned} \tag{6.11}$$

Since the constraint $R_i^T R_i = 1$ is not honored by the linear system the rotor R_i given as solution must be normalized. As shown in [6] the solution of 5.7 gives a approximate solution to the optimal rotor. Our results confirm that the proposed linearization is not affecting accuracy in any significant way.

6.2. Rotation constraints R_i^{const}

The rotation constraints R_i^{const} in 5.11 have to be computed from equation 5.7 for each constrained point p_i^{const}:

$$R_i^{const} = (M_i + \alpha AI)^{-1} \sum_{j \in N(i)} \alpha A w_{ij} R_j^{prev}$$

where neighbor rotations R_j^{prev} are known from previous iteration. So computing constraint rotors amounts to solve a small 4×4 linear system. The resulting rotor R_i^{const} must be normalized.

6.3. Rotation's Feedback

The linear system of 5.11 find rotors from scratch i.e., it doesn't take into account the previous state of the rotors. That might lead to undesired results in an animation sequence. The sense of the rotations found by 5.11 for some animation step are not always respecting the sense of previous rotations. The abrupt change in the sense of rotation cause animation artifacts specially for large rotations.

The equation 5.11 allow us to introduce *feedback* rotors to the RHS which acts as *hints* to find rotors close to the ones in previous iteration. Given the rotor R_i^{prev} and a positive small scalar value ϵ_i we augment equation 5.7 in the following way:

$$(M_i + \alpha AI)R_i = \alpha A \sum_{j \in N(i)} w_{ij} R_j + \epsilon R_i^{prev} \tag{6.12}$$

where $\sum_{j\in N(i)} w_{ij} = 1 - \epsilon$. Our intention is to make R_i a neighbor of itself, in some sense. The strength of feedback is given by ϵ. So the global system is changed as follows:

$$\begin{aligned}(\mathrm{M} - \mathrm{W})\mathrm{R} &= \epsilon \mathrm{R}^{prev} \\ s.t.\ R_i &= R_i^{const}\end{aligned} \tag{6.13}$$

where R^{prev} is a $1 \times 4n$ column vector stacking all the unconstrained rotors R_i^{prev} from previous iteration.

7. Multiresolution Optimization

For achieving real-time performance we optimize the SR-ARAP energy in a low resolution version of the input mesh and then we transfer that solution to the full resolution mesh. To obtain the low resolution mesh we simplify the mesh using *half edge collapses* (i.e., the simplified mesh is a triangulation of a subset of the original vertices) while minimizing the Quadrics error metric. After we obtained the optimal deformed shape on the simplified mesh we transfer the optimized rotations to the full resolution mesh using Harmonic Interpolation of rotors (see Section 6.1) and then solve the following linear system using the optimized vertices of the low resolution mesh as positional constraints:

$$\sum_{j\in N(i)} c_{ij}(p_j - p_i) = \sum_{j\in N(i)} \frac{c_{ij}}{2}(R_i q_{ij} \tilde{R}_i + R_j q_{ij} \tilde{R}_j)$$

7.1. Harmonic Interpolation of Rotors

The seminal work of Pinkall and Polthier [5] shows how to interpolate given data over a discrete domain using harmonic maps. Harmonic maps are critical points of the Dirichlet energy (stretching energy) $\int_\Omega |\nabla f|^2 dA$, which generates minimal surfaces [5]. The *discrete harmonic energy* [5] of a map f defined on mesh vertices $\{p_i\}$ has the form:

$$E(f) = \sum_{i,j} c_{ij} \|f(p_j) - f(p_i)\|^2 \tag{7.1}$$

where w_{ij} are the (symmetric) cotangent weights [5] defined on triangle edges going from p_i to p_j. The discrete Laplacian operator can be identified in that energy as:

$$\Delta f = \sum_{j\in N_i} c_{ij}(f(p_j) - f(p_i)) \tag{7.2}$$

where N_i is the set of indices of the neighbors of vertex p_i. It is known that harmonic energy minimizes angular distortions. That means that harmonic functions smoothly blend boundary conditions over the domain. Harmonic functions are intrinsic to surfaces and independent of the discretization used to produce meshes. In spirit similar to [7], we propose the interpolation of

rotors over a mesh using harmonic functions. The harmonic rotor interpolation over a surface mesh can be formulated as the solution to the following discrete harmonic equation:

$$\sum_{j \in N(i)} c_{ij}(R_j - R_i) = 0 \tag{7.3}$$

subject to Dirichlet boundary conditions $R_k = R_k^{const}$, where $\{R_k^{const}\}$ are the optimized rotors from the lower resolution mesh and $\{c_{ij}\}$ are the cotangent weights. This leads to the solution of a sparse linear matrix system. This interpolation produces a smooth field of rotors by "averaging" the boundary conditions gradually over the surface. This is in effect equivalent to a linear interpolation of boundary conditions across the surface. It can be written in matrix form as:

$$\begin{aligned} \mathrm{L\ R} &= 0 \\ s.t.\ R_k &= R_k^{const} \end{aligned} \tag{7.4}$$

where L is discrete Laplacian operator used to solve the global step.

8. The form of M_i

In this section we primarily work with bivectos instead of vectors for the sake of mathematical convenience as the geometric product of bivectors contains other bivector. Due to mathematical (and geometric) duality of vectors and bivectors in $\mathbb{G}_3$ our formulation is also valid for vectors. Let q_{ij} be a bivector defined as $q_{ij} = (q_j - q_i) \cdot \tilde{I}$ and p_{ij} be a bivector defined as $p_{ij} = (p_j - p_i) \cdot \tilde{I}$ where $\tilde{I}$ is the reversed pseudoscalar defined as $\tilde{I} = e_{321}$.

Notice that the membrane term $\sum_j \|R_i q_{ij} \tilde{R}_i - p_{ij}\|^2$ is equivalent to $\sum_j \|R_i q_{ij} - p_{ij} R_i\|^2$ under the L2 norm. Also notice that $R_i q_{ij} - p_{ij} R_i$ can be rewriten as $(w_i + B_i)q_{ij} - p_{ij}(w_i + B_i)$ for some a scalar w_i and bivector B_i such that $R_i = w_i + B_i$. Expanding the geometric product of bivectors in terms of the inner product and the communtator product we get:

$$(q_{ij} - p_{ij})w_i - (q_{ij} - p_{ij}) \cdot B_i - (q_{ij} + p_{ij}) \times B_i \tag{8.1}$$

This geometric algebra expression above can be defined in matrix language following the matrix system $M_{ij} R_i$:

$$M_{ij} R_i = \begin{bmatrix} 0 & -d_{ij}^T \\ d_{ij} & [s_{ij}]_\times^T \end{bmatrix} \begin{bmatrix} w_i \\ B_i \end{bmatrix} = \begin{bmatrix} -d_{ij}^T B_i \\ w_i d_{ij} - s_{ij} \times B_i \end{bmatrix} \tag{8.2}$$

$$d_{ij} = q_{ij} - p_{ij} \quad s_{ij} = p_{ij} + q_{ij}$$

where d_{ij} and s_{ij} are 3×1 column vectors holding bivector's coefficients, M_{ij} is a skew-symmetric 4×4 real matrix, so that $M_{ij}^T = -M_{ij}$. The rotor R_i is represented as 4×1 column vector made of the scalar w_i and the 3×1 column vector B_i holding the bivector's components. The 3×3 matrix $[s_{ij}]_\times$ is representing the skew-symmetric cross-product matrix as usually defined for vectors in $\mathbb{R}^3$.

We can express $\sum_j \|R_i q_{ij} - p_{ji} R_i\|^2$ as the quadratic form $R_i^T M_i R_i$ where $M_i = \sum_j^n c_{ij} M_{ij}^T M_{ij}$. Note that since M_{ij} is skew-symmetric, the product $M_{ij}^T M_{ij}$ is symmetric and positive semi-definite. Consequently the matrix M_i is also symmetric positive semi-definite. It follows that all eigenvalues of M_i are real and $\lambda_i \geq 0$.

$$M_{ij}^T M_{ij} = \begin{bmatrix} \|d_{ij}\|^2 & (s_{ij} \times d_{ij})^T \\ s_{ij} \times d_{ij} & d_{ij} d_{ij}^T - [s_j]_\times^2 \end{bmatrix} \tag{8.3}$$
$$d_{ij} = q_{ij} - p_{ij} \quad s_{ij} = p_{ij} + q_{ij}$$

By the spectral theorem the minimizer of $R_i^T M_i R_i$ subject to $R_i^T R_i = 1$ is the eigenvector of M_i associated with the smallest eigenvalue which is a positive number. This fact can be used to robustly solve the best alignment of vectors a.k.a. the Wahba problem.

8.1. Efficient Computation of M_i

The symmetric matrix $M_{ij}^T M_{ij}$ has a simple form:

$$M_{ij}^T M_{ij} = \begin{bmatrix} \|d_{ij}\|^2 & (s_{ij} \times d_{ij})^T \\ s_{ij} \times d_{ij} & d_{ij} d_{ij}^T - s_{ij} s_{ij}^T + \|s_{ij}\|^2 I_{3\times 3} \end{bmatrix}$$
$$d_{ij} = q_{ij} - p_{ij} \quad s_{ij} = p_{ij} + q_{ij}$$

Writing it in terms of p_{ij} and q_{ij} we get:

$$M_{ij}^T M_{ij} = 2 \begin{bmatrix} q_{ij}^T p_{ij} & (q_{ij} \times p_{ij})^T \\ q_{ij} \times p_{ij} & q_{ij} p_{ij}^T + p_{ij} q_{ij}^T - q_{ij}^T p_{ij} I_{3\times 3} \end{bmatrix} \tag{8.4}$$
$$+(\|p_{ij}\|^2 + \|q_{ij}\|^2) I_{4\times 4}$$

All terms of 7.4 can be derived from the dyadic tensor $q_{ij} p_{ij}^T$ plus the quantity $\|p_{ij}\|^2 + \|q_{ij}\|^2$. Since matrix $q_{ij} p_{ij}^T$ is of 3×3 its computation is efficient.

9. Implementation

The author's source code is publicly available at github [4].

10. Results

11. Conclusion

References

[1] L. Dorst, D. Fontijne, and S. Mann. *Geometric Algebra for Computer Science.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[2] Z. Levi and C. Gotsman. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE transactions on visualization and computer graphics*, 21(2):264—277, February 2015.

[3] M. C. Lopez Belon. Applications of conformal geometric algebra in mesh deformation. In *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI - Conference on*, pages 39–46, Aug 2013.

[4] M. C. Lopez Belon. Project on github. Available at https://github.com/mauriciocele/arap-sr-linearized, 2020.

[5] U. Pinkall, S. D. Juni, and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[6] J. Wu, M. Lopez, M. Liu, and Y. Zhu. Linear geometric algebra rotor estimator for efficient mesh deformation. *IET Cyber-systems and Robotics*, 2(2):88–95, 2020.

[7] R. Zayer, C. Rössl, Z. Karni, and H. peter Seidel. Harmonic guidance for surface deformation. In *In Proc. of Eurographics 05*, pages 601–609, 2005.

Mauricio Cele Lopez Belon
Madrid, Spain
e-mail: `mclopez@outlook.com`