A High-Performance Singularity-Free Rotation Estimator in Geometric Algebra

Mauricio Cele Lopez Belon

Abstract. Well known methods for solving the Wahba's problem are formulated in the linear algebra framework, mainly using the SVD, or in quaternion's framework which leads to solve a max-eigenvalue problem. Those methods are based on well stablished numerical algorithms which on the other hand are iterative and slow. Recently, closed form solutions for computing the optimal quaternion has been proposed base on solving the roots of the quartic polynomial associated with the eigenvalue problem with an analytic formula. Those methods are fast but they have singularities i.e., they completely fail on certain input data. In this paper we propose a high-performance, singularity-free estimator of the best quaternion aligning two sets of corresponding vectors i.e., equivalent to Wahba's problem. It is based on a new formulation of the problem in the Geometric Algebra representation of rotations which is isomorphic to Hamilton's quaternions. The proposed method converge in a fixed number of iterations being is as fast as the closed form solutions reported in literature and at the same time is free of singularities, robust to noise and simple to implement.

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

Keywords. Geometric Algebra, Rotation Estimation, Wahba Problem.

1. Introduction

Wahba's problem has been studied for over half a century since 1965 [12]. The problem looks for the optimal rotation between two sets of corresponding vectors. Many effective methods have been developed to solving the problem [1,4,6,9,15] mainly by formulating it in the quaternions framework, which lead us to solve an eigenvalue problem, or by formulating it in the linear algebra framework and using the Singular Value Decomposition (SVD) and recently by formulating it in the Geometric Algebra framework [3, 8] which at some point casts the solution into linear algebra framework and SVD due to the lack of geometric algebra numerical algorithms. More recently, closed form solutions for computing the optimal quaternion has been proposed [?, ?, 14] based on solving the roots of the quartic polynomial associated with the eigenvalue problem with an analytic formula.

Accuracy and speed of most methods have been compared in previous works [14] where the trade-off between performance and robustness is valorated. In particular SVD based methods exhibit very good accuracy but low performance and quaternion based methods are in the other end depending on the implementation. Regarding the later methods, the proposed closed form solutions exhibit the best trade-off so far but they have singularities i.e., they completely fail on certain input data.

In this paper we propose a high-performance, singularity-free estimator of the best quaternion aligning two sets of corresponding vectors i.e., equivalent to Wahba's problem. It is based on a new formulation of the problem in the Geometric Algebra representation of rotations which is isomorphic to Hamilton's quaternions. The proposed method converge in a fixed number of iterations being is as fast as the closed form solutions reported in literature and at the same time is free of singularities, robust to noise and simple to implement.

In this work we propose a high-performance, singuarity-free method for solving the Wahba's problem. It is based on a maximizing a convex quadratic energy functional, which is equivalent to Wahba's problem, formulated in geometric algebra framework $\mathbb{G}_3$. The proposed method converges in a fixed number of iterations being is as fast as the closed form solutions reported in literature and at the same time is free of singularities and simple to implement.

Applications of Wahba's solutions are diverse from aerospace engineering computation of spacecraft attitude [14] to mesh deformation in computer graphics [10, 11] for accurate motion construction and restoration [7, 8]. An important characteristic of our method is that it is based almost entirely on symmetric-matrix multiplication so it can be hardware-accelerated using vector instructions or graphics hardware without effort. In engineering fields requiring high-performance vector alignment our method can lower down the time by taking advantage of the AVX (Advanced Vector Extensions) support available in modern microprocessors. Modern matrix packages such as Eigen [?] can produce apropriate AVX instructions automatically. In computer graphics applications our method can lower down the time even more by taking advantage of GPU vector instructions obtaining a performance boost not present in other numerical algorithms.

Few work has been done on studying this problem using geometric algebra. Geometric algebra rotors are closely related to quaternions (quaternion algebra can be regarded as a geometric algebra defined on a set of imaginary basis vectors) we find geometric algebra to be a more natural choice for studying this problem since it is defined over a Euclidean vector space $\mathbb{R}^3$, where original data is defined. In contrast to previous works we base our formulation on bivectos instead of vectors for the sake of mathematical convenience. Due to mathematical duality of vector and bivectors in $\mathbb{G}_3$ our formulation is also valid for vectors. The implementation of our algorithm does not require any geometric algebra library, we present an implementation based on standard matrix and quaternion library.

This paper is arranged as follows: Section II introduces the geometric algebra $\mathbb{G}_3$, Section III includes the presentation of our fast rotor estimation algorithm. Section IV demonstrates the experimental results.

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

A geometric algebra $\mathbb{G}_3$ 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 so that the square of any vector is a scalar $aa = a^2 \in \mathbb{R}$. From the vector space $\mathbb{R}^3$, the geometric product generates the geometric algebra $\mathbb{G}_3$ with elements $\{X, R, A...\}$ 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, 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. 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:

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

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

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 $B = \frac{a\wedge b}{\|a\wedge b\|}$.

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

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. Geometric Algebra Rotor Estimation

Given two sets of n corresponding bivectors $P = \{p_j\}_{j=1}^n$ and $Q = \{q_j\}_{j=1}^n$, we attempt to minimize the following error function:

$$E(R) = \min_{R\in\mathbb{G}_3^+} \sum_j c_j \|q_j - Rp_i\tilde{R}\|^2$$
$$s.t. \; R\tilde{R} = 1$$

where $\{c_j\}_{j=1}^n$ are scalar weights such that $\sum_j^n c_j = 1$. In that form, the minimization is a non-linear least squares problem with a non-linear constraint in R. Notice that the error term $q_j - Rp_i\tilde{R}$ is equivalent to $q_jR - Rp_j$ by multiplying by R on the right and using the fact that $R\tilde{R} = 1$. The equivalent problem is:

$$E(R) = \min_{R\in\mathbb{G}_3^+} \sum_j c_j \|p_jR - Rq_j\|^2$$
$$s.t. \; R\tilde{R} = 1$$

which is a linear least squares problem with a non-linear constraint in R.

We define the commutator product of two bivectors p_j and q_j as $p_j \times q_j = \frac{1}{2}(p_jq_j - q_jp_j)$. The commutator product of bivectors in $\mathbb{G}_3$ can be interpreted as a cross-product of bivectors in the sense that the resulting bivector $B = p_j \times q_j$ is orthogonal to both p_j and q_j. The commutator product allow us to define the geometric product of two bivectors as $AB = A \cdot B + A \times B$. Recall that the inner product of bivectors is not the same as the inner product of vectors, since the square of bivectors is negative, the inner product of bivectors has the opposite sign e.g., $(ae_{12} + be_{13} + ce_{23}) \cdot (de_{12} + ee_{13} + fe_{23}) = -ad - be - cf$.

For a pair of unit bivectors, A and B, the rotor R aligning them can be defined as $R = A\frac{(A+B)}{\|A+B\|}$. Defining the unit bivector $C = \frac{A+B}{\|A+B\|}$ the rotor R is also $R = AC = w + L$ where $w = A \cdot C = \cos(\theta/2)$ and $L = A \times C = \sin(\theta/2)\frac{A\times B}{\|A\times B\|}$.

The constraint $p_jR - Rq_j = 0$ can be rewriten as $p_j(w+L) - (w+L)q_j = (p_j - q_j) \cdot (w + L) + (p_j + q_j) \times L$. Since the inner product $(p_j - q_j) \cdot L = 0$, the constraint could be simplified as follows for a pair of vectors:

$$(p_j + q_j) \times L = -w(p_j - q_j)$$

However the above constraint holds only when the input data is not noisy. For our purposes keeping the term $(p_j - q_j) \cdot L$ aids the optimization on enforcing orthogonality while finding a minimum possible value:

$$(p_j - q_j) \cdot (w + L) + (p_j + q_j) \times L = 0$$

In matrix language we can define the following matrix system $M_j R = 0$:

$$\begin{bmatrix} 0 & -d_j^T \\ d_j & [s_j]_\times \end{bmatrix} \begin{bmatrix} w \\ L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$d_j = p_j - q_j \quad s_j = p_j + q_j$$

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

Let define the function F^j representing the linear transformation:

$$F^j = \sqrt{c_j} M_j R$$

Let F be a column vector of n stacked functions F^j

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

such that the energy $E(R)$ can be expressed as matrix product

$$E(R) = F^T F = \begin{bmatrix} F^{1T} & \cdots & F^{nT} \end{bmatrix} \begin{bmatrix} F^1 \\ \vdots \\ F^n \end{bmatrix}$$

We can express $E(R)$ as the following quadratic form:

$$E(R) = \min_R R^T M R$$
$$s.t.\ R^T R = 1$$

where $M = \sum_j^n c_j M_j^T M_j$. Note that since M_j is skew-symmetric, the product $M_j^T M_j$ is symmetric and positive semi-definite. Consequently the matrix M is also symmetric positive semi-definite. It follows that all eigenvalues of M are real and $\lambda_i \geq 0$.

$$M_j^T M_j = \begin{bmatrix} \|d_j\|^2 & (d_j \times s_j)^T \\ d_j \times s_j & d_j d_j^T - [s_j]_\times^2 \end{bmatrix}$$
$$d_j = p_j - q_j \quad s_j = p_j + q_j$$

Using the spectral theorem it is easy to show that the minimizer of $E(R)$ is the eigenvector of M associated with the minimum eigenvalue.

The equation $M_j R = 0$ holds when the rotor R aligns perfectly the input bivectors i.e., R is in the null space of M_j and the matrix $M_j^T M_j$ is singular as at least one eigenvalue is zero $\lambda_{\min} = 0$ (two eigenvalues will be zero for a single pair of corresponding bivectors). That implies that when all input bivectors can be aligned perfectly the energy $E(R)$ is non strictly convex since it has infinitely many points at minimum energy value i.e., the solution is in general non-unique. That is problematic since we are interested in finding an eigenvector with least eigenvalue. For instance to use the Inverse Power Method we would need to find the pseudo-inverse of M which need the use of SVD, or the use of Tikhonov regularization:

$$R_{i+1} = \epsilon(M + \epsilon I)^{-1} R_i$$

Where ϵ is a small real value $0 < \epsilon < 1$. The use of Tikhonov regularization makes M symmetric positive-definite matrix, making the energy $E(R)$ strictly convex. Note that the Hessian matrix of $E^j = R^T M_j^T M_j R$ is $H^j = 2M_j^T M_j$, so the Hessian matrix $H = \sum_j M_j^T M_j$ of $E(R)$ is equal to the matrix M. The Tikhonov regularization makes the Hessian matrix positive-definite.

However, instead of rely on the Inverse Power Method and the need of finding the approximate inverse of M we reformulate the optimization problem to the equivalent maximization:

$$E(R) = \max_{R \in \mathbb{G}_3^+} \sum_j c_j \|p_j + \tilde{R} q_i R\|^2$$
$$s.t. \; R\tilde{R} = 1$$

That is a non-linear quadratic maximization problem with a non-linear constraint. Notice that the term $\|p_j + \tilde{R} q_i R\|^2$ is dual to $\|q_j - R p_j \tilde{R}\|^2$ in the sense that the optimal R makes the first maximum at $(\|p_j\| + \|q_j\|)^2$ and the second minimal at 0. The equivalent problem is:

$$E(R) = \max_{R \in \mathbb{G}_3^+} \sum_j c_j \|R p_j + q_j R\|^2$$
$$s.t. \; R\tilde{R} = 1$$

The constraint $R p_j + q_j R$ can be rewriten as $(w + L) p_j + q_j (w + L)$.

$$w(p_j + q_j) + L \cdot (p_j + q_j) + (q_j - p_j) \times L$$

In matrix language we can define the following matrix system:

$$M_j R = \begin{bmatrix} 0 & -s_j^T \\ s_j & [d_j]_\times \end{bmatrix} \begin{bmatrix} w \\ L \end{bmatrix} = \begin{bmatrix} -s_j^T L \\ w s_j + d_j \times L \end{bmatrix}$$
$$d_j = q_j - p_j \quad s_j = p_j + q_j$$

M_j is a skew-symmetric 4×4 real matrix. We can then express $E(R)$ as the following quadratic form:

$$E(R) = \max_R R^T (\sum_j^n c_j M_j^T M_j) R$$
$$s.t.\ R^T R = 1$$

Note that since M_j is skew-symmetric, the product $M_j^T M_j$ is symmetric and positive semi-definite. Consequently the matrix M is also symmetric positive semi-definite i.e., $R^T M R \geq 0$ and all eigenvalues of M are real and $\lambda_i \geq 0$.

$$M_j^T M_j = \begin{bmatrix} \|s_j\|^2 & (s_j \times d_j)^T \\ s_j \times d_j & s_j s_j^T - [d_j]_\times^2 \end{bmatrix}$$
$$d_j = q_j - p_j \quad s_j = p_j + q_j$$

By the spectral theorem the maximizer of $E(R)$ is the eigenvector of M associated with the largest eigenvalue which is a positive number. Typically it can be obtained with great precision using SVD or QR factorization. Nevertheless, in this work we seek for an efficient solution which allow us to take advantage of nowadays computing hardware.

4. Convexity

The convexity of the energy $E(R)$ can be proof by showing that its Hessian matrix of second partial derivatives is positive semi-definite. The Hessian matrix of $E(R)$ is $\frac{\partial^2 E(R)}{\partial R} = \sum_j^n c_j M_j^T M_j$ which is symmetric, moreover since M_j is skew-symmetric matrix, the product $M_j^T M_j$ is symmetric positive semi-definite. Then follows that $\sum_j^n c_j M_j^T M_j$ is positive semi-definite and therefore convex, provided that the sum of weights is convex i.e., $\sum_j^n c_j = 1$ and $\forall j, c_j \geq 0$.

5. Fast Geometric Algebra Rotor Estimation

The most time consuming task of the estimation is to compute the eigenvector of M associated with the greatest eigenvalue. Sound numerical algorithms for computing eigenvectors of a real symmetric matrix are SVD, QR factorization and Jacobi iteration among others. However those algorithms finds all eigenvectors of a matrix. Since we are interested in finding an eigenvector with greatest eigenvalue, our best choice is to use the Power Iteration. Which have the following form:

$$R_{i+1} = \frac{MR_i}{\|MR_i\|}$$

It has been established that convergence of power iteration is linear or *geometric*, depending on the ratio of the largest eigenvalue to the second eigenvalue $\lambda_2/\lambda_{\max}$ i.e., the method converges slowly if there is an eigenvalue close in magnitude to the largest eigenvalue. A well-known algorithm for finding dominant eigenvectors of matrices consists of repeatedly squaring a matrix, renormalizing the matrix after each squaring. See [13].

$$M_{i+1} = \frac{M_i M_i}{\|M_i\|_F}$$

In [5] it is showed that the convergence of the repeated squaring is *quadratic* and for real symmetric $N \times N$ matrices the average number of iterations required is $\mathrm{O}(\ln N + \ln \|\ln \epsilon\|)$. Moreover, the following upper bound is provided $\frac{1}{\ln 2}[\ln(3\sqrt{2}(N(N-1)/2+N)) + 1 + \ln(2\ln N + \|\ln \epsilon\|)] + 1$. Thus for a symmetric 4×4 matrix, taking a small $\epsilon = 10^{-13}$ the average number of iterations is 6 and upper bound is 12. That can be exploited for creating deterministic algorithms.

Several mathematicians have proved bounds on eigenvalues in terms of the value of the matrix. A simple test in [2], is based on the L1-norm of the columns nd rows. Let a_{ij} be the elements, $R_i = \sum_j |a_{ij}|$ be the L1-norm of the ith row and $C_j = \sum_i |a_{ij}|$ the L1-norm of the jth column. Let $\mathcal{R}$ be the largest R_i and $\mathcal{C}$ the largest C_j. Then for all eigenvalues $|\lambda| \leq \min(\mathcal{R}, \mathcal{C})$. Since our matrix is symmetric the upper bound is the same for rows and columns. Going back to the power method, we see that after a small number of repeated matrix squaring, the dominant eigenvector correspond to the jth column having the largest C_j value.

6. Optimal Computation of M

The symmetric matrix $M^j = M_j^T M_j$ has a simple form:

$$M_j^T M_j = \begin{bmatrix} \|s_j\|^2 & (s_j \times d_j)^T \\ s_j \times d_j & s_j s_j^T - d_j d_j^T + \|d_j\|^2 I \end{bmatrix}$$
$$d_j = q_j - p_j \quad s_j = p_j + q_j$$

In terms of p_j and q_j it is:

$$M_j^T M_j = 2\begin{bmatrix} p_j^T q_j & (p_j \times q_j)^T \\ p_j \times q_j & p_j q_j^T + q_j p_j^T - p_j^T q_j I_{3\times3} \end{bmatrix} + (\|p_j\|^2 + \|q_j\|^2) I_{4\times4}$$

All terms of the matrix $M_j^T M_j$ can be derived from the matrix $B = p_j q_j^T$ plus the quantiry $\|p_j\|^2 + \|q_j\|^2$. Since $M_j^T M_j$ is symmetric, only 10 out of

16 elements need to be actually computed. Notice that the trace of $M_j^T M_j$ is $Tr(M_j^T M_j) = 4\|p_j\|^2 + 4\|q_j\|^2$ and $Tr(H) = 4\sum_j \left(\|p_j\|^2 + \|q_j\|^2\right)$.

7. Algorithms

Our algorithm is based on the repeated squaring method, which is an iterative algorithm, although it always performs a fixed number of matrix squaring operations. Since the upper bound of iterations has been established and numerical tests confirms that no more iterations are needed to achieve the required accuracy. In contrast to the power iteration our method doesn't need a random initial guess i.e., the best R is always calculated by matrix squaring.

The proposed method is shown in Algorithm 1.

Algorithm 1 Fast Rotor Estimation

Require: $P = \{p_j\}_{j=1}^n, Q = \{q_j\}_{j=1}^n, C = \{c_j\}_{j=1}^n$

for $j = 1$ **to** n **do**

$\quad S = c_j(p_j + q_j)$

$\quad B = B + c_j p_j q_j^T$

end for

$$H = \begin{bmatrix} 0 & B_{12} - B_{21} & B_{20} - B_{02} & B_{01} - B_{10} \\ B_{12} - B_{21} & 0 & B_{01} + B_{10} & B_{20} + B_{02} \\ B_{20} - B_{02} & B_{01} + B_{10} & 0 & B_{12} + B_{21} \\ B_{01} - B_{10} & B_{20} + B_{02} & B_{12} + B_{21} & 0 \end{bmatrix}$$

$H_{00} = S^T S + Tr(B)$

$H_{11} = 2B_{00} + S^T S - Tr(B)$

$H_{22} = 2B_{11} + S^T S - Tr(B)$

$H_{33} = 2B_{22} + S^T S - Tr(B)$

for $i = 1$ **to** 12 **do**

$\quad H^{i+1} = \frac{H^i H^i}{Tr(H^i)}$

end for

$\mathcal{C}_j = H_{j0}, \forall j = 0, ..., 3$

$R = normalize(\mathcal{C}_j)$

return $R(0) + R(1)e_{12} + R(2)e_{13} + R(3)e_{23}$

The C++ code using the Eigen library can be found in Listing 1.

8. Comparisons

We select several representative methods e.g. FLAE [14], SVD [4] and QUEST [9] to implement the algorithm. The Eigen library is employed to implement the SVD and QUEST. Another version of QUEST using the Newton iteration is also added for comparison. The tests are ran on a MacBook Pro 13' 2017 computer with the CPU of Intel 3.1GHz 4-core i5. The Visual Studio 2019 C++ compiler is used in our experiments.

9. Conclusion

In this paper, we solve the problem of estimating the best rotation for the alignment of two sets of corresponding 3D vectors. It is based on solving the linear equations derived from the formulation of the problem in Geometric Algebra. The method is fast, robust to noise, accurate and simpler than most other methods. Experimental validation of the performance of the proposed algorithm is presented. The results show that the slightly losing accuracy of the proposed method is well worth the huge advance in execution time consumption. For many applications the vector datasets are very huge i.e. the computation time would be more important than slight change of accuracy. Therefore, we hope that the proposed algorithm would be of benefit to such applications.

10. C++ code

Listing 1. C++ code for rotor estimation

```
Quaterniond GAFastRotorEstimator(
    const vector<Vector3d>& P,
    const vector<Vector3d>& Q,
    const vector<double>& w)
{
    Matrix4d H;
    Matrix3d B;
    Vector3d S;
    const size_t N = P.size();
    const size_t MAX_STEPS = 12;
    B.setZero();
    S.setZero();
    for (size_t j = 0; j < N; ++j) {
        S.noalias() += w[j] * (P[j] + Q[j]);
        B.noalias() += (w[j] * P[j]) * Q[j].transpose();
    }
    double diag = S.dot(S);
    H(3, 3) = diag + B.trace();
    diag = diag - B.trace();
    H(3, 0) = (B(1, 2) - B(2, 1));
    H(3, 1) = (B(2, 0) - B(0, 2));
    H(3, 2) = (B(0, 1) - B(1, 0));
    H(0, 0) = 2.0 * B(0, 0) + diag;
    H(1, 0) = (B(0, 1) + B(1, 0));
    H(2, 0) = (B(2, 0) + B(0, 2));
    H(1, 1) = 2.0 * B(1, 1) + diag;
    H(2, 1) = (B(1, 2) + B(2, 1));
    H(2, 2) = 2.0 * B(2, 2) + diag;
    H.selfadjointView<Eigen::Lower>().evalTo(H);
    for (size_t j = 0; j < MAX_STEPS; ++j) {
        H *= H;
        H *= 1.0 / H.trace();
    }
    return Quaterniond(H.col(0).normalized());
}
```

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, Sep. 1987.

[2] A. Brauer. Limits for the characteristic roots of a matrix. *Duke Mathematical Journal*, 13(3):387–395, 09 1946.

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

[4] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[5] E. Kostlan. Statistical complexity of dominant eigenvector calculation. *Journal of Complexity*, 1991.

[6] D. Mortari. EULER-q algorithm for attitude determination from vector observations. *Journal of Guidance, Control, and Dynamics*, 21(2):328–334, 1998.

[7] J. R. Nieto and A. Susín. Cage based deformations: A survey. *Deformation Models: Tracking, Animation and Applications*, 7:75, 2012.

[8] C. B. U. Perwass. *Geometric algebra with applications in engineering*, volume 4 of *Geometry and Computing.* Springer, Berlin; Heidelberg, 2009.

[9] M. D. Shuster and S. D. Oh. Three-axis attitude determination from vector observations. *Journal of Guidance, Control, and Dynamics*, 4(1):70–77, 1981.

[10] F. Sin, D. Schroeder, and J. Barbic. Vega: Non-linear fem deformable object simulator. *Comput. Graph. Forum*, 32(1):36–48, 2013.

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

[12] G. Wahba. A Least Squares Estimate of Satellite Attitude, 1965.

[13] J. H. Wilkinson, editor. *The Algebraic Eigenvalue Problem.* Oxford University Press, Inc., New York, NY, USA, 1988.

[14] J. Wu, Z. Zhou, B. Gao, R. Li, Y. Cheng, and H. Fourati. Fast Linear Quaternion Attitude Estimator Using Vector Observations. *IEEE Transactions on Automation Science and Engineering*, X(X):1–13, 2017.

[15] Y. Yang. Attitude determination using Newton's method on Riemannian manifold. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 229(14):2737–2742, 2015.

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