Geometric-Algebra LMS Adaptive Filter and its Application to Rotation Estimation

Wilder B. Lopes, *Student Member, IEEE,* Anas Al-Nuaimi, Cassio G. Lopes, *Member, IEEE*

***Abstract*—This paper exploits Geometric (Clifford) Algebra (GA) theory in order to devise and introduce a new adaptive filtering strategy. From a least-squares cost function, the gradient is calculated following results from Geometric Calculus (GC), the extension of GA to handle differential and integral calculus. The novel GA least-mean-squares (GA-LMS) adaptive filter, which inherits properties from standard adaptive filters and from GA, is developed to recursively estimate a rotor (multivector), a hypercomplex quantity able to describe rotations in any dimension. The adaptive filter (AF) performance is assessed via a 3D point-clouds registration problem, which contains a rotation estimation step. Calculating the AF computational complexity suggests that it can contribute to reduce the cost of a full-blown 3D registration algorithm, especially when the number of points to be processed grows. Moreover, the employed GA/GC framework allows for easily applying the resulting filter to estimating rotors in higher dimensions.**

***Index Terms*—Adaptive Filters, Geometric Algebra, 3D Registration, Point Cloud Alignment.**

I. Introduction

STANDARD adaptive filtering theory is based on vector calculus and matrix/linear algebra. Given a cost function, usually a least-(mean)-squares criterion, gradient-descent methods are employed, resulting in a myriad of adaptive algorithms that minimize the original cost function in an adaptive manner [1], [2].

This work introduces a new adaptive filtering technique based on GA and GC. Such frameworks generalize linear algebra and vector calculus for hypercomplex variables, specially regarding the representation of geometric transformations [3]–[9]. In this sense, the GA-LMS is devised in light of GA and using results from GC (instead of vector calculus). The new approach is motivated via an actual computer vision (CV) problem, namely 3D registration of point clouds [10]. To validate the algorithm, simulations are run in artificial and real data. The new GA adaptive filtering technique renders an algorithm that may ultimately be a candidate for real-time online rotation estimation.

II. Standard Rotation Estimation

Consider two sets of points – point clouds (PCDs) – in the $\mathbb{R}^3$, Y (*Target*) and X (*Source*), related via a 1-1 correspondence, in which X is a rotated version of Y. Each PCD has K points, $\{y_n\} \in$ Y and $\{x_n\} \in$ X, $n = 1...K$, and their centroids are located at the coordinate system origin.

In the registration process, one needs to find the *linear operator*, i.e., the 3×3 rotation matrix R ([11], p.320), that maps X onto Y. Existing methods pose it as a constrained least-squares problem in terms of R,

$$\mathcal{F}(R)=\frac{1}{K}\sum_{n=1}^{K}\left\|y_n - Rx_n\right\|^2, \text{ subject to } R^*R=RR^*=I_d, \tag{1}$$

in which * denotes the conjugate transpose, and I_d is the identity matrix. To minimize (1), some methods in the literature estimate R directly [12] by calculating the PCDs cross-covariance matrix and performing a singular value decomposition (SVD) [13], [14]. Others use quaternion algebra to represent rotations, recovering the equivalent matrix via a well-known relation [15]–[17].

To estimate a (transformation) matrix, one may consider using Kronecker products and vectorization [2]. However, the matrix size and the possible constraints to which its entries are subject might result in extensive analytic procedures and expendable computational complexity.

Describing 3D rotations via quaternions has several advantages over matrices, e.g., intuitive geometric interpretation, and independence of the coordinate system [18]. Particularly, quaternions require only one constraint – the rotation quaternion should have norm equal to one – whereas rotation matrices need six: each row must be a unity vector (norm one) and the columns must be mutually orthogonal (see [19], p.30). Nevertheless, performing standard vector calculus in quaternion algebra (to calculate the gradient of the error vector) incur a cumbersome analytic derivation [20]–[22]. To circumvent that, (1) is recast in GA (which encompasses quaternion algebra) by introducing the concept of *multivectors*. This allows for utilizing GC to obtain a neat and compact analytic derivation of the gradient of the error vector. Using that, the GA-based AF is conceived without restrictions to the dimension of the underlying vector space (otherwise impossible with quaternion algebra), allowing it to be readily applicable to high-dimensional ($\mathbb{R}^n, n > 3$) rotation estimation problems ([4], p.581).

III. Geometric-Algebra Approach

A. Elements of Geometric Algebra

In a nutshell, the GA $\mathcal{G}(\mathbb{R}^n)$ is a geometric extension of $\mathbb{R}^n$ which enables algebraic representation of orientation and magnitude. Vectors in $\mathbb{R}^n$ are also vectors in $\mathcal{G}(\mathbb{R}^n)$. Each

W. B. Lopes (wilder@usp.br) and C. G. Lopes (cassio@lps.usp.br) are with the Department of Electronic Systems Engineering, University of Sao Paulo, Brazil. A. Al-Nuaimi (anas.alnuaimi@tum.de) is with the Chair of Media Technology (LMT), TU München, Germany. The first author was supported by CAPES Foundation, Ministry of Education of Brazil, under Grant BEX 14601/13-3. Work conducted during a research stay at LMT, TU München.

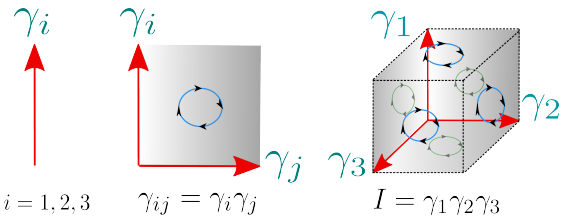

Fig. 1. The elements of $\mathcal{G}(\mathbb{R}^3)$ basis (besides the scalar 1): 3 vectors, 3 bivectors (oriented areas) γ_{ij}, and the pseudoscalar I (trivector/oriented volume).

orthogonal basis in $\mathbb{R}^n$, together with the scalar 1, generates 2^n members (*multivectors*) of $\mathcal{G}(\mathbb{R}^n)$ via the *geometric product* operated over the $\mathbb{R}^n$ ([5], p.19).

Consider vectors a and b in $\mathbb{R}^n$. The geometric product is defined as $ab \triangleq a\cdot b + a\wedge b$, in terms of the inner $(\cdot)$ and outer $(\wedge)$ products ([3], Sec. 2.2). Note that in general the geometric product is noncommutative because $a\wedge b = -(b\wedge a)$. In this text, from now on, all products are geometric products.

For the $\mathbb{R}^3$ case, $\mathcal{G}(\mathbb{R}^3)$ has dimension $2^3 = 8$, with basis $\{1, \gamma_1, \gamma_2, \gamma_3, \gamma_{12}, \gamma_{23}, \gamma_{31}, I\}$, i.e., one scalar, three orthogonal vectors γ_i (basis for $\mathbb{R}^3$), three bivectors $\gamma_{ij} \triangleq \gamma_i\gamma_j = \gamma_i \wedge \gamma_j, i \neq j$ $(\gamma_i \cdot \gamma_j = 0, i \neq j)$, and one trivector (*pseudoscalar*) $I \triangleq \gamma_1\gamma_2\gamma_3$ (Fig.1). To illustrate the geometric multiplication, take two vectors $a = \gamma_1$ and $b = 2\gamma_1 + 4\gamma_3$. Then, $ab = \gamma_1(2\gamma_1 + 4\gamma_3) = \gamma_1\cdot(2\gamma_1 + 4\gamma_3) + \gamma_1 \wedge (2\gamma_1 + 4\gamma_3) = 2 + 4(\gamma_1\wedge\gamma_3) = 2 + 4\gamma_{13}$ (a scalar plus a bivector).

The basic element of a GA is a multivector A,

$$A = \langle A\rangle_0 + \langle A\rangle_1 + \langle A\rangle_2 + \cdots = \sum_g \langle A\rangle_g, \tag{2}$$

which is comprised of its g-grades (or g-vectors) $\langle\cdot\rangle_g$, e.g., $g = 0$ (scalars), $g = 1$ (vectors), $g = 2$ (bivectors), $g = 3$ (trivectors), and so on. The ability to group together scalars, vectors, and hyperplanes in an unique element (the multivector A) is the foundation on top of which GA theory is built on. Except where otherwise noted, scalars $(g = 0)$ and vectors $(g = 1)$ are represented by lower-case letters, e.g., a and b, and general multivectors by upper-case letters, e.g., A and B. Also, in $\mathbb{R}^3$, $\langle A\rangle_g = 0$, $g > 3$ ([4], p.42).

The *reverse* of a multivector A (analogous to conjugation of complex numbers and quaternions) is defined as

$$\widetilde{A} \triangleq \sum_{g=0}^{n} (-1)^{g(g-1)/2} \langle A\rangle_g. \tag{3}$$

For example, the reverse of the bivector $A = \langle A\rangle_0 + \langle A\rangle_1 + \langle A\rangle_2$ is $\widetilde{A} = \widetilde{\langle A\rangle_0} + \widetilde{\langle A\rangle_1} + \widetilde{\langle A\rangle_2} = A_0 + A_1 - A_2$.

The GA *scalar product* $*$ between two multivectors A and B is defined as $A*B = \langle AB\rangle$, in which $\langle\cdot\rangle \equiv \langle\cdot\rangle_0$. From that, the *magnitude* of a multivector is defined as $|A|^2 = A*\widetilde{A} = \sum_g |A|_g^2$, in terms of the magnitude $|A|_g$ of the grades.

B. The Estimation Problem in GA

The problem (1) may be posed in GA as follows. The rotation matrix R in the error vector $y_n - Rx_n$ is substituted by the rotation operator comprised by the bivector r (the only bivector in this paper written with lower-case letter) and its *reversed* version $\widetilde{r}$ [4],

$$e_n = y_n - rx_n\widetilde{r}, \text{ subject to } r\widetilde{r} = \widetilde{r}r = |r|^2 = 1. \tag{4}$$

Thus, r is a *unit rotor* in $\mathcal{G}(\mathbb{R}^3)$. Note that the term $rx_n\widetilde{r}$ is simply a rotated version of the vector x_n ([3], Eq.54). This rotation description is similar to the one provided by quaternion algebra. In fact, it can be shown that the subalgebra of $\mathcal{G}(\mathbb{R}^3)$ containing only the multivectors with even grades (rotors) is *isomorphic* to quaternions [3]. However, unlike quaternions, GA enables to describe rotations in any dimension. More importantly, with the support of GC, optimization problems can be carried out in a clear and compact manner [5], [23].

Hypercomplex AFs available in the literature make use of quaternion algebra [24], [25] and even GA theory [26]. However, the error vector therein has the form $e = y - rx$, which is not appropriate to model rotation error since it lacks $\widetilde{r}$ multiplying x from the right.

This way, (1) is rewritten using (4), generating a new cost function,

$$J(r) = \frac{1}{K}\sum_{n=1}^{K} e_n * \widetilde{e}_n = \frac{1}{K}\sum_{n=1}^{K}\langle e_n\widetilde{e}_n\rangle = \frac{1}{K}\sum_{n=1}^{K}|e_n|^2, \tag{5}$$

subject to $r\widetilde{r} = \widetilde{r}r = |r|^2 = 1$.

IV. Geometric-Algebra LMS

The GA-LMS designed in the sequel should make $rx_n\widetilde{r}$ as close as possible to y_n in order to minimize (5). The AF provides an estimate for the bivector r via a recursive rule of the form,

$$r_i = r_{i-1} + \mu G, \tag{6}$$

where i is the (time) iteration, μ is the AF step size, and G is a multivector-valued update quantity related to the estimation error (4) (analogous to the standard formulation in [2], p.143).

A proper selection of G is required to enforce $J(r_i) < J(r_{i-1})$ at each iteration. This work adopts the steepest-descent rule [1], [2], in which the AF is designed to follow the opposite direction of the reversed gradient of the cost function, namely $\widetilde{\nabla}J(r_{i-1})$ (note the analogy between the reversed $\widetilde{\nabla}$ and the hermitian conjugate ∇^* from the standard formulation). This way, G is proportional to $\widetilde{\nabla}J(r_{i-1})$,

$$G \triangleq -B\widetilde{\nabla}J(r_{i-1}), \tag{7}$$

in which B is a general multivector, in contrast with the standard case in which B would be a matrix [2]. In the AF literature, setting B equal to the identity matrix results in the steepest-descent update rule ([2], Eq. 8-19). In GA though, the multiplicative identity is the multivector (scalar) 1 ([5], p.3), thus $B = 1$.

Embedding $1/K$ into $J(r)$ and expanding yields,

$$\begin{aligned} J(r) &= \sum_{n=1}^{K} \left(y_n - rx_n\widetilde{r}\right) * \widetilde{\left(y_n - rx_n\widetilde{r}\right)} \\ &= \sum_{n=1}^{K} \left[y_n * \widetilde{y}_n - y_n * (r\widetilde{x}\widetilde{r}) - (rx\widetilde{r}) * \widetilde{y}_n + (rx\widetilde{r}) * \widetilde{(rx\widetilde{r})}\right] \\ &= \sum_{n=1}^{K} |y_n|^2 + |x_n|^2 - 2\langle y_n r x_n \widetilde{r}\rangle, \end{aligned} \tag{8}$$

where the reversion rule (3) was used to conclude that $y_n = \widetilde{y}_n$, $x_n = \widetilde{x}_n$ (they are vectors), and $r\widetilde{r} = \widetilde{r}r = 1$.

Using Geometric calculus techniques [5], [23], [27], the gradient of $J(r)$ is calculated from (8),

$$\begin{aligned}\nabla J(r) = \partial_r J(r) &= -2\partial_r \sum_{n=1}^{K} \langle y_n r x_n \widetilde{r} \rangle \\ &= -2\left[\sum_{n=1}^{K} \partial_r \langle \dot{r} M_n \rangle + \partial_r \langle T_n \dot{\widetilde{r}} \rangle\right],\end{aligned} \tag{9}$$

in which the product rule ([23], Eq. 5.12) was used and the overdots emphasize which quantity is being differentiated by ∂_r ([23], Eq. 2.43). The terms $M_n = x_n \widetilde{r} y_n$ and $T_n = y_n r x_n$ are obtained applying the *cyclic reordering property* $\langle AD\cdots C\rangle = \langle D\cdots CA\rangle = \langle CAD\cdots\rangle$ [3]. The first term on the right-hand side of (9) is $\partial_r\langle \dot{r} M_n\rangle = M_n$ ([23], Eq. 7.10), and the second term is $\partial_r\langle T_n \dot{\widetilde{r}}\rangle = -\widetilde{r} T_n \widetilde{r} = -\widetilde{r}(y_n r x_n)\widetilde{r}$ (see the Appendix). Plugging back into (9), the GA-form of the gradient of $J(r)$ is obtained

$$\begin{aligned}\partial_r J(r) &= -2\sum_{n=1}^{K} x_n \widetilde{r} y_n - \widetilde{r}(y_n r x_n)\widetilde{r} \\ &= -2\widetilde{r}\sum_{n=1}^{K}(r x_n \widetilde{r}) y_n - y_n (r x_n \widetilde{r}) = 4\widetilde{r}\sum_{n=1}^{K} y_n \wedge (r x_n \widetilde{r}),\end{aligned} \tag{10}$$

where the relation $ab - ba = 2(a \wedge b)$ was used ([4], p.39).

In [27], the GA framework to handle linear transformations is applied for mapping (10) back into matrix algebra, obtaining a rotation matrix (and not a rotor). Here, on the other hand, the algorithm steps are completely carried out in GA (design and computation), since the goal is to devise an AF to estimate a multivector quantity (rotor) for PCDs rotation problems.

Substituting (10) into (7) (with $B = 1$, as aforementioned) and explicitly showing the term $1/K$ results in

$$G = \frac{4}{K}\left[\sum_{n=1}^{K} y_n \wedge (r_{i-1} x_n \widetilde{r}_{i-1})\right] r_{i-1}, \tag{11}$$

which upon plugging into (6) yields

$$\boxed{r_i = r_{i-1} + \mu\frac{4}{m}\left[\sum_{n=1}^{m} y_n \wedge (r_{i-1} x_n \widetilde{r}_{i-1})\right] r_{i-1}}, \tag{12}$$

where a substitution of variables was performed to enable writing the algorithm in terms of a rank captured by m, i.e., one can select $m \in [1, K]$ to choose how many correspondence pairs are used at each iteration. This allows for balancing computational cost and performance, similar to the Affine Projection Algorithm (APA) rank [1], [2]. If $m = K$, (12) uses all the available points, originating the *geometric-algebra steepest-descent* algorithm. This paper focuses on the case $m = 1$ (one pair per iteration) which is equivalent to approximating $\widetilde{\nabla} J(r)$ by its *current value* in (11) [2],

$$\frac{4}{K}\left[\sum_{n=1}^{K} y_n \wedge (r_{i-1} x_n \widetilde{r}_{i-1})\right] r_{i-1} \approx 4\left[y_i \wedge (r_{i-1} x_i \widetilde{r}_{i-1})\right] r_{i-1}, \tag{13}$$

resulting in the GA-LMS update rule,

$$\boxed{r_i = r_{i-1} + \mu\left[y_i \wedge (r_{i-1} x_i \widetilde{r}_{i-1})\right] r_{i-1}}, \tag{14}$$

in which the factor 4 was absorbed by μ. Note that (14) was obtained without restrictions to the dimension of the vector space containing $\{y_n, x_n\}$.

Adopting (13) has an important practical consequence for the registration of PCDs. Instead of "looking at" the sum of all correspondence-pairs outer products ($m = K$), when $m = 1$ the filter uses only the pair at iteration i, $\{y_i, x_i\}$, to update r_{i-1}. Thus, from an information-theoretic point of view, the GA-LMS uses *less information per iteration* when compared to methods in the literature [12]–[17] that require all the correspondences at each algorithm iteration.

From GA theory it is known that any multiple of a unit rotor q, namely $\lambda q, \lambda \in \mathbb{R}\setminus\{0\}$, $|\lambda q| = \lambda$, provides the same rotation as q. However, it scales the magnitude of the rotated vector by a factor of λ^2, $|(\lambda q) x \widetilde{(\lambda q)}| = \lambda^2 |x|$. Thus, to comply with $r\widetilde{r} = \widetilde{r} r = |r_i|^2 = 1$ (see (5)) and avoid scaling the PCD points, the estimate r_i in (14) is normalized at each iteration when implementing the GA-LMS.

Note on computational complexity. The computational cost is calculated by breaking (14) into parts. The term $r_{i-1} x_i \widetilde{r}_{i-1}$ has two geometric multiplications, which amounts to 28 real multiplications (RM) and 20 real additions (RA). The outer product $y_i \wedge (r_{i-1} x_i \widetilde{r}_{i-1})$ amounts to 6 RM and 3 RA. The evaluation of $\mu\left[y_i \wedge (r_{i-1} x_i \widetilde{r}_{i-1})\right] r_{i-1}$ requires more 20 RM and 12 RA. Finally, $r_{i-1} + \mu\left[y_i \wedge (r_{i-1} x_i \widetilde{r}_{i-1})\right] r_{i-1}$ requires more 4 RA. Summarizing, the cost of the GA-LMS is 54 RM and 39 RA per iteration. SVD-based methods compute the covariance matrix of the $3 \times K$ PCDs at each iteration, which has cost $O(K)$, i.e., it depends on the number of points. This suggests adopting the GA-LMS instead of SVD can contribute to reduce the computational cost when aligning PCDs with a great number of points, especially when $K \gg 54$ and data reuse, typical in CV, is hindered due to time constraints.

V. Simulations

Given K corresponding source and target points (X and Y), the GA-LMS estimates the rotor r which aligns the input vectors in X to the desired output vectors in Y. At first, a "toy problem" is provided depicting the alignment of two cubes PCDs. Then, the AF performance is further tested when registering two PCDs from the "Stanford Bunny"[1], one of the most popular 3D datasets [28].

The GA-LMS is implemented using the GAALET C++ library [29] which enables users to compute the geometric product (and also the outer and inner products) between two multivectors. For all simulations, the rotor initial value is $r = 0.5 + 0.5\gamma_{12} + 0.5\gamma_{23} + 0.5\gamma_{31}$ ($|r| = 1$).

A. Cube registration

Two artificial cube PCDs with edges of 0.5 meters and $K = 1728$ points were created. The relative rotation between the source and target PCDs is 120°, 90°, and 45°, about the x, y, and z axes, respectively. Simulations are performed assuming different levels of measurement noise in the points of the Target PCD, i.e., y_i is perturbed by v_i, a 3×1 random vector with entries drawn from a white Gaussian process of variance $\sigma_v^2 \in \{0, 10^{-9}, 10^{-5}, 10^{-2}\}$.

Fig. 2 shows curves of the excess mean-square error (EMSE$(i) = \mathrm{E}|y_i - r_{i-1} x_i \widetilde{r}_{i-1}|^2$) averaged over 200 realizations. Fig. 2 (top) depicts the typical trade-off between

[1]This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes an .avi video showing the alignment of the PCD sets, the MATLAB code to reproduce the simulations, and a readme file. This material is 25 MB in size.

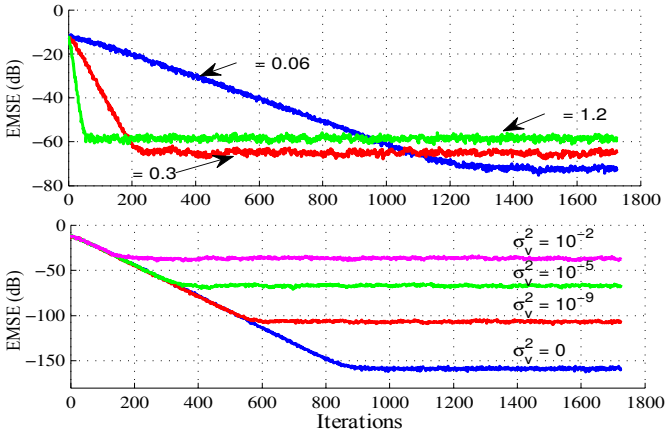

Fig. 2. Cube set. (top) EMSE for $\sigma_v^2 = 10^{-5}$ and different values of μ. (bottom) EMSE for $\mu = 0.2$ and different noise variances σ_v^2. For all cases, the steady state is achieved using only part of the correspondence points. The curves are averaged over 200 realizations.

convergence speed and steady-state error when selecting the values of μ for a given σ_v^2, e.g., for $\mu = 0.3$ the filter takes around 300 iterations (correspondence pairs) to converge, whereas for $\mu = 0.06$ it needs around 1400 pairs. Fig. 2 (bottom) shows how the AF performance is degraded when σ_v^2 increases. The correct rotation is recovered for all cases above. For $\sigma_v^2 > 10^{-2}$ the rotation error approaches the order of magnitude of the cube edges (0.5 meters). For the noise variances in Fig. 2 (bottom), the SVD-based method [14] implemented by the Point Cloud Library (PCL) [10] achieves similar results except for $\sigma_v^2 = 0$, when SVD reaches $-128dB$ compared to $-158dB$ of GA-LMS.

B. Bunny registration

Two specific scans of the "Stanford Bunny" dataset [28] are selected (see Fig. 3), with a relative rotation of $45°$ about the z axis. Each bunny has an average nearest-neighbor (NN) distance of around $0.5mm$. The correspondence between source and target points is pre-established using the matching system described in [30]. It suffices to say the point matching is not perfect and hence the number of true correspondence (TCs) and its ratio with respect to the total number of correspondences is $191/245 = 77\%$.

The performance of the GA-LMS with $\mu = 8$ (selected via extensive parametric simulations) is depicted in Fig. 4. It shows the curve (in blue) for the mean-square error (MSE), which is approximated by the instantaneous squared error (MSE$(i) \approx |d_i - r_{i-1} x_i \widetilde{r}_{i-1}|^2$), where $d_i = y_i + v_i$ is the noise-corrupted version of y_i (in order to model acquisition noise in the scan). As in a real-time online registration, the AF runs only one realization, producing a noisy MSE curve (it is not an ensemble average). Nevertheless, from the cost function (5) curve (in green), plotted on top of the MSE using only the good correspondences, one can see the GA-LMS minimizes it, achieving a steady-state error of $-50.67dB$ at $i \approx 210$. The PCL SVD-based method achieves a slightly lower error of $-51.81dB$ (see supplementary material), although using all the 245 pairs at each iteration. The GA-LMS uses only 1 pair.

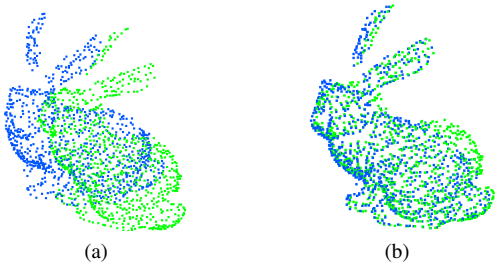

Fig. 3. PCDs of the bunny set. (a) Unaligned, (b) after GA-LMS alignment.

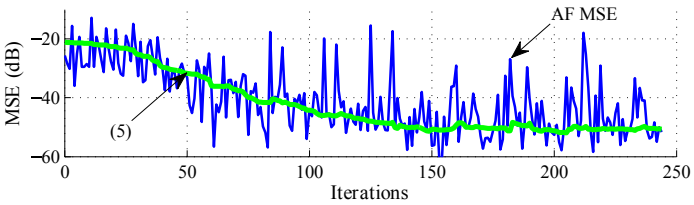

Fig. 4. Bunny set, $\mu = 8$. The cost function (5) curve is plotted on top of the MSE to emphasize the minimization performed by the AF. The steady state is reached before using all the available correspondences.

VI. Conclusion

This work introduced a new AF completely derived using GA theory. The GA-LMS was shown to be functional when estimating the relative rotation between two PCDs, achieving errors similar to those provided by the SVD-based method used for comparison. The GA-LMS, unlike the SVD method, allows to assess each correspondence pair individually (one pair per iteration). That fact is reflected on the GA-LMS computational cost per iteration – it does not depend on the number of correspondence pairs (points) K to be processed, which can lower the computation cost (compared to SVD) of a complete registration algorithm, particularly when K grows. To improve performance, some strategies could be adopted: reprocessing iterations in which the MSE(i) changes abruptly, and data reuse techniques [31]–[33]. A natural extension is to generalize the method to estimate multivectors of any grade, covering a wider range of applications.

Appendix

For a general multivector A and a unit rotor Ω, it holds that

$$\partial_\Omega \langle A \dot{\widetilde{\Omega}} \rangle = -\widetilde{\Omega} A \widetilde{\Omega}. \tag{15}$$

Proof: Given that the scalar part (0-grade) of a multivector is not affected by rotation ($\partial_\Omega \langle \Omega A \widetilde{\Omega} \rangle = 0$), and using the product rule, one can write $\partial_\Omega \langle \Omega A \widetilde{\Omega} \rangle = A\widetilde{\Omega} + \partial_\Omega \langle \Omega A \dot{\widetilde{\Omega}} \rangle = 0$,

$$\partial_\Omega \langle \Omega A \dot{\widetilde{\Omega}} \rangle = -A\widetilde{\Omega}. \tag{16}$$

Using the scalar product definition, the cyclic reordering property, and Eq. (7.2) in [23], $\partial_\Omega \langle \Omega A \dot{\widetilde{\Omega}} \rangle = \partial_\Omega \left[\dot{\widetilde{\Omega}} * (\Omega A) \right] = [(\Omega A) * \partial_\Omega] \widetilde{\Omega}$. Plugging back into (16) and multiplying by $\widetilde{\Omega}$ from the left, $\widetilde{\Omega} [(\Omega A) * \partial_\Omega] \widetilde{\Omega} = -\widetilde{\Omega} A \widetilde{\Omega}$. Since the term $[(\Omega A) * \partial_\Omega]$ is an algebraic scalar, $\widetilde{\Omega} [(\Omega A) * \partial_\Omega] \widetilde{\Omega} = [(\Omega A) * \partial_\Omega] \widetilde{\Omega}\widetilde{\Omega} = \partial_\Omega \left[(\dot{\widetilde{\Omega}}\widetilde{\Omega}) * (\Omega A) \right] = \partial_\Omega \langle (\dot{\widetilde{\Omega}}\widetilde{\Omega}) \Omega A \rangle = \partial_\Omega \langle A \dot{\widetilde{\Omega}} \rangle \Rightarrow \boxed{\partial_\Omega \langle A \dot{\widetilde{\Omega}} \rangle = -\widetilde{\Omega} A \widetilde{\Omega}}$. ■

REFERENCES

[1] Paulo S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, Springer US, 4 edition, 2013.
[2] A.H. Sayed, *Adaptive filters*, Wiley-IEEE Press, 2008.
[3] E. Hitzer, "Introduction to Clifford's Geometric Algebra," *Journal of the Society of Instrument and Control Engineers*, vol. 51, no. 4, pp. 338–350, 2012.
[4] D. Hestenes, *New Foundations for Classical Mechanics*, Fundamental Theories of Physics. Springer, 1999.
[5] D. Hestenes and G. Sobczyk, *Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics*, Fundamental Theories of Physics. Springer Netherlands, 1987.
[6] Leo Dorst, Daniel Fontijne, and Stephen Mann, *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry (The Morgan Kaufmann Series in Computer Graphics)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.
[7] Roxana Bujack, Gerik Scheuermann, and Eckhard Hitzer, "Detection of outer rotations on 3d-vector fields with iterative geometric correlation and its efficiency," *Advances in Applied Clifford Algebras*, pp. 1–19, 2013.
[8] C.J.L. Doran and A.N. Lasenby, *Geometric Algebra for Physicists*, Cambridge University Press, 2003.
[9] Chris J. L. Doran, *Geometric Algebra and Its Application to Mathematical Physics*, Ph.D. thesis, University of Cambridge, 1994.
[10] Radu Bogdan Rusu and Steve Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.
[11] Carl D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, 2001.
[12] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 13, no. 4, pp. 376–380, Apr 1991.
[13] Berthold K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.
[14] G.E. Forsythe and P. Henrici, "The cyclic jacobi method for computing the principal values of a complex matrix," in *Trans. Amer. Math. Soc., Volume 94, Issue 1, Pages 1-23*, 1960.
[15] Michael W. Walker, Lejun Shao, and Richard A. Volz, "Estimating 3-d location parameters using dual number quaternions," *CVGIP: Image Underst.*, vol. 54, no. 3, pp. 358–367, Oct. 1991.
[16] P.J. Besl and Neil D. McKay, "A method for registration of 3-d shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 2, pp. 239–256, 1992.
[17] Zhengyou Zhang, "Iterative point matching for registration of free-form curves and surfaces," *Int. J. Comput. Vision*, vol. 13, no. 2, pp. 119–152, Oct. 1994.
[18] Patrick R. Girard, *Quaternions, Clifford Algebras and Relativistic Physics*, Birkhäuser Basel, 2007.
[19] Erik B. Dam, Martin Koch, and Martin Lillholm, "Quaternions, interpolation and animation - diku-tr-98/5," Tech. Rep., Department of Computer Science, University of Copenhagen, 1998. Available: http://web.mit.edu/2.998/www/QuaternionReport1.pdf.
[20] D.P. Mandic, C. Jahanchahi, and C.C. Took, "A quaternion gradient operator and its applications," *Signal Processing Letters, IEEE*, vol. 18, no. 1, pp. 47–50, Jan 2011.
[21] C. Jahanchahi, C.C. Took, and D.P. Mandic, "On gradient calculation in quaternion adaptive filtering," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, March 2012, pp. 3773–3776.
[22] Mengdi Jiang, Wei Liu, and Yi Li, "A general quaternion-valued gradient operator and its applications to computational fluid dynamics and adaptive beamforming," in *Digital Signal Processing (DSP), 2014 19th International Conference on*, Aug 2014, pp. 821–826.
[23] E. Hitzer, "Multivector differential calculus," *Advances in Applied Clifford Algebras*, vol. 12, no. 2, pp. 135–182, 2002.
[24] C.C. Took and D.P. Mandic, "The quaternion lms algorithm for adaptive filtering of hypercomplex processes," *Signal Processing, IEEE Transactions on*, vol. 57, no. 4, pp. 1316–1327, April 2009.
[25] F.G.A. Neto and V.H. Nascimento, "A novel reduced-complexity widely linear qlms algorithm," in *Statistical Signal Processing Workshop (SSP), 2011 IEEE*, June 2011, pp. 81–84.
[26] E. Hitzer, "Algebraic foundations of split hypercomplex nonlinear adaptive filtering," *Mathematical Methods in the Applied Sciences*, vol. 36, no. 9, pp. 1042–1055, 2013.
[27] J. Lasenby, W. J. Fitzgerald, A. N. Lasenby, and C. J. L. Doran, "New geometric methods for computer vision: An application to structure and motion estimation," *Int. J. Comput. Vision*, vol. 26, no. 3, pp. 191–213, Feb. 1998.
[28] Greg Turk and Marc Levoy, "Zippered polygon meshes from range images," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1994, SIGGRAPH '94, pp. 311–318, ACM.
[29] Florian Seybold and U Wössner, "Gaalet-a c++ expression template library for implementing geometric algebra," in *6th High-End Visualization Workshop*, 2010.
[30] Anas Al-Nuaimi, Martin Piccolorovazzi, Suat Gedikli, Eckehard Steinbach, and Georg Schroth, "Indoor location recognition using shape matching of kinectfusion scans to large-scale indoor point clouds.," in *Eurographics, Workshop on 3D Object Retrieval*, 2015.
[31] W.B. Lopes and C.G. Lopes, "Incremental-cooperative strategies in combination of adaptive filters," in *Int. Conf. in Acoust. Speech and Signal Process.* IEEE, 2011, pp. 4132 –4135.
[32] W.B. Lopes and C.G. Lopes, "Incremental combination of rls and lms adaptive filters in nonstationary scenarios," in *Int. Conf. in Acoust. Speech and Signal Process.* IEEE, 2013.
[33] Luiz F. O. Chamon and Cassio G. Lopes, "There's plenty of room at the bottom: Incremental combinations of sign-error lms filters.," in *ICASSP*, 2014, pp. 7248–7252.