IET Research Journals

Submission Template for IET Research Journal Papers

IET Journals
The Institution of Engineering and Technology

A Linear Geometric Algebra Rotor Estimator for Efficient Mesh Deformation

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Jin Wu[1], *Mauricio Lopez*[2*], *Ming Liu*[1], *Yilong Zhu*[1]

[1] *Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China*
[2] *Buenos Aires Computer Corporation, Argentina*
J. Wu and M. Lopez contributed equally to the content of this paper.

Abstract: We solve the problem of estimating the best rotation aligning two sets of corresponding vectors (also known as Wahba's problem or point cloud registration). The proposed method is among the fastest methods reported in recent literatures, moreover it is robust to noise, accurate and simpler than most other methods. It is based on solving the linear equations derived from the formulation of the problem in Euclidean Geometric Algebra. We show its efficiency in two applications: the As-Rigid-As-Possible (ARAP) Surface Modeling and the more Smooth Rotation enhanced As-Rigid-As-Possible (SR-ARAP) mesh animation which is the only method capable of deforming surface modes with quality of tetrahedral models. Mesh deformation is a key technique in games, automated construction and robotics. The ARAP technique along with its improved variants, although have been extensively studied, can still not be achieved efficiently. Linear geometric algebra based rotor solution proposed in this paper gives another perspective of the kernel problem. This, however, not only improves the real performance of the 3-dimensional mesh deformation, but also provides a brand new computationally efficient solution to the Wahba's problem and point cloud registration, which has been closely related to the automation science and engineering.

1 Introduction

The concept of mesh deformation arises from the emerging needs of the highly dynamic 3-dimensional (3-D) computer animation, which has been extensively employed in computer games, visualization, computer aided design (CAD) and intelligent manufacturing [1]. Another similar terminology i.e. the surface flattening has also been widely studied in automated construction [2]. In principle, the target of these approaches is using 2-D meshes to construct specific 3-D models. As complex 3-D models own quite abundant surface details, the deformation process is time consuming. The As-Rigid-As-Possible (ARAP, [3]) models such process by introducing various local rigid transformations. This determines that the mathematical model of ARAP is similar with the form of point-point matching, i.e. a classical problem aligning two vector sets (may be of different numbers of points) [4]. This problem is renowned as the Wahba's problem [5] and point cloud registration [6] in aerospace engineering and robotics, respectively. Since the proposal of Wahba's problem in 1965, many robust and computationally efficient algorithms have been proposed. Early efforts have been paid by Shuster, Markley, Mortari who developed different solvers based on various attitude representations like quaternion, rotation matrix, Rodrigues vector and etc. [7–9]. Recently, Wu et al. studied fast results for online determination of attitude using quaternion formulation of the Wahba's problem [10, 11]. Related applications like registration using iterative closest point (ICP) also highly relies on the efficiency of matching [12].

Historically, representative solutions mainly cover the single-point batch processing of point-set alignment. However, it is required in many applications that the solution should be either accurate, robust or continuous, so that the animated meshes can be smoothly placed. A recent method using the gradient descent algorithm (GDA) of the Wahba's problem solves this problem but is not linear [13]. It has also been proposed in [14] that some sub-optimal filters can produce continuous results but the accuracy is not optimal. The target of ARAP-based mesh deformation is similar to Wahba's problem but as the amount of meshes are usually huge. Therefore the practitioners require a new method that is not only accurate, robust, continuous but also, ARAP-friendly. Guided by this requirement, in the paper, the main contributions are:

1.A geometric algebra based formulation of the kernel problem in ARAP has been proposed, which is intuitive, linear and simple.
2.It is able for us to construct continuous Newton algorithm to converge to the ideal solution within only several simple iterations.
3.ARAP and its improved variants are combined with the proposed method in a friendly manner allowing real-time performance of accurate mesh animation.

We thus study the analytical forms of all the operations required. Through derivations we are able to give linear solutions with simple matrix manipulations. It has been compared with representatives of Wahba's problem and point cloud registration and has shown that it owns the same accuracy but less computational burden. The performance of the mesh animation also proves that the accuracy belongs to the optimal ones while the computational efficiency is the highest.

This paper is then structured as: Section II provides the details of the proposed derivations and solution. Section III investigates the application of the method to the ARAP and its variants. Section IV gives experimental results and comparisons and the concluding remarks are drawn in Section V.

2 Fast Rotor Estimation Algorithm

2.1 Preliminaries

All the rotation matrices belong to the space of special orthogonal groups, i.e. for 3×3 case, the 3-dimensional $SO(3) := \left\{\boldsymbol{R}|\boldsymbol{R} \in \mathbb{R}^{3\times3}, \boldsymbol{R}^\top\boldsymbol{R} = \boldsymbol{I}, \det(\boldsymbol{R}) = 1\right\}$. The geometric algebra allows for the geometric product of two 3×1 vectors a and b such that

$$ab = a \cdot b + a \times b \tag{1}$$

in which $\cdot$ and $\times$ are inner (dot) and outer (cross) products. Note that the outer product $a \times b = \mathcal{B}$ forms a bivector $\mathcal{B}$ also in the 3-dimensional geometric algebra space $\mathbb{G}_3$. This further gives a normalized version within the scope of the even subalgebra $\mathbb{G}_3^+$, which produces a group of rotor (spin) vectors

$$SP(3) = \left\{R|R \in \mathbb{G}_3^+, R\tilde{R} = \tilde{R}R = 1\right\} \tag{2}$$

where $\tilde{R} = b \times a$ gives the reverse of R supposing that $R = a \times b$. We use $< R >_k$ to extract the k-th element of R.

2.2 Proposed Work

Given two sets of n points in $P = \{p_j\}_{j=1}^n, Q = \{q_j\}_{j=1}^n$ we try to minimize the following error:

$$E(R) = \min_{R \in SP(3)} \sum_{j=1}^{n} c_j \|q_j - R p_i \tilde{R}\|^2 \tag{3}$$

where $\{c_j\}_{j=1}^n$ are scalar weights such that $\sum_{j=1}^n c_j = 1$. In that form, the minimization is nonlinear in R. Following some manipulations, we can make it linear by multiplying by R on the right

$$q_j R - R p_j = 0 \tag{4}$$

so the constraint $R\tilde{R} = 1$ is implicit in the energy $E(R)$, although we will need to project R back to the manifold through normalization. Let F be a column vector of functions which square amounts to the energy $E(R)$.

$$F = \begin{bmatrix} F^1 \\ \vdots \\ F^n \end{bmatrix} \tag{5}$$

The energy $E(R)$ can be expressed as the matrix product:

$$E(R) = F^\top F \tag{6}$$

where:

$$F^j(R) = \sqrt{c_j}(R p_j - q_j R) \tag{7}$$

F^j is producing a multivector with the basis $\{e_1, e_2, e_3, e_{123}\}$.

The critical points of the energy $E(R)$ where it is minimized can be found solving $\nabla E(R) = 0$. Where the gradient has the following form:

$$g = \nabla E(R) = \nabla(F^\top F) = 2J^\top F \tag{8}$$

where J is the Jacobian matrix of F. Since the energy $E(R)$ is purely quadratic in R, the solution for $\nabla E(R) = 0$ can be found by solving a linear system of equations. An optimal rotor is in the null space of a particular matrix derived from $J^\top F = 0$. An easy way to obtain a solution in the null-space is using one iteration of Newton's method. Due to linearity of the equation $J^\top F = 0$ w.r.t. R one iteration suffice for obtaining a solution. The optimal increment for $E(R)$ is given by the Newton formula $\Delta R = H^{-1}\nabla E$, provided that the inverse of Hessian matrix H^{-1} exists. Noting that J does not depend on R, i.e. is constant, the Hessian matrix H takes the simple form:

$$H = \frac{\partial(2J^\top F)}{\partial R} = 2J^\top \frac{\partial F}{\partial R} = 2J^\top J \tag{9}$$

where H is independent of R. An optimal solution can then be found by solving a linear system.

$$R^* = R_0 - H^{-1} g(R_0) \tag{10}$$

Where R_0 is an initial rotor and $g(R_0)$ is the gradient of E evaluated at R_0. Of course, the choice of initial rotor R_0 has a particular effect on finding a local minimum. The energy $E(R)$ is non-convex, its shape is similar to a sinusoidal wave with infinitely many points at minimum energy value, each one at rotor $R_i = e^{-(\theta + i\pi)B}$ for $i \in \mathbb{N}$, being B the optimal unique attitude bivector and θ an optimal angle with minimal absolute value. The choice of R_0 leads to find a local solution close to it, the choice $R_0 = [1, 0, 0, 0]^\top$ is optimal from the computational point of view and is also desirable because lead us to find solutions close to the *identity* rotor. Since there are infinite solutions, H is a singular matrix. A simple and efficient way to find its pseudo-inverse is to add a very small regularization term to $E(R)$. Our strategy is to introduce a new constant rotor R_i within a new term $\epsilon\|R - R_i\|^2$, which we can express as matrix product $F_2^\top F_2$:

$$F_2 = \sqrt{\epsilon}(R - R_i) \tag{11}$$

with Jacobian

$$J_2 = \frac{\partial F_2}{\partial R} = \sqrt{\epsilon} I \tag{12}$$

where I is the 4×4 Identity matrix. The energy now looks like this:

$$E(R) = \min_{R \in SP(3)} \sum_{j=1}^{n} c_j \|R p_j - q_j R\|^2 + \epsilon \|R - R_i\|^2 \tag{13}$$

where $0 < \epsilon < 1$ is constant but small (we use $\epsilon = 10^{-6}$). The term $\epsilon\|R - R_i\|^2$ is a regularization term that helps on finding a solution *biased* towards R_i, so one can think R_i as a constant rotor which is pointing R toward a given minimum point. With this regularization strategy we can not only find the pseudo-inverse of H but also we can make sure to obtain a stable R that is not stuck when the minimal point is close to some far angle such $\pm\pi$ radians. For instance by taking R_i as the previous rotor calculated with Newton iteration, we can check that R is at some minimum point, because in the next iteration we should obtain the same R. In the edge cases near to $\pm\pi$ where the R may not be at minimum, the previous solution R_i will direct the next solution to a minimum, commonly in one to three iterations. So regularization term can be seen as a feedback for reaching a stable minimal R. We will see this more clearly in the algorithm. Keep in mind that we are solving the linear system just once (i.e., we are inverting H once) because the R_i term is affecting only the gradient. The gradient and Hessian are now:

$$g = \nabla(F^\top F + F_2^\top F_2) = 2(J^\top F + J_2^\top F_2) \tag{14}$$

$$H = 2\frac{\partial(J^\top F + J_2^\top F_2)}{\partial R} = 2(J^\top J + \epsilon I) \tag{15}$$

The optimal rotor is as before:

$$R^* = R_0 - H^{-1} g(R_0) \tag{16}$$

The Hessian matrix H resembles now a form of *Tikhonov* regularization, whose inverse H^{-1} approaches to the *Moore-Penrose* pseudo-inverse as ϵ approaches to zero. We now need to determine how to obtain R_i. An *stand alone* algorithm can take R_i as the previous rotor calculated with Newton iteration. But we also can take R_i to be a previous solution in a simulation. We have used both choices in our experiments, with excellent results in both cases. We noticed in a simulation that the previous rotor helps to get the next one preserving the same sense of rotation. We now proceed to lie down our algorithm taking R_i from previous rotor calculated with Newton iteration.

2.2.1 Optimal Computation of Jacobians:

Lets recall that each term of the sum $\sum_{j=1}^n c_j \|R p_j - q_j R\|^2$ can be expressed in matrix form as function $F^{jT} F^j$ where:

$$F^j(R) = \sqrt{c_j}(R p_j - q_j R) \tag{17}$$

F^j is producing a multivector with the basis $\{e_1, e_2, e_3, e_{123}\}$. Jacobian of F^j is $\frac{\partial F^j}{\partial R}$:

$$J^j = \begin{bmatrix} \frac{\partial F^j}{\partial w} & \frac{\partial F^j}{\partial e_{12}} & \frac{\partial F^j}{\partial e_{13}} & \frac{\partial F^j}{\partial e_{23}} \end{bmatrix} \tag{18}$$

Solution to $H\Delta R = g(R_0)$ looks like this:

$$\left(\sum_{j=1}^{n} J^{jT} J^j\right) \Delta R = -\sum_{j=1}^{n} J^{jT} F^j$$
$$R^* = R_0 + \Delta R \tag{19}$$

$$H^j = J^{jT} J^j = c_{ij} \begin{bmatrix} D_1^2 + D_2^2 + D_3^2 & D_1S_2 - D_2S_1 & D_1S_3 - D_3S_1 & D_2S_3 - D_3S_2 \\ D_1S_2 - D_2S_1 & S_2^2 + S_1^2 + D_3^2 & S_2S_3 - D_3D_2 & D_3D_1 - S_1S_3 \\ D_1S_3 - D_3S_1 & S_2S_3 - D_3D_2 & S_3^2 + S_1^2 + D_2^2 & S_1S_2 - D_2D_1 \\ D_2S_3 - D_3S_2 & D_3D_1 - S_1S_3 & S_1S_2 - D_2D_1 & S_3^2 + S_2^2 + D_1^2 \end{bmatrix} \tag{25}$$

where we should normalize the rotor R^* for sending it back to the manifold.

2.2.2 Form of the Jacobians: The Jacobian J^j is 4×4 which four columns are the directional derivatives of $F^j = \sqrt{c_j}(Rp_j - q_jR)$

$$\frac{\partial F^j}{\partial w} = \sqrt{c_j}(p_j - q_j) = \sqrt{c_j} \begin{bmatrix} (p_j - q_j) \cdot e_1 \\ (p_j - q_j) \cdot e_2 \\ (p_j - q_j) \cdot e_3 \\ 0 \end{bmatrix} \tag{20}$$

$$\frac{\partial F^j}{\partial e_{12}} = \sqrt{c_j} \begin{bmatrix} (p_j + q_j) \cdot e_2 \\ -(p_j + q_j) \cdot e_1 \\ 0 \\ (p_j - q_j) \cdot e_3 \end{bmatrix} \tag{21}$$

$$\frac{\partial F^j}{\partial e_{13}} = \sqrt{c_j} \begin{bmatrix} (p_j + q_j) \cdot e_3 \\ 0 \\ -(p_j + q_j) \cdot e_1 \\ -(p_j - q_j) \cdot e_2 \end{bmatrix} \tag{22}$$

$$\frac{\partial F^j}{\partial e_{23}} = \sqrt{c_j} \begin{bmatrix} 0 \\ (p_j + q_j) \cdot e_3 \\ -(p_j + q_j) \cdot e_2 \\ (p_j - q_j) \cdot e_1 \end{bmatrix} \tag{23}$$

Thus the leading symmetric matrix $H^j = J^{jT} J^j$ has a simple form. The symmetric matrix $H^j = J^{jT} J^j$ has a simple form in (25) provided that

$$\begin{aligned} S_1 = (q_j + p_j) \cdot e_1 \ , \ S_2 = (q_j + p_j) \cdot e_2 \ , \ S_3 = (q_j + p_j) \cdot e_3 \\ D_1 = (p_j - q_j) \cdot e_1 \ , \ D_2 = (p_j - q_j) \cdot e_2 \ , \ D_3 = (p_j - q_j) \cdot e_3 \end{aligned} \tag{24}$$

Since H^j is symmetric only 10 out of 16 elements need to be actually computed. The system now looks like this:

$$\left(\sum_{j=1}^{n} H^j \right) \Delta R = - \sum_{j=1}^{n} J^{jT} F^j \tag{26}$$

Solving just one iteration of the Newton suffice for solving the system i.e. 19. In the iteration the symmetric matrix $H = \sum_{j=1}^{n} H^j$ is constant, but the gradient at right-hand-side depends on R since F^j depends on it.

We now can proceed to optimize a little more our method. We incorporate the fixed initial guess $R_0 = 1$ into the gradient.

$$F^j = \sqrt{c_j}(R_0 p_j - q_j R_0) F^j = \sqrt{c_j}(p_j - q_j) \tag{27}$$

$$g^j = J^{jT} F^j = c_{ij} \begin{bmatrix} D_1^2 + D_2^2 + D_3^2 \\ D_1S_2 - D_2S_1 \\ D_1S_3 - D_3S_1 \\ D_2S_3 - D_3S_2 \end{bmatrix} \tag{28}$$

Notice that the first row of H^j is equal to g^j, so g^j does not need to be explicitly calculated. Now we introduce the regularization term F_2 and proceed to replace again R_0 on it, which look like this:

$$F_2 = \sqrt{\epsilon}(R_0 - R_i) F_2 = \sqrt{\epsilon}(1 - < R >_i) \tag{29}$$

$$d = J_2^\top F_2 = \epsilon \begin{bmatrix} 1 - < R >_i \\ -R_i \cdot e_{12} \\ -R_i \cdot e_{13} \\ -R_i \cdot e_{23} \end{bmatrix} \tag{30}$$

The final iteration is reduced to:

$$\left(\sum_{j=1}^{n} H^j + \epsilon I \right) \Delta R = - \sum_{j=1}^{n} g^j - d \tag{31}$$

Notice that all terms are constant except d which depends on the previous iteration. Note that the matrix $\sum_{j=1}^{n} H^j + \epsilon I$ does not depend on R and its inverse can be precomputed. Since the matrix $(\sum_{j=1}^{n} H^j + \epsilon I)^{-1}$ and the vector $\sum_{j=1}^{n} g^j$ are constant, the inner loop amounts to compute a cheap matrix-vector multiplication. Also notice that since the matrix $\sum_{j=1}^{n} H^j$ is symmetric, only 10 components of J need to be actually computed and since the matrix $\sum_{j=1}^{n} H^j + \epsilon I$ is symmetric its inverse is symmetric and also cheap to compute.

3 Applications

3.1 ARAP

Given two meshes P and Q consisting of vertices p_i and q_i respectively, and directed edges $p_{ij} = p_j - p_i$ and $q_{ij} = q_j - q_i$, the discrete ARAP energy is defined as:

$$E(P, Q) = \sum_{i=1}^{m} \sum_{(i,j) \in \mathcal{E}_i} c_{ij} \| q_{ij} - \boldsymbol{R}_i p_{ij} \|^2 \tag{32}$$

where $\boldsymbol{R}_1, ..., \boldsymbol{R}_m \in SO(3)$ are optimal local rotations, $\mathcal{E}_1, ..., \mathcal{E}_m$ are their corresponding set of 1-ring edges and c_{ij} are weighting coefficients, typically the familiar cotangent weights.

The main idea of this method is breaking the surface into overlapping cells $\mathcal{E}_i$ 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 vertices of mesh P are in original position while vertices of mesh Q are the deformed vertices and the matrix 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 can be efficiently solved by a simple iterative method that solves two linear least squares sub-problems on each iteration.

The first step is to consider the vertices of P and Q constant and obtain the best rigid transformation R_i for each cell $\mathcal{E}_i$. The second step is to consider the rotations R_i constant and computing the optimal deformed vertices q_i in the least squares sense. This can be achieved by taking the gradient of $E(P, Q)$ w.r.t. Q and equating the result to zero, which leads to formulate a linear system of equations which can be solved by constraining the position of at least two vertices of Q. The main source of inefficiency in this method is that the first step typically involves solving a series of singular value decomposition (SVD) problems, which is slow even using the optimized solver for 3×3 matrices of McAdams *et. al.*. We show how a Geometric Algebra approach can speedup the technique. We first change

the energy $E(P,Q)$ for using rotors instead of rotation matrices:

$$E(P,Q) = \sum_{i=1}^{m} \sum_{(i,j)\in\mathcal{E}_i} c_{ij}\|q_{ij} - R_i p_{ij} \tilde{R}_i\|^2 s.t.\ R_i\tilde{R}_i = 1, \forall i \tag{33}$$

We first solve the for the best rotors incrementally first by applying the algorithm developed above. That is an incremental version of our algorithm that only require to solve a linear system for computing the next best rotor.

The second step is computing the optimal vertices q_i. Taking the partial derivatives of $E(P,Q)$ w.r.t. q_i and equating the result to zero we obtain:

$$\sum_{(i,j)\in\mathcal{E}_i} c_{ij} q_{ij} = \sum_{(i,j)\in\mathcal{E}_i} \frac{c_{ij}}{2}(R_i p_{ij} \tilde{R}_i + R_j p_{ij} \tilde{R}_j) \tag{34}$$

(34) can be expressed in matrix form as $\mathrm{L}Q = C$, where L is the symmetric Laplace-Beltrami operator, Q is the column of target positions and C is the right hand side of (34). That is, a Poisson equation. Constraints of the form $q_i = c_i$ 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:

$$(\mathrm{L}^\top \mathrm{L})\, Q = \mathrm{L}^\top\, C \tag{35}$$

Mesh deformations are achieved by repositioning the constrained vertices $q_i = c_i$, solving the linear subproblem for rotors R_i, updating the right-hand-side of (34) and solving the LLS system for q_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 performance improvements over the ARAP Surface Modeling technique are significant as can be seen in the comparison. 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.

3.2 SR-ARAP Energy

The smooth-rotation ARAP (SR-ARAP, [15]) energy for a smooth map between two 2-manifolds $f : P \mapsto Q$ is:

$$E_{SR}(f) = \int_P \min_{\boldsymbol{R}\in SO(3)} (\|\mathrm{d}f - \boldsymbol{R}\|_F^2 + \alpha\hat{A}\|\mathrm{d}\boldsymbol{R}\|_F^2) \tag{36}$$

The first term in the integral is a membrane energy, and the second term is a bending energy that penalizes the difference between rotations. $\alpha\hat{A}$ is a weighting scalar, where $\hat{A}$ is the area of the whole surface. Normally, the differential of a mapping of a 2-manifold is a 2×2 matrix, which maps tangent vectors from the parametric domain to a tangent plane at a surface point. Here, df is a 3×3 matrix, which maps the 3D embedding of the tangent vectors from one surface to another (which is simply the Jacobian matrix of $f : \mathbb{R}^3 \mapsto \mathbb{R}^3$). The discretization is:

$$E_{SR}(P,Q) = \min_{\boldsymbol{R}_1,\ldots,\boldsymbol{R}_m\in SO(3)} \sum_{i=1}^{m} (\sum_{(i,j)\in\mathcal{E}_i} c_{ij}\|q_{ij} - \boldsymbol{R}_i p_{ij}\|^2 + \alpha\hat{A} \sum_{(i,j)\in\mathcal{E}_i} w_{ij}\|\boldsymbol{R}_i - \boldsymbol{R}_j\|_F^2) \tag{37}$$

where $\boldsymbol{R}_1, ..., \boldsymbol{R}_m \in SO(3)$ are optimal local rotations associated with the vertices; $\mathcal{E}_i$ is the set of 1-ring edges which are neighbors of i-th vertex; w_{ij} are scalar weights; and $\hat{A}$ is the triangle mesh area, which is used to make the energy scale invariant. Here the regularization is conducted in the form of the chordal distance $\|\boldsymbol{R}_i - \boldsymbol{R}_j\|_F^2$ between $\boldsymbol{R}_i$ and $\boldsymbol{R}_j$. The first term, the membrane energy, is similar to the ARAP discretization, and the second term, the bending energy, penalizes the difference between an edge set rotation and the rotations of neighboring edge sets. The objective of the membrane term is to lower the distortion of an edge set (resists stretching and shearing), by keeping the map differential close to rigid. The objective of the bending term is to keep the variation in the rotations in an edge set neighborhood low, such that the neighborhood would transform as a unit, as much as possible.

3.3 GA SR-ARAP Energy

The geometric-algebra SR-ARAP (GA SR-ARAP) energy is shown as follows

$$E_{SR}(P,Q) = \min_{R_1,\ldots,R_m\in SP(3)} \sum_{i=1}^{m} (\sum_{(i,j)\in\mathcal{E}_i} c_{ij}\|q_{ij} - R_i p_{ij} \tilde{R}_i\|^2 + \alpha\hat{A} \sum_{(i,j)\in\mathcal{E}_i} w_{ij}\|R_i - R_j\|^2) \tag{38}$$

Notice that we are expressing rotations using Rotors instead of 3×3 rotation matrices, also notice that in the second term, the bending energy, we are using the Euclidean (L^2) norm instead of the Frobenius norm. Those changes are not affecting the resulting surface deformation in any noticeable way but are very convenient since the resulting optimization problem can be solved with a much more efficient algorithm as we will show.

3.4 Local Step

In the local step we consider vertices of P and Q as constants and solve for the best rotation of each cell $\mathcal{E}_i$ independently by minimizing:

$$\begin{aligned} &\arg\min_{R_i\in SP(3)} \sum_{(i,j)\in\mathcal{E}_i} c_{ij}\|R_i p_{ij} \tilde{R}_i - q_{ij}\|^2 \\ &+\alpha\hat{A} \sum_{(i,j)\in\mathcal{E}_i} w_{ij}\|R_i - R_j\|^2, \quad s.t.\ R_i\tilde{R}_i = 1 \end{aligned} \tag{39}$$

Like in the Wahba problem, in the first term the minimization is nonlinear in R_i. Following Perwass, we can make it linear by multiplying by R_i on the right.

$$\begin{aligned} q_{ij}R_i - R_i p_{ij}\tilde{R}_i R_i = 0 \\ q_{ij}R_i - R_i p_{ij} = 0 \end{aligned} \tag{40}$$

This amounts to minimize:

$$E(R_i) = \sum_{(i,j)\in\mathcal{E}_i} c_{ij}\|q_{ij}R_i - R_i p_{ij}\|^2 + \alpha\hat{A} \sum_{(i,j)\in\mathcal{E}_i} w_{ij}\|R_i - R_j\|^2 \tag{41}$$

where the constraint $R_i\tilde{R}_i = 1$ is now implicit, although we will need to project R_i back to the rotor manifold through normalization. Let $F(R)$ be the vector of functions which square equals to the energy $E(R) = F^\top F$. The column vector F can be split in two blocks so that:

$$F = \left[\ (F_1^1)^\top, \cdots, (F_1^1)^\top, (F_2^1)^\top, \cdots, (F_2^n)^\top\ \right]^\top \tag{42}$$

where n is the degree of the i-th vertex i.e., $n = |\mathcal{E}_i|$. The energy $E(R)$ can be expressed as the matrix product:

$$E(R) = F^\top F \tag{43}$$

where $F_1^j(R_i) = \sqrt{c_{ij}}(R_i p_{ij} - q_{ij}R_i)$
$F_2^j(R_i) = \sqrt{w_{ij}}(R_i - R_j)$ in which the factor w_{ij} is absorbing

$$H^j = J_1^{jT} J_1^j = c_{ij} \begin{bmatrix} D_1^2 + D_2^2 + D_3^2 & D_1S_2 - D_2S_1 & D_1S_3 - D_3S_1 & D_2S_3 - D_3S_2 \\ D_1S_2 - D_2S_1 & S_2^2 + S_1^2 + D_3^2 & S_2S_3 - D_3D_2 & D_3D_1 - S_1S_3 \\ D_1S_3 - D_3S_1 & S_2S_3 - D_3D_2 & S_3^2 + S_1^2 + D_2^2 & S_1S_2 - D_2D_1 \\ D_2S_3 - D_3S_2 & D_3D_1 - S_1S_3 & S_1S_2 - D_2D_1 & S_3^2 + S_2^2 + D_1^2 \end{bmatrix} \tag{49}$$

the whole factor $\frac{\alpha\hat{A}}{n} w_{ij}$ for the sake of notation simplicity. F_1^j is producing a multivector with the basis $\{e_1, e_2, e_3, e_{123}\}$ and F_2^j is producing a rotor in the basis $\{1, e_{12}, e_{13}, e_{23}\}$. The critical points of $E(R)$ are solutions of the equation $\nabla E(R) = 0$

$$\begin{aligned} \nabla E(R) &= \frac{\partial (F^\top F)}{\partial R} \\ &= 2\sum_{j=1}^{n} J_1^{jT} F_1^j + 2\sum_{j=1}^{n} J_2^{jT} F_2^j \end{aligned} \tag{44}$$

Since J_1 and J_2 are constant, the equation $\nabla E(R) = 0$ is linear in R and the solution amounts to solve the null-space of a linear system. As before, we can solve it using only one iteration of Newton method. The Newton increment is given by $\Delta R = -H^{-1}\nabla E(R)$. The Hessian matrix H is:

$$\begin{aligned} H &= 2\frac{\partial (\sum_{j=1}^{n} J_1^{jT} F_1^j + \sum_{j=1}^{n} J_2^{jT} F_2^j)}{\partial R} \\ &= 2\sum_{j=1}^{n} J_1^{jT} J_1^j + 2\sum_{j=1}^{n} J_2^{jT} J_2^j \end{aligned} \tag{45}$$

So the full system looks like this:

$$\begin{aligned} \left(\sum_{j=1}^{n} J_1^{jT} J_1^j + \sum_{j=1}^{n} J_2^{jT} J_2^j \right) \Delta R &= -\sum_{j=1}^{n} J_1^{jT} F_1^j - \sum_{j=1}^{n} J_2^{jT} F_2^j \\ R^* &= R_0 + \Delta R \end{aligned} \tag{46}$$

For some initial guess R_0, that we will choose to be $R_0 = 1$ as in the previous section. The rest of the work focuses on the computation of Jacobian and Hessian which is in the similar form shown in aformentioned contents. Here we define

$$g^j = J_1^{jT} F_1^j = c_{ij} \begin{bmatrix} D_1^2 + D_2^2 + D_3^2 \\ D_1S_2 - D_2S_1 \\ D_1S_3 - D_3S_1 \\ D_2S_3 - D_3S_2 \end{bmatrix} \tag{47}$$

$$d^j = J_2^{jT} F_2^j = w_{ij} \begin{bmatrix} 1 - < R_j >_0 \\ -R_j \cdot e_{12} \\ -R_j \cdot e_{13} \\ -R_j \cdot e_{23} \end{bmatrix} \tag{48}$$

Notice that the first row of H^j is equal to g^j, so g^j does not need to be calculated. Now the right-hand-side is does not depend on R_i. The final system to solve is:

$$\begin{aligned} \Delta R &= -\left(\sum_{j=1}^{n} H^j + \alpha\hat{A}I \right)^{-1} (\sum_{j=1}^{n} (g^j + d^j)) \\ R^* &= R_0 + \Delta R \end{aligned} \tag{50}$$

3.5 Global Step

The global step is computing the optimal vertices $\{q_i\} \in Q$.

$$\begin{aligned} E_{SR}(P, Q) = \min_{R_1,\ldots,R_m \in SP(3)} \sum_{i=1}^{m} (& \sum_{(i,j)\in\mathcal{E}_i} c_{ij} \|q_{ij} - R_i p_{ij} \tilde{R}_i\|^2 \\ & + \alpha\hat{A} \sum_{(i,j)\in\mathcal{E}_i} w_{ij} \|R_i - R_j\|^2) \end{aligned} \tag{51}$$

Taking the partial derivatives of $E_{SR}(P, Q)$ w.r.t. q_i and equating the result to zero lead us to obtain the linear system of (34):

$$\sum_{(i,j)\in\mathcal{E}_i} c_{ij} q_{ij} = \sum_{(i,j)\in\mathcal{E}_i} \frac{c_{ij}}{2} (R_i p_{ij} \tilde{R}_i + R_j p_{ij} \tilde{R}_j) \tag{52}$$

which can be expressed in matrix form as $\mathrm{L}Q = C$, where L is the symmetric Laplace-Beltrami operator, Q is the column of target positions and C is the right hand side of (34). Constraints of the form $q_i = c_i$ 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 solution is $(\mathrm{L}^\top \mathrm{L})\, Q = \mathrm{L}^\top\, C$, as described previously.

3.5.1 Local Relaxation: In the SR-ARAP method, mesh deformations are obtaining by repositioning the constrained vertices $q_i = c_i$, solving *local step* subproblem for rotors R_i, updating the right-hand-side of (34) and solving the *global step* subproblem system for q_i. Unlike the ARAP surface, the optimized rotations in the *local step* for SR-ARAP are codependent. Since we optimize each rotation independently, while fixing the others, the local step can be considered as a *relaxation*, thus more than one local iteration should be executed before performing the *global step*. At least two steps of local relaxations should be done for each global iteration, although for better convergence we should perform more local relaxations for each global iteration.

3.6 Real-time SR-ARAP

The local relaxation used to be a serious bottleneck for the SR-ARAP method. However our fast rotor estimation SR-ARAP can be used to compute as many local relaxation step as needed at the same cost as the simple ARAP method (i.e., without slowing down the solver). Also we show how real-time performance can be achieved on SR-ARAP by adding a simple multiresolution step.

3.6.1 Rotor Relaxation: Recall that rotor estimation for SR-ARAP amounts to solve one linear system:

$$R^* = R_0 - H^{-1}(g + d) \tag{53}$$

where $H = \left(\sum_{j=1}^{n} H^j + \alpha\hat{A}I\right), g = \sum_{j=1}^{n} g^j$ and $d = \sum_{j=1}^{n} d^j$. The only term involving the neighbor rotations is $\sum_{j=1}^{n} d^j$. So for the entire *local relaxation* we can precompute H^{-1} and g. So the computational cost of solving each relaxation step amounts to do a matrix multiplication.

3.6.2 Multiresolution SR-ARAP: 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 and solve the linear system of (34) using as position constraints the optimized vertices of the low resolution mesh.

The transference of rotations from the low-res mesh to the high-res mesh is equivalent to clustering rotations in the high resolution mesh. The clustering of rotations is based on the connectivity graph of the simplified mesh. Having the a injective map $f : Q_{low} \to Q_{hi}$ that maps rotations from low-res mesh Q_{low} to hi-res mesh Q_{hi} we generate the surjective map $c : Q_{hi} \to Q_{low}$ which is mapping several (equal) rotations of Q_{hi} to one corresponding rotation of Q_{low}. Our algorithm to construct the surjection c is a simple diffusion process: We start by mapping the rotations from low-res mesh to hi-res mesh using the injection f, then we iteratively copy the assigned rotations to the neighboring vertices that still does not have a rotation, repeating the process until there are no vertices with neighbors without a rotation assigned. That process is used to precompute the surjection $c : Q_{hi} \to Q_{lqw}$ which is used later for efficient transferring of rotations. Let f^{-1} denote the inverse image of the injection f. So the range of f^{-1} is the entire set of vertices in Q_{low} and the domain of f^{-1} is the image of f.

4 Experimental results

In this section, we use some datasets to conduct registration and mesh deformation using our proposed method and other existing algorithms. In the first study, we show the accuracy consistency of the proposed method with representative methods. We use the following model to generate the point cloud data:

$$\mathcal{L} = \sum_{i=1}^{n} \|\boldsymbol{b}_i - \boldsymbol{R}\boldsymbol{r}_i\|^2 \tag{54}$$

in which $\boldsymbol{r}_i$ is the reference vector and $\boldsymbol{b}_i$ stand for the measurement vector in the body frame. We use the ground truth data of $\boldsymbol{R}$ and reference data as shown in Table III in [10]. Here various algorithms for registration are employed for assembled test, they are SVD method by Horn [16], Fast Linear Attitude Estimator (FLAE) [10], GDA [13], Fast Symbolic 3D Registration (FS3R) [11], Fast Analytical 3D Registration (FA3R) [17]. Each of them has been proven to be accurate, robust and computationally efficient. We evaluate various representatives using the loss function value $\mathcal{L}$ and computation time on a personal computer of MacBook Pro 2017 with i7-CPU and MATLAB environment. The required parameters of these algorithms are:

- Proposed: relative stop criteria 1×10^{-12}.
- SVD: no parameter.
- FLAE: no parameter.
- GDA: descent step length 1×10^{-3}; relative stop criteria 1×10^{-12}; maximum iteration number 100.
- FS3R: singularity check factor: 1×10^{-5}.
- FA3R: relative stop criteria 1×10^{-12}.

With 12 classical cases in [10], we are able to generate the results shown in Table 1. One may observe that the accuracy of compared methods are almost identical. The reason is that these solutions are all based on thee framework of (54) so the resulted analytical formulations achieve equivalent mathematical errors.

The developed method has been combined with ARAP and its variants in the aforementioned contents. Based on the energy functions defined using the geometric algebra, we are able to implement the proposed algorithm. Here, the `libigl-2.1.0` library [18, 19] has been invoked for high-efficiency implementation of the mesh deformation via parallel computation through the `BLASX` library [20]. We use the `decimated-knight` and `cactus` models to evaluate the various ARAP variants under the kernel energy of the proposed geometric-algebra solution.

The model of knight contains 1500 vertices and simulated control points are generated using ground truth rotation. Fig. 1 shows the refined ARAP results of various algorithms. Gradient descent is introduced for classical implementation where the descent factor is set to 5×10^{-2}. Seen from the figure, it is observed that the geometric algebra (GA) supervised ARAP can achieve quite accurate deformation after 20 iterations. However, the GDA according to its nature of convergence problem, can not fully converge to the ground truth value. That is to say in engineering, we may need more iterations or improved Newton methods to converge to a satisfactory result. However, we need to note that this is not computationally efficient and thus should be compared with the proposed GA algorithm.

The `cactus` model is employed for mesh deformation and the result of GA SR-ARAP is shown in Fig. 3. We can see that under high noise level the algorithm is still capable of achieving good deformation results. This indicates the accuracy of the GA-based point cloud registration. Now we need to compare the computational efficiency of various algorithms. All these algorithms are then implemented in an unoptimized (compilation level) manner guaranteeing the fairness of all the run-time results. The termination criteria of all the algorithms is that the relative difference of the minimized energy of two successive iterations reaches 1×10^{-8}. The details of the execution time for the cactus model with different numbers of vertices are summarized in Table 2. The GA-based ARAP and variants are more time-efficient than other representatives. We need to point out that although FLAE is analytical and computationally more efficient than the iterative GA algorithm proposed in this paper, it is not so friendly combing with ARAP. Therefore, the convergence of the FLAE-based ARAP will require more computational burden.

Table 1 RMSE of Wahba's Loss Functions $\mathcal{L}$ Using Various Methods

Case	Proposed	SVD Horn 1987 [16]	FLAE 2018 [10]	GDA 2018 [13]	FS3R 2019 [11]	FA3R 2020 [17]
1	4.9709×10^{-13}	4.9709×10^{-13}	4.9709×10^{-13}	4.9709×10^{-13}	4.9709×10^{-13}	4.9709×10^{-13}
2	2.4983×10^{-13}	2.4983×10^{-13}	2.4983×10^{-13}	2.4983×10^{-13}	2.4983×10^{-13}	2.4983×10^{-13}
3	5.0554×10^{-05}	5.0554×10^{-05}	5.0554×10^{-05}	5.0554×10^{-05}	5.0554×10^{-05}	5.0554×10^{-05}
4	2.4957×10^{-05}	2.4957×10^{-05}	2.4957×10^{-05}	2.4957×10^{-05}	2.4957×10^{-05}	2.4957×10^{-05}
5	5.0278×10^{-13}	5.0278×10^{-13}	7.1999×10^{-10}	5.0278×10^{-13}	5.0444×10^{-13}	7.1999×10^{-10}
6	4.9635×10^{-13}	4.9635×10^{-13}	4.9635×10^{-13}	4.9635×10^{-13}	4.9635×10^{-13}	4.9635×10^{-13}
7	2.4841×10^{-13}	2.4841×10^{-13}	2.4841×10^{-13}	2.4841×10^{-13}	2.4841×10^{-13}	2.4841×10^{-13}
8	4.7623×10^{-05}	4.7623×10^{-05}	4.7623×10^{-05}	4.7623×10^{-05}	4.7623×10^{-05}	4.7623×10^{-05}
9	2.5273×10^{-05}	2.5273×10^{-05}	2.5273×10^{-05}	2.5273×10^{-05}	2.5273×10^{-05}	2.5273×10^{-05}
10	1.5007×10^{-12}	1.5007×10^{-12}	1.7529×10^{-12}	1.5007×10^{-12}	1.5007×10^{-12}	1.7529×10^{-12}
11	4.9458×10^{-13}	4.9458×10^{-13}	9.5140×10^{-13}	4.9458×10^{-13}	4.9471×10^{-13}	9.5140×10^{-13}
12	5.0559×10^{-13}	5.0559×10^{-13}	3.4190×10^{-11}	5.0559×10^{-13}	7.5501×10^{-11}	3.4190×10^{-11}

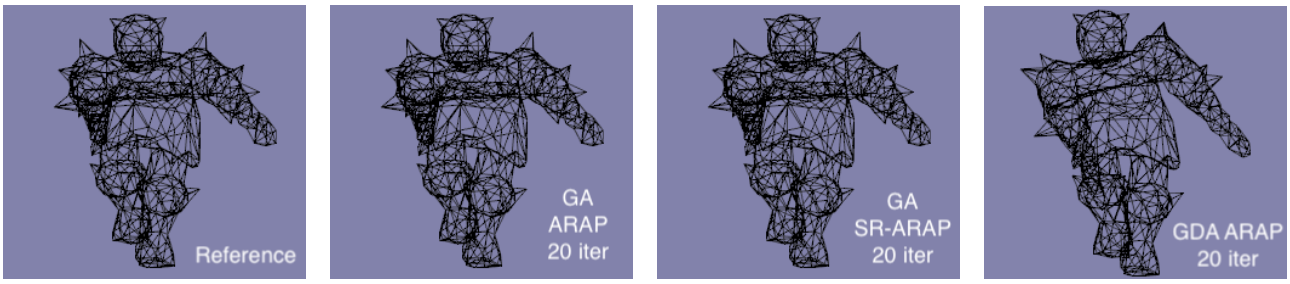

Fig. 1: Knight deformation: Left: reference model of the knight; Middle: deformation using the proposed method combining with ARAP; Right: deformation using the GDA [13] combining with ARAP.

Table 2 Mesh Deformation Computation Time Using Various Methods

Case	ARAP	SR-ARAP	GA-ARAP	GA SR-ARAP	GDA ARAP	FLAE ARAP
100	1.234960×10^{-02}s	1.369739×10^{-01}s	$\mathbf{7.786490 \times 10^{-03}}$s	$\mathbf{2.870024 \times 10^{-02}}$s	8.915073×10^{-01}s	8.915073×10^{-02}s
199	3.714611×10^{-02}s	9.279175×10^{-02}s	$\mathbf{1.155189 \times 10^{-02}}$s	$\mathbf{1.613069 \times 10^{-02}}$s	$1.144537 \times 10^{+00}$s	1.144537×10^{-01}s
398	5.528854×10^{-02}s	1.070500×10^{-01}s	$\mathbf{2.611447 \times 10^{-02}}$s	$\mathbf{3.654026 \times 10^{-02}}$s	$1.647663 \times 10^{+00}$s	1.647663×10^{-01}s
794	1.889311×10^{-01}s	4.307448×10^{-01}s	$\mathbf{5.891055 \times 10^{-02}}$s	$\mathbf{1.047712 \times 10^{-01}}$s	$6.138786 \times 10^{+00}$s	6.138786×10^{-01}s
1584	1.846733×10^{-01}s	3.815289×10^{-01}s	$\mathbf{6.808507 \times 10^{-02}}$s	$\mathbf{8.913603 \times 10^{-02}}$s	$5.572538 \times 10^{+00}$s	5.572538×10^{-01}s
3162	3.787262×10^{-01}s	7.035758×10^{-01}s	$\mathbf{1.392477 \times 10^{-01}}$s	$\mathbf{1.800131 \times 10^{-01}}$s	$1.169642 \times 10^{+01}$s	$1.169642 \times 10^{+00}$s
6309	5.088200×10^{-01}s	8.757848×10^{-01}s	$\mathbf{1.776170 \times 10^{-01}}$s	$\mathbf{2.162810 \times 10^{-01}}$s	$1.457190 \times 10^{+01}$s	$1.457190 \times 10^{+00}$s

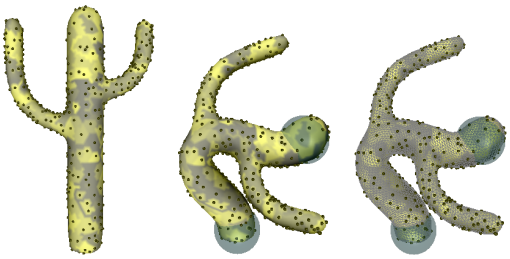

Fig. 3: GA SR-ARAP clusters: Left: Hi-res cactus model with Low-res points marked. Clusters formed around vertices are colored with same intensity. Middle and Right: deformation of Low-res model transferred to Hi-res model.

For illustration of the proposed method for large numbers of meshes, the Armadillo model has been chosen which contains over 43000 vertices and 80000 edges [21]. For ARAP, it is hard to implement real-time animation of the Armadillo model for satisfactory update frequency. Using the proposed GA-ARAP, it is able to speed up the mesh deformation to a large extent. Fig. 2 shows the details of the reference and the deformation result from the proposed GA-ARAP. The developed algorithm maintains good accuracy and can achieve very high visualizatoin frequency of 60fps on a typical personal computer with i7-4core CPU. For conventional ARAP, since the computational burden is relatively higher, it can only reach the update rate of 12fps at most. The good computational behavior of the proposed method provides the possibility of performing highly real-time mesh deformation such as games and high-speed visualization.

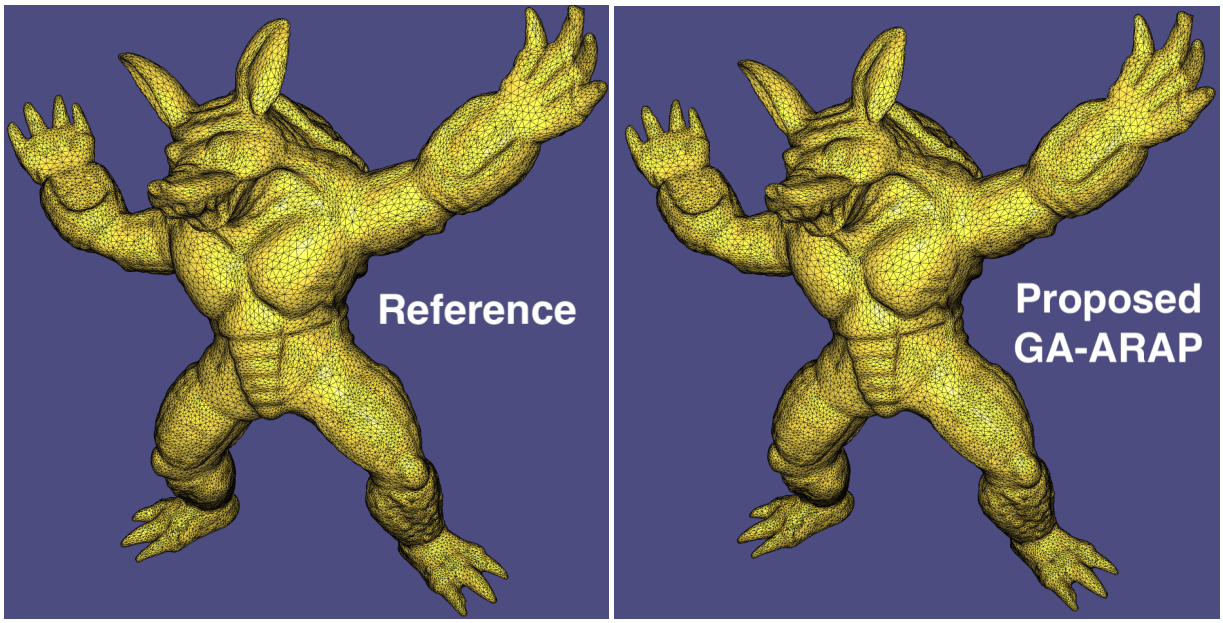

Fig. 2: Armadillo deformation: Left: Reference shape; Right: Shape from Proposed GA-ARAP after 10 iterations.

5 Conclusion

In this paper, we solve the problem of estimating the best rotation for the alignment of two sets of corresponding 3D points. It is based on solving the linear equations derived from the formulation of the problem in Euclidean Geometric Algebra. The method is fast, robust to noise, accurate, simple and GPU friendly. Applications and experimental validation of mesh deformation are presented for description of the performance of the proposed algorithm. The results show that the slightly losing accuracy of the proposed method leads to advance in execution time consumption. The designed algorithms are ARAP-friendly and may be helpful for real-time mesh deformation in related applications.

6 References

1 Yi, R., Wu, C., Liu, Y.J., He, Y., Wang, C.C.L.: 'Delta DLP 3-D Printing of Large Models', *IEEE Trans Autom Sci Eng*, 2018, **15**, (3), pp. 1193–1204

2 Zhang, Y., Wang, C.C.L.: 'WireWarping++: Robust and flexible surface flattening with length control', *IEEE Trans Autom Sci Eng*, 2011, **8**, (1), pp. 205–215

3 Sorkine, O., Alexa, M. 'As-rigid-as-possible surface modeling'. In: Proceedings of the fifth Eurographics symposium on Geometry processing. SGP '07. (Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007. pp. 109–116

4 Arun, K.S., Huang, T.S., Blostein, S.D.: 'Least-Squares Fitting of Two 3-D Point Sets', *IEEE Trans Pattern Anal Mach Intell*, 1987, **PAMI-9**, (5), pp. 698–700

5 Wahba, G.: 'A Least Squares Estimate of Satellite Attitude', *SIAM Rev*, 1965, **7**, (3), pp. 409

6 Besl, P.J., McKay, N.D.: 'A Method for Registration of 3-D Shapes', *IEEE Trans Pattern Anal Mach Intell*, 1992, **14**, (2), pp. 239–256

7 Shuster, M.D., Oh, S.D.: 'Three-axis attitude determination from vector observations', *J Guid Control Dyn*, 1981, **4**, (1), pp. 70–77

8 Markley, F.L.: 'Attitude Determination using Vector Observations and the Singular Value Decomposition', *J Astronaut Sci*, 1988, **36**, (3), pp. 245–258

9 Mortari, D., Markley, F.L., Singla, P.: 'Optimal linear attitude estimator', *J Guid Control Dyn*, 2007, **30**, (6), pp. 1619–1627

10 Wu, J., Zhou, Z., Gao, B., Li, R., Cheng, Y., Fourati, H.: 'Fast Linear Quaternion Attitude Estimator Using Vector Observations', *IEEE Trans Autom Sci Eng*, 2018, **15**, (1), pp. 307–319

11 Wu, J., Liu, M., Zhou, Z., Li, R.: 'Fast Symbolic 3D Registration Solution', *IEEE Trans Autom Sci Eng*, 2018, **PP**, pp. 1–10

12 Ying, S., Peng, J., Du, S., Qiao, H.: 'A scale stretch method based on ICP for 3D data registration', *IEEE Trans Autom Sci Eng*, 2009, **6**, (3), pp. 559–565

13 Wu, J., Zhou, Z., Fourati, H., Li, R., Liu, M.: 'Generalized Linear Quaternion Complementary Filter for Attitude Estimation From Multisensor Observations: An Optimization Approach', *IEEE Trans Autom Sci Eng*, 2019, **16**, (3), pp. 1330–1343

14 Wu, J., Zhou, Z., Fourati, H., Liu, M.: 'Recursive linear continuous quaternion attitude estimator from vector observations', *IET Radar, Sonar Navig*, 2018, **12**, (11), pp. 1196–1207

15 Levi, Z., Gotsman, C.: 'Smooth Rotation Enhanced As-Rigid-As-Possible Mesh Animation', *IEEE Trans Visuliz Comput Graph*, 2014, **21**, (2), pp. 264–277

16 Horn, B.K.P.: 'Closed-form solution of absolute orientation using unit quaternions', *J Optic Soc America A*, 1987, **4**, (4), pp. 629–642

17 Wu, J.: 'Rigid 3D Registration: A Simple Free of SVD and Eigen-decomposition', *IEEE Trans Instrum Meas*, 2020,

18 Jacobson, A., Panozzo, D. 'libigl: prototyping geometry processing research in c++'. In: SIGGRAPH Asia 2017 courses. (, 2017. pp. 1–172

19 Jacobson, A., Panozzo, D., Schüller, C., Diamanti, O., Zhou, Q., Pietroni, N., et al.. 'libigl: A simple c++ geometry processing library'. (Retrieved 2017-10-18 from http://libigl. github. io/libigl, 2016

20 Wang, L., Wu, W., Xu, Z., Xiao, J., Yang, Y. 'BLASX: A High Performance Level-3 BLAS Library for Heterogeneous Multi-GPU Computing'. In: Proceedings of the 2016 International Conference on Supercomputing. (, 2016. pp. 1–11

21 Lavoué, G.: 'A roughness measure for 3D mesh visual masking', *ACM Int Conf Proceeding Ser*, 2007, **253**, pp. 57–60