WILDER BEZERRA LOPES

GEOMETRIC-ALGEBRA ADAPTIVE FILTERS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Ciências.

São Paulo
2016

WILDER BEZERRA LOPES

GEOMETRIC-ALGEBRA ADAPTIVE FILTERS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Ciências.

Área de Concentração:

Sistemas Eletrônicos

Orientador:

Prof. Dr. Cássio G. Lopes

São Paulo
2016

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: Wilder B. Lopes

Assinatura do orientador: _____

Catalogação-na-publicação

Lopes, Wilder Bezerra
Geometric-Algebra Adaptive Filters / W. B. Lopes -- versão corr. -- São Paulo, 2016.
101 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.Filtros Elétricos Adaptativos 2.Processamento de Sinais I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

To my family.

ACKNOWLEDGMENTS

To my advisor Prof. Cássio Guimarães Lopes for his full-time support and trust in my work. After six years working together (since my Master's) we learned how to handle our differences for the benefit of the work. He taught me the technical and political aspects of research, skills that I use every day in my job as a researcher/engineer. He gave me his blessing when I decided to spend one year in Germany working on a secondary research topic – without his understanding, that topic would never had become the core of this thesis. Unfortunately, for several reasons, a lot of ideas we came up with were left behind. Hopefully those will become research topics of his future students. This way, I will be happy to know that I contributed a little bit for the continuation of his work.

To my parents Wilson and Derci, and my sister Petúnia, who even far away are very present and supportive in many ways: my deepest gratitude. They are always source of motivation to keep going on what I believe to be the right path.

To my girlfriend Claire for her unconditional love and support. In the last three years she kept me sane in Munich, São Paulo, and now Paris. Without her this work would probably not exist.

To the Professors and students at the Signal Processing Laboratory (LPS-USP): I learned a lot with you all. That laboratory became a second home to me, and I will never forget the period I spent there. I owe special thanks to Prof. Vítor Heloiz Nascimento, who was always helpful, whatever if I had a technical question or if I was trying to figure out my career path.

To Prof. Eckehard Steinbach, my co-advisor in Germany, who kindly hosted me at the Media Technology Chair (LMT) of the Technische Universität München (TUM), where great part of this work was done.

To the researchers and staff of LMT-TUM: the friendly and professional environment was crucial to develop my work and improve my research skills. I will always cherish the time I spent there. Special thanks go to Anas Al-Nuaimi: I am very glad to see that our e-mail discussions, started back in February 2013, built up to this point.

To the friendships I made at the University of São Paulo: Fernando, Chamon, Murilo, Matheus, Amanda, Manolis, David, Renato, Humberto, Yannick. I hope we are able to keep in touch as the years go by.

To the friends spread around the globe, specially Gabriel Silva and Eduardo Sarquis: thanks for the support!

And at last but not least, I would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for the financial support to this work, especially the grant that made possible my research stay in Germany (BEX 14601-13/3).

ABSTRACT

This document introduces a new class of adaptive filters, namely Geometric-Algebra Adaptive Filters (GAAFs). Those are generated by formulating the underlying minimization problem (a least-squares cost function) from the perspective of Geometric Algebra (GA), a comprehensive mathematical language well-suited for the description of geometric transformations. Also, differently from the usual linear algebra approach, Geometric Calculus (the extension of Geometric Algebra to differential calculus) allows to apply the same derivation techniques regardless of the type (subalgebra) of the data, i.e., real, complex-numbers, quaternions etc. Exploiting those characteristics, among others, a general least-squares cost function is posed, from which two types of GAAFs are designed. The first one, called standard, provides a generalization of regular adaptive filters for any subalgebra of GA. From the obtained update rule, it is shown how to recover the following least-mean squares (LMS) adaptive filter variants: real-entries LMS, complex LMS, and quaternions LMS. Mean-square analysis and simulations in a system identification scenario are provided, showing almost perfect agreement for different levels of measurement noise. The second type, called pose estimation, is designed to estimate rigid transformations – rotation and translation – in n-dimensional spaces. The GA-LMS performance is assessed in a 3-dimensional registration problem, in which it is able to estimate the rigid transformation that aligns two point clouds that share common parts.

Keywords – Adaptive filtering, geometric algebra, point-clouds registration, quaternions.

RESUMO

Este documento introduz uma nova classe de filtros adaptativos, entitulados *Geometric-Algebra Adaptive Filters* (GAAFs). Eles são projetados via formulação do problema de minimização (uma função custo de mínimos quadrados) do ponto de vista de álgebra geométrica (GA), uma abrangente linguagem matemática apropriada para a descrição de transformações geométricas. Adicionalmente, diferente do que ocorre na formulação com álgebra linear, cálculo geométrico (a extensão de álgebra geométrica que possibilita o uso de cálculo diferencial) permite aplicar as mesmas técnicas de derivação independentemente do tipo de dados (subálgebra), isto é, números reais, números complexos, quaternions etc. Usando essas e outras características, uma função custo geral de mínimos quadrados é proposta, da qual dois tipos de GAAFs são gerados. O primeiro, chamado *standard*, generaliza filtros adaptativos da literatura concebidos sob a perspectiva de subálgebras de GA. As seguintes variantes do filtro least-mean squares (LMS) são obtidas como casos particulares: LMS real, LMS complexo e LMS quaternions. Uma análise *mean-square* é desenvolvida e corroborada por simulações para diferentes níveis de ruído de medição em um cenário de identificação de sistemas. O segundo tipo, chamado *pose estimation*, é projetado para estimar transformações rígidas – rotação e translação – em espaços n-dimensionais. A performance do filtro GA-LMS é avaliada em uma aplicação de alinhamento tridimensional na qual ele estima a tranformação rígida que alinha duas nuvens de pontos com partes em comum.

Palavras-Chave – Filtragem adaptativa, álgebra geométrica, alinhamento de nuvens de pontos, quaternions.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Since many decades ago, linear algebra (LA) has been the mathematical *lingua franca* across many scientific disciplines. Engineering sciences have resorted to the analytical tools of LA to understand and document their theoretical and experimental results. This is particularly true in signal processing, where basically all the theory can be described by matrices, vectors, and a norm induced by an inner product.

Adaptive filtering, which inherited the mathematical mindset of its parent disciplines (signal processing and control theory), has been successful in expanding its results based on LA. In the design of adaptive filters (AFs) for estimating vectors with real entries, there is no doubt about the efficiency of LA and standard vector calculus. Even if the number field is changed, e.g., from real numbers to complex numbers, requiring new calculus rules to be adopted (*Cauchy-Riemann* conditions) [1], LA is still a reliable tool.

However, the history of mathematics is richer than what is usually covered in engineering courses [2]. One may ask: how LA constructed its reputation along the years? Why is it largely adopted? And more importantly: is it the only way to describe and understand linear transformations? No, it is not. As a matter of fact, it can be shown that the tools of LA are only a subset of something larger. This work takes advantage of this more comprehensive theory, namely *geometric algebra* (GA), which encompasses not only LA but a number of other algebraic

systems [3, 4].

One may interpret LA-based AFs as instances for geometric estimation, since the vectors to be estimated represent directed lines in an underlying vector space. However, to estimate areas, volumes, and hypersurfaces, a regular adaptive filter designed in light of LA might not be very helpful. As it turns out, LA has limitations regarding the representation of geometric structures [4]. Take for instance the inner product between two vectors: it always results in a scalar. Thus, one may wonder if it is possible to construct a new kind of product that takes two vectors (directed lines) and returns an area (or hypersurface, for a vector space with dimension $n > 3$). Or even a product that takes a vector and an area and returns a volume (or hypervolume). Similar ideas have been present since the advent of algebra, in an attempt to establish a deep connection with geometry.

The "new" product aforementioned is the *geometric product*, which is the product operation of GA (as defined in the next chapters). In fact, GA and its product are anything but new. They have been available since the second half of the 19th century, about the same time LA started its ascention to be largely adopted (Section 3.1). The geometric product allows one to actually map a set of vectors not only onto scalars, but also onto hypersurfaces, hypervolumes, and so on. Thus, the use of GA increases the portifolio of geometric shapes and transformations one can represent. Also, its extension to calculus, *geometric calculus* (GC), allows for a clear and compact way to perform calculus with hypercomplex quantities, i.e., elements that generalize the complex numbers for higher dimensions (Section 3.5).

It can be shown that *multivectors* (the fundamental hypercomplex elements of GA) are originated by operating elements of an *orthonormal basis* for an n-dimensional vector space over $\mathbb{R}$ (real numbers) via the geometric product [3–5].

It means that hypercomplex quantities, e.g., complex numbers, quaternions etc., *can be originated without resorting to a number field greater than the reals*. This is an interesting feature of geometric algebras. It greatly simplifies the task of performing calculus with hypercomplex-valued elements, avoiding the need to adopt specific calculus rules for each type of multivector – GC inherently implements that.

Taking advantage of that, this work uses GA and GC to expand concepts of adaptive filtering theory and introduce new elements into it. The filters devised herein, namely Geometric-Algebra Adaptive Filters (GAAFs), are able to naturally estimate hypersurfaces, hypervolumes, and elements of greater dimensions (multivectors). In this sense, this research exploits the tools of GA and GC to generate a new class of AFs capable of encompassing the regular ones. For instance, filters like the regular least-mean squares (LMS) – with real entries, the Complex LMS (CLMS) – with complex entries, and the Quaternion LMS (QLMS) – with quaternion entries, are recovered as special cases of the more comprehensive GA-LMS introduced by this work.

Two applications are employed to assess the performance of the GAAFs. The first one, system identification (system ID), tests the ability of the GAAFs to estimate the multivector-valued coefficients of a finite impulse response (FIR) plant. The second one, three-dimensional (3D) registration (alignment) of point clouds (PCDs) – a typical computer vision (CV) problem –, exploits the geometric estimation capabilities of GAAFs to align 3D objects sharing common parts. Both applications highlight the unique features originated from the combination of GA and adaptive filtering theories.

1.1 Contributions of the Work

The main contributions of this research are listed below:

1. *Recast of central concepts of linear estimation into GA framework* - In the standard literature, those concepts are presented in light of LA. Chapter 4 shows how to describe them using GA language. Among the definitions provided therein are: array of multivectors and random multivectors.

2. *GAAFs (standard shape)* - These are the first type of GAAFs introduced by this work. They aim at generalizing LA-based AFs. Their key application is system ID. Chapter 5 presents:

   - Design of GA Least-Mean Squares (GA-LMS);
   - Steady-state mean-square analysis, in which the steady-state errors for LA-based AFs (e.g., real-entries LMS, complex-entries LMS, quaternion LMS) are recovered as particular cases.

3. *GAAFs (pose estimation)* - These are the second type of GAAFs introduced by this work. Their key application is estimation of the rigid transformation that aligns a pair of 3D PCDs. Chapter 6 presents:

   - Design of GA-LMS for pose estimation;
   - Evaluation of the computational complexity;
   - Calculation of step-size bounds as a function of the PCDs points and their greatest dimension.

4. *Computational implementation* - GA requires special libraries and/or toolboxes to implement the geometric product. A C++ library was adopted and improved with new *headers* in order to write the GAAFs source codes. Those were compiled and the binaries were called from MATLAB® to run

the experiments. The experiments presented in Chapter 7 show that the GAAFs are successful in both tasks (system ID and 3D registration).

All source codes and scripts are available on openga.org, a companion website to this text.

Above all, this work is expected to motivate the use of GA, this rather neglected yet versatile mathematical language, among scientists and engineers.

1.2 About Text Organization

It was chosen to provide the reader with the necessary background material to follow the derivations. Chapter 2 presents the notation and explains the scenarios for testing the GAAFs (system ID and 3D registration of PCDs). Chapter 3 covers the fundamentals of GA, providing a brief history on the subject, several important definitions and relations, and the very basics of GC. Hopefully, only very specific points will require consulting extra literature. Whenever this happens, proper references are cited.

The style of presentation is biased towards the signal processing area, particularly adaptive filtering. It is assumed the reader is fluent in LA and has experience with stochastic processes theory. Previous knowledge on the design of adaptive filters based on LA will certainly help to appreciate the work and perceive the key differences, however it is not strictly necessary. In fact, those with little or no experience in LA-based adaptive filtering might benefit from the comprehensiveness of GA and GC to appreciate some details that are not evident for those used to the LA approach.

Chapter 4 recasts standard linear estimation results into GA. Definitions like random multivectors, array of multivectors and array product are provided. More importantly, it is shown that the cost function of the system ID problem and the

one of 3D registration of PCDs are particular cases of a general cost function that can only be written using geometric multiplication.

Chapter 5 introduces GAAFs for the system ID application. The problem is posed in light of GA, the gradient of the cost function is calculated and the GA-LMS is devised. Also, mean-square analysis (steady state) is provided with the support of the energy conservation relations [1].

Chapter 6 presents GAAFs for the 3D registration of PCDs. One filter is devised (GA-LMS for pose estimation) and its computational complexity and step-size bounds are evaluated. The mean-square analysis of the GAAFs for pose estimation is not featured here, the reason being that it was not concluded at the time of the text submission. Other filters, namely GA-NLMS and GA-RLS, were derived (see Appendix A), however computational implementation for them is missing as well as experiments. This will be covered in future works.

Experiments for the two scenarios (system ID and 3D registration of PCDs) are shown in Chapter 7: in the system ID case, simulations depicting several learning curves corroborate theoretical predictions almost perfectly. This study is performed for four different subalgebras of GA; in the 3D registration case, the GA-LMS is shown to be able to estimate the correct rigid transformation that aligns the pair of PCDs, achieving results similar to a standard registration method available in the literature. Ultimately, the GA-LMS low computational complexity (compared to standard registration algorithms) and its adaptive nature turn it into a candidate to substitute the estimation algorithms of full-blown 3D alignment methods.

Finally, discussion and conclusion are presented in Chapter 8. Information about ongoing work and topics for future research are also provided.

2 PRELIMINARIES

This chapter introduces preliminary material to support the exposition made in the following chapters. Particularly, the two testbench cases used to assess the performance of the designed AFs, namely system identification and registration of point clouds, are explained.

2.1 Notation

The notation adopted in this text is summarized below. When necessary, it is reminded to the reader throughout the text.

- Boldface letters are used for random quantities and normal font letters for deterministic quantities. For instance, z is a realization (deterministic) of the random variable $\boldsymbol{z}$.

- Capital letters are used for general multivectors (see Definition 6 further ahead) and matrices. For instance, A is a general multivector, while $\boldsymbol{A}$ is a random multivector. There are only two matrices in this text: rotation matrix R and identity I_d.

- Small letters represent arrays of multivectors, vectors and scalars. For example, z is an array of multivectors, a vector or scalar. The type of the variable will be clear from the context. Also, it is important to notice the following exceptions (which are properly justified in the body of the text):

  - The small letter r is used to represent a rotor (a kind of multivector);
  - The small letter d represents a general multivector;
  - The small letter v represents a general multivector.

- The time-dependency of a scalar or multivector quantity is denoted by parentheses, while subscripts are employed to denote the time-dependency of arrays and vectors. For instance, $u(i)$ is a time-varying scalar and u_i is a time-varying array or vector, while $U(i)$ is a time-varying general multivector.

- The symbol * is used to represent the *reverse array*, i.e., an array whose entries are reversed (see Definition 31 further ahead). Its utility is clarified in Chapter 4.

2.2 The System Identification Problem

A system identification scenario is adopted in Section 7.2 to evaluate the performance of the *standard* GAAFs devised in Chapter 5.

Consider the schematic depicted in Fig. 1. The goal is to estimate the entries (coefficients) of an unknown plant (system) modeled by an $M \times 1$ vector w^o, which relates the system input-output via

$$d(i) = u_i^H w^o + v(i), \tag{2.1}$$

where u_i is an $M \times 1$ vector (sometimes known as the regressor, which collects input samples as $u_i = [u(i)\,u(i-1)\cdots u(i-M+1)]$), ^{H} denotes Hermitian conjugate, and $v(i)$ represents measurement noise typically modeled as a white Gaussian process with variance σ_v^2 [1, 6].

At each iteration, the unknown plant and the adaptive system w_i are fed with the same regressor u_i. The output $d(i)$ of the unknown system is contaminated

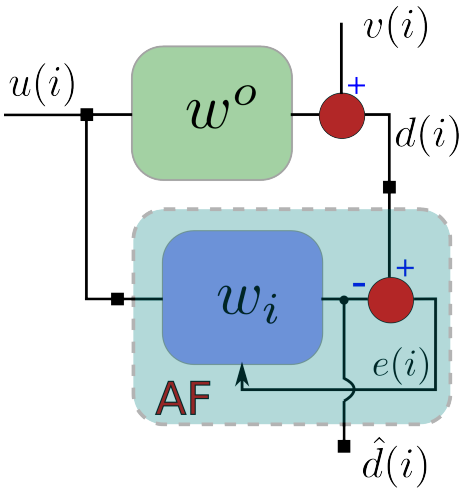


Figure 1: The system identification scenario.

by measurement noise $v(i)$ and the adaptive system output is subtracted from $d(i)$. This generates the output estimation error $e(i)$ which is fed back into the estimator in order to update its coefficients w_i. That iterative process continues until the adaptive system has converged (steady-state), minimizing $e(i)$, usually in the mean-square sense. At that stage ($i \to \infty$), w_i is the best estimate of the unknown plant w^o.

2.3 Registration of Point Clouds

A point cloud (PCD) is a data structure used to represent a collection of multi-dimensional points. For the three-dimensional case (Euclidean space), its points are the geometric coordinates of an underlying surface in $\mathbb{R}^3$ [7].

The PCD registration problem is concerned about aligning two PCDs of the same object (generated from different perspectives), which share a common region. Figure 2 shows the standard registration pipeline. At first, the intersecting region of the PCDs is identified via detection of features (points of interest, e.g., corners) in each PCD. Then, the features of one PCD are matched to the features in the other PCD (Feature Matching), producing pair of points called *correspondences* (see Figure 3). The correspondences are then fed into an estimation al-

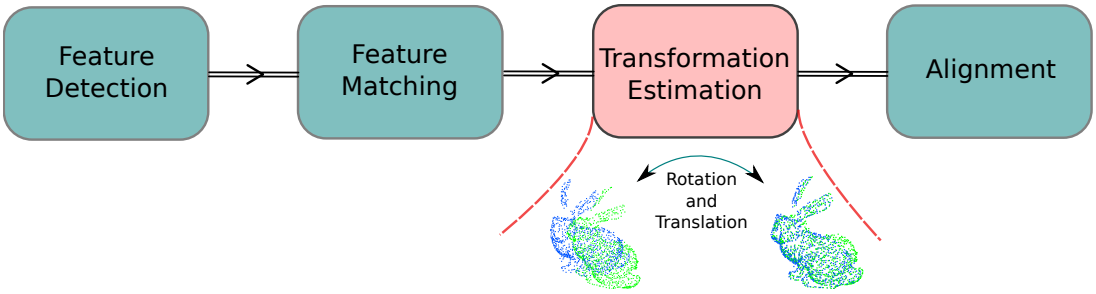


Figure 2: Registration Pipeline. The goal is to match two PCDs (in this case, bunnies) which are initially unaligned. This work focuses on the "Transformation Estimation" phase, where a new estimator based on GA and AFs is introduced.

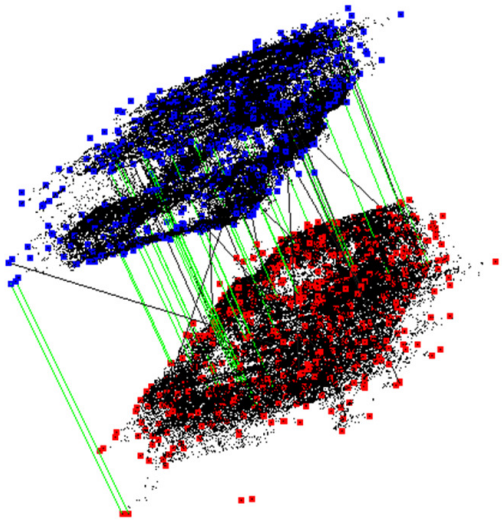


Figure 3: Feature matching of two point clouds with a common region. Notice the green lines, which represent established correspondences between the PCDs points. Source: courtesy of Anas Al-Nuaimi (http://www.lmt.ei.tum.de/team/mitarbeiter/anas-al-nuaimi.html).

gorithm in order to calculate the rigid transformation (rotation and translation) that aligns the PCDs. The best transformation obtained during the estimation phase is then employed to effectively register the PCDs [8–11].

Chapter 6 introduces a GA-based adaptive filter which can be used as the rigid-transformation estimator in a full-blown registration pipeline. It is shown that the so-called GAAFs *for pose estimation* can successfully recover the six degrees-of-freedom (6DOF) transformation, i.e., rotation and translation, that aligns two PCDs (Section 7.3).

3 FUNDAMENTALS OF GEOMETRIC ALGEBRA

Geometric Algebra (also called Clifford Algebra after the British mathematician William Kingdon Clifford) was first developed as a mathematical language to unify all the different algebraic systems trying to express geometric relations/transformations, e.g., rotation and translation [3–5, 12]. All the following geometric systems are particular cases (*subalgebras*) of GA: vector and matrix algebras, complex numbers, and quaternions (see Section 3.3). Depending on the application, one system is more appropriate than the others, and sometimes it is necessary to employ two or more of those algebras in order to precisely describe the geometric relations. Before the advent of GA, this eventually resulted in lack of clarity due to the extensive translations from one system to the other.

This chapter provides the fundamentals (history and mathematical theory) of GA necessary for the derivation of the AFs in Chapters 5 and 6. Moreover, the extension of GA to enable differential and integral calculus in this comprehensive algebra, namely *Geometric Calculus*, is introduced.

A complete coverage of GA theory is not in the scope of this text. For an in-depth discussion about GA theory & history, and its importance to Physics, please refer to [3–5, 12–14]. For applications of GA in engineering and computer science, check [15–20]. Finally, to contextualize the development of GA with the advent of *abstract algebra*, the reading of the historical report in [2] is recommended.

3.1 A Brief History of GA

Geometric algebra theory is an answer to a series of questions first posed many years ago: is it possible to remove the constraints that make this world 3-dimensional? What does it take to be capable of seeing beyond the limits of 3-dimensional space? Once free of the constraints, how this new space can be described?

It comes with no surprise that Clifford, besides a mathematician, was also a philosopher. He is the main actor in the history of GA: it was him who, in the second half of the nineteenth century, was able to see that many different systems trying to describe geometric relations could be unified under the same language. Moreover, he built upon that, generating new results, and foresaw many years ahead.

Greek geometry was concerned with describing the forms of the physical world. The manipulation of straight lines, circles, and other shapes were the tools to represent bodies and forms. Although there was no correspondence between those geometric tools and numbers (the latter were only associated to the activity of counting). Along the following centuries, Arabic science evolved, delivering to the world the arabic numbers and the embryo of algebra.

It was not until the work of Descartes, in the middle of the seventeenth century, that a deep connection between algebra and geometry was established. By uniquely associating each line segment to a letter representing its numerical length (magnitude), Descartes could apply the fundamental operations (sum, subtraction, multiplication, division, and root extraction) to the letters in order to perform geometric transformations on the line segments. That simple, yet powerful, correspondence between algebraic and geometric elements laid out the way to go beyond the 3-dimensional world. Indeed, from that point on, the

tools of algebra started to enable the description of *abstract* forms and shapes, a previously unachievable task if one had to resort to pure geometry (direct manipulation of lines and shapes).

The necessity for an algebraic tool to describe the *orthogonal projection* (a concept already present in Greek geometry) of one line segment on another motivated the advent of the *inner product* and the concept of directed segments (vectors). This marks the beginning of *vector algebra*, usually credited to J. W. Gibbs, who worked on that subject during the 1870s. However, the concepts of vectors, inner product, and vector space had already been introduced back in the 1840s by the works of Hermann Grassmann and William Hamilton [21].

Grassmann's theory – "Ausdehnungslehre" (theory of extension) – was more comprehensive than Gibbs' since, besides the inner product, it introduced the concept of *outer product*. This product captures the geometric fact that two non-parallel directed segments determine a parallelogram, a notion which can not be described by the inner product. Due to multiple reasons, the scientific community did not fully appreciate Grassman's work, keeping it mostly unnoticed till the late 1870s. At this time, Gibbs' approach had already made its way into the scientific community, particularly physicists, who adopted it as a substitute to Hamilton's quaternion algebra (considered a redundant and complicated language) to describe electromagnetic fields.

That unfortunate turn of events had deep consequences on the way scientists think the relationship algebra-geometry. Grassmann's exterior algebra, which introduced the key elements for a complete algebraic description of geometric transformations, remained in the background of mathematics, despite the fact that its notation could be used to simplify much of classical Physics [4]. As pointed out by Gian-Carlo Rota (1932–1999) [22]: "The neglect of exterior algebra is the mathematical tragedy of this century. Only now is it slowly being

corrected."

Hamilton's quaternions had the chance to be on the center of the stage of 19th century Physics and Mathematics. Nevertheless, the quaternion product (which required to separate scalar part from vector part) introduced a number of difficulties, creating a reputation of overcomplicated for Hamilton's brainchild. There was one piece missing: to remove the cumbersomeness, scalar and vector parts should be treated as elements of the same set.

Clifford came up with a brilliant solution for that: the *geometric product*. He defined that product in terms of the inner and outer products (see Definition 5). In this sense, Clifford built upon Grassmann's work to unify the algebras of inner and outer products and create a new one – Geometric Algebra. Additionally, he introduced the concept of multivectors (or Clifford numbers), the basic elements of GA: hypercomplex quantities that generalize scalars, vectors, complex numbers, quaternions, and so on. This naturally turned them elements of the same group, which can be operated regardless of their type.

Clifford's developments in GA put him in a privileged position to foresee the future of Mathematical Physics. His most astounding ideas are documented in [23] and [24], published in 1876 and 1878, respectively. Although it is far from being a full-blown theory of spacetime, he interpreted matter as a manifestation of curvature in a spacetime manifold, anticipating Albert Einstein's ideas on general relativity (published in 1915) in approximately 40 years. Shortly after publishing [24], Clifford passed away at the age of 33, leaving a number of ideas unfinished. His premature death was another unfortunate event in the history of GA, what increased the delay in propagating the theory.

In recent times, David Hestenes published [3] (1984) and [4] (1999) in an attempt to promote the effective use of GA in mathematics and physics. His work has influenced a number of scientists and engineers who have adopted GA

as the mathematical language in their own research [15–20].

3.2 Constructing the Geometric Algebra of a Vector Space

In this Section, the Geometric algebra of a vector space is gradually constructed. Along the process, a series of definitions are presented. The explanation starts with the definition of algebra.

Definition 1 (Definition of Algebra)**.** A vector space $\mathcal{V}$ over the reals $\mathbb{R}$, equipped with a bilinear product $\mathcal{V} \times \mathcal{V} \to \mathcal{V}$ denoted by $\circ$, is said to be an *algebra* over $\mathbb{R}$ if the following relations hold $\forall\{a, b, c\} \in \mathcal{V}$ and $\{\alpha, \beta\} \in \mathbb{R}$ [5, 12]:

$$\begin{aligned} (a+b)\circ c &= a\circ c + b\circ c \text{ (Left distributivity)} \\ c\circ(a+b) &= c\circ a + c\circ b \text{ (Right distributivity)} \\ (\alpha a)\circ(\beta b) &= (\alpha\beta)(a\circ b) \text{ (Compatibility with scalars).} \end{aligned} \tag{3.1}$$

The associative property, i.e., $(a \circ b) \circ c = a \circ (b \circ c)$, does not necessarily hold for the product $\circ$. □

In a nutshell, the GA of a vector space $\mathcal{V}$ over the reals $\mathbb{R}$, namelly $\mathcal{G}(\mathcal{V})$, is a geometric extension of $\mathcal{V}$ which enables algebraic representation of orientation and magnitude. Vectors in $\mathcal{V}$ are also vectors in $\mathcal{G}(\mathcal{V})$. The properties of $\mathcal{G}(\mathcal{V})$ are defined by the signature of $\mathcal{V}$:

Definition 2 (Signature of a Vector Space/Algebra)**.** Let $\mathcal{V} = \mathbb{R}^n = \mathbb{R}^{p,q,r}$, with $n = p + q + r$. The *signature of a vector space* (and by extension of the algebra constructed from it) is expressed in terms of the values $\{p, q, r\}$, i.e., $\mathbb{R}^n = \mathbb{R}^{p,q,r}$ has signature $\{p, q, r\}$. An orthonormal basis of $\mathbb{R}^n = \mathbb{R}^{p,q,r}$ has p vectors that square to 1, q vectors that square to -1, and r vectors that square to 0. □

In the signal processing literature, which is built on top of the theory of

linear algebra (LA), one usually considers only vector spaces for which the basis elements square to 1, i.e., $q = r = 0 \Rightarrow \mathbb{R}^{p,0,0} = \mathbb{R}^{n,0,0} = \mathbb{R}^n$. Thus, one can say that $\mathbb{R}^{p,0,0}$ has *Euclidean signature* (see [3, p.42 and p.102]). GA allows for a more comprehensive approach to vector spaces. It naturally takes into account the so-called *pseudo-Euclidean* spaces, where q and r can be different than zero. Such feature allows to build algebras with pseudo-Euclidean signatures. From here on, the derivations require only Euclidean signatures, except when otherwise noted.

The main product of the algebra $\mathcal{G}(\mathcal{V})$ is the so-called *geometric product*. Before defining it, it is first necessary to define the *inner* and *outer* products. Those are approached by considering vectors a and b in the vector space $\mathbb{R}^n$.

Definition 3 (Inner Product of Vectors)**.** The inner product $a \cdot b$, $\{a, b\} \in \mathbb{R}^n$, is the usual vector product of linear algebra, defining the (linear) algebra generated by the vector space $\mathbb{R}^n$. This way, $a \cdot b$ results in a scalar,

$$a \cdot b = |a||b|cos\theta, \tag{3.2}$$

in which θ is the angle between a and b. Additionaly, the inner product is commutative, i.e., $a \cdot b = b \cdot a$. See Figure 4. □

Definition 4 (Outer Product of Vectors)**.** The outer product $a \wedge b$, $\{a, b\} \in \mathbb{R}^n$, is the usual product in the exterior algebra of Grassman [12]. The multiplication $a \wedge b$ results in an *oriented area* or *bivector*. Such an area can be interpreted as the parallelogram (hyperplane) generated when vector a is swept on the direction determined by vector b (See Figure 4). The resulting bivector (oriented area) is uniquely determined by this geometric construction. That is the reason it may be considered as a kind of product of the vectors a and b. This way, $a \wedge b = C$, where C is the oriented area (bivector). Alternatively, the outer product can be defined as a function of the angle θ between a and b

$$a \wedge b = C = I_{a,b}|a||b|sin\theta, \tag{3.3}$$

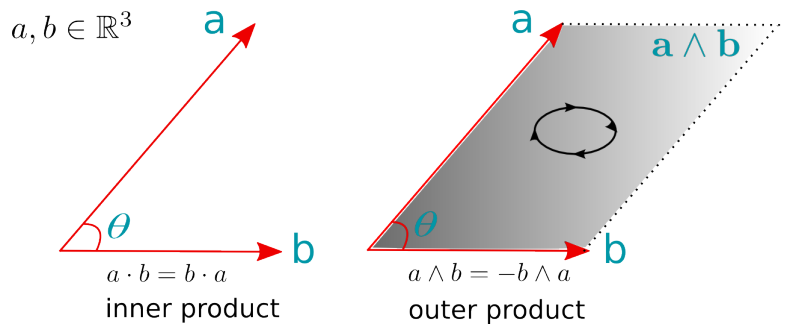


Figure 4: Visualization of the inner and outer products in $\mathbb{R}^3$. In the outer product case, the orientation of the circle defines the orientation of the area (bivector).

where $I_{a,b}$ is the *unit bivector*[1] that defines the orientation of the hyperplane $a \wedge b$ [4, p.66].

The outer product is noncommutative, i.e., $a \wedge b = -b \wedge a$. This can be concluded from Figure 4: the orientation of the area generated by sweeping a along b $(a \wedge b)$ is opposite to the orientation of the area generated be sweeping b along a $(b \wedge a)$.

For a detailed exposition on the nature of the outer product, please refer to [4, p.20] and [12, p.32]. □

Definition 5 (Geometric Product of Vectors)**.** The geometric product is defined as

$$ab \triangleq a \cdot b + a \wedge b, \tag{3.4}$$

in terms of the inner $(\cdot)$ and outer $(\wedge)$ products ([5], Sec. 2.2). □

Remark 1. Note that in general the geometric product is *noncommutative* because $a \wedge b = -(b \wedge a)$, resulting in $ab = -ba$. Also, it is *associative*, $a(bc) = (ab)c$, $\{a, b, c\} \in \mathbb{R}^n$. □

In this text, from now on, all products are geometric products, unless otherwise noted.

[1] An unit bivector is the result of the outer product between two unit vectors, i.e., vector with unitary norm.

Next, the general element of a geometric algebra, the so-called multivector, is defined.

Definition 6 (Multivector (Clifford number))**.** A is a multivector (Clifford number), the basic element of a Geometric Algebra $\mathcal{G}$,

$$A = \langle A\rangle_0 + \langle A\rangle_1 + \langle A\rangle_2 + \cdots = \sum_g \langle A\rangle_g, \tag{3.5}$$

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, generated via the geometric multiplication of two vectors), $g = 3$ (trivectors, generated via the geometric multiplication of three vectors), and so on. The ability to group together scalars, vectors, and hyperplanes in a unique element (the multivector A) is the foundation on top of which GA theory is built on. □

Remark 2. Recall Section 2.1: 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 $\mathbb{R}^3$, $\langle A\rangle_g = 0$, $g > 3$, i.e., there are no grades greater than three [4, p.42]. □

Definition 7 (Grade operator)**.** To retrieve the grade p of a multivector,

$$\langle A\rangle_p \triangleq A_p; \quad p = 0 \Rightarrow \langle A\rangle_0 \equiv \langle A\rangle. \tag{3.6}$$

□

This way, multivectors are the elements that populate the geometric algebra of a given vector space. Moreover, the concept of a multivector, which is central in GA theory, allows for "summing apples and oranges" in a well-defined fashion. Vectors can be added to (multiplied by) scalars, which can then be added to (multiplied by) bivectors, and so on, without having to adopt special rules: the same algebraic tools can be applied to any of those quantities (subalgebras).

This represents an amazing analytic advantage when compared to linear algebra, where scalars and vectors belong to separated realms. It also gives support to the idea presented in Chapter 1: the field of real numbers, combined with a sophisticated algebra like GA, is enough to perform analysis with hypercomplex quantities (there might be no need for a number field more comprehensive than $\mathbb{R}$, e.g., the complex numbers field $\mathbb{C}$).

Now let the set of vectors $\{\gamma_k\} \in \mathbb{R}^n, k = 1, 2, \cdots, n,$

$$\{\gamma_1, \gamma_2, \cdots, \gamma_p, \gamma_{p+1}, \cdots, \gamma_{p+q}, \gamma_{p+q+1}, \cdots, \gamma_n\}, \text{with } n = p+q+r \text{ (recall Definition 2)}, \tag{3.7}$$

for which the following relations hold

$$\gamma_k^2 = \begin{cases} 1, & k = 1, \cdots, p \text{ (square to 1)} \\ -1, & k = p+1, \cdots, p+q \text{ (square to -1)} \\ 0, & k = p+q+1, \cdots, n \text{ (square to 0)}, \end{cases} \tag{3.8}$$

be an *orthonormal basis* of $\mathbb{R}^n$. Using that, the Geometric (Clifford) algebra can be formally defined:

Definition 8 (Clifford Algebra)**.** Given an orthonormal basis of $\mathbb{R}^n$, its elements form a Geometric (Clifford) algebra $\mathcal{G}(\mathbb{R}^n)$ via the geometric product according to the rule [5, 12]

$$\gamma_k \gamma_j + \gamma_j \gamma_k = 2\gamma_k^2 \delta_{k,j}, \quad k, j = 1, \cdots, n, \tag{3.9}$$

where $\delta_{k,j} = 1$ for $k = j$, and $\delta_{k,j} = 0$ for $k \neq j$, which emphasizes the noncommutativity of the geometric product. □

Thus, a basis for the geometric algebra $\mathcal{G}(\mathbb{R}^n)$ is obtained by multiplying the n vectors in (3.7) (plus the scalar 1) according to (3.9). This procedure generates 2^n members (multivectors), defining an algebra and its dimension.

Definition 9 (Subspaces and dimensions)**.** Consider a vector space $\mathcal{V}$, whose basis has dimension n, which generates the *complete Geometric Algebra* of $\mathcal{V}$ (or $\mathcal{G}(\mathcal{V})$). Adding and mutiplying g linearly-independent vectors ($g \leq n$) in $\mathcal{V}$ generates a linear subspace $\mathcal{G}^g(\mathcal{V})$ (closed under the geometric product) of $\mathcal{G}(\mathcal{V})$. The dimension of each subspace $\mathcal{G}^g(\mathcal{V})$ is $\binom{n}{g}$. Thus, the dimension of the complete algebra $\mathcal{G}(\mathcal{V})$ is ([3], p.19)

$$\dim\{\mathcal{G}(\mathcal{V})\} = \sum_{g=0}^{n} \dim\{\mathcal{G}^g(\mathcal{V})\} = \sum_{g=0}^{n} \binom{n}{g} = 2^n \tag{3.10}$$

□

When $n = 3 \Rightarrow \mathcal{V} = \mathbb{R}^3$, which is the main case studied in this work, (3.7) becomes

$$\{\gamma_1, \gamma_2, \gamma_3\}. \tag{3.11}$$

This way, according to (3.10), $\mathcal{G}(\mathbb{R}^3)$ has dimension $2^3 = 8$, with basis

$$\{1, \gamma_1, \gamma_2, \gamma_3, \gamma_{12}, \gamma_{23}, \gamma_{31}, I\}, \tag{3.12}$$

which, as aforementioned, is obtained by multiplying the elements of (3.11) (plus the scalar 1) via the geometric product. Note that (3.12) has one scalar, three orthonormal vectors γ_i (basis for $\mathbb{R}^3$), three bivectors (oriented areas) $\gamma_{ij} \triangleq \gamma_i\gamma_j = \gamma_i \wedge \gamma_j, i \neq j$ ($\gamma_i \cdot \gamma_j = 0, i \neq j$), and one trivector (*pseudoscalar*[2]) $I \triangleq \gamma_1\gamma_2\gamma_3 = \gamma_{123}$ (Figure 5).

To illustrate the geometric multiplication between elements of $\mathcal{G}(\mathbb{R}^3)$, take two multivectors $A = \gamma_1$ 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).

In the sequel, it is shown how the geometric algebra $\mathcal{G}(\mathbb{R}^n)$ encompasses sub-algebras of interest, e.g., rotor algebra. In particular, some well-known algebras

[2]The proper definition of pseudoscalar is given further ahead in (3.28).

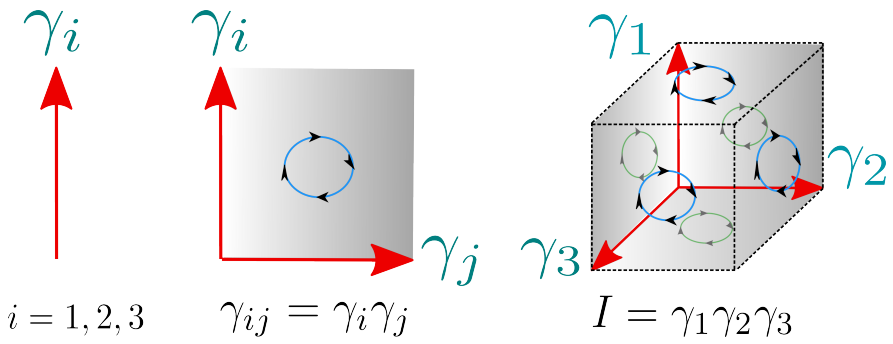


Figure 5: The elements of $\mathcal{G}(\mathbb{R}^3)$ basis (besides the scalar 1): 3 vectors, 3 bivectors (oriented areas) γ_{ij}, and the trivector I (pseudoscalar/oriented volume).

like complex numbers and quaternion algebras are retrieved from the complete $\mathcal{G}(\mathbb{R}^n)$ via isomorphism.

3.3 Subalgebras and Isomorphisms

As pointed out in Definition 9, adding and multiplying g linearly-independent vectors in a given set $\mathcal{V}$ generates a subalgebra $\mathcal{G}^g(\mathcal{V})$ (closed under the geometric product) of $\mathcal{G}(\mathcal{V})$. This endows the GA of $\mathcal{V}$ with the capability of encompassing previously known algebras, like the ones originated by real, complex, and quaternion numbers.

In abstract algebra, two structures are said to be *isomorphic* if they have equivalent algebraic properties, enabling the use of one or the other interchangeably [4,5]. In other words, the algebras are mutually identified, with well-defined correspondences (bijective relationship) between their elements.

This section highlights the isomorphism between subalgebras of GA and two algebras commonly used in the adaptive filtering and optimization literature: complex numbers and quaternions [25–31]. In particular, it is shown how those algebras fit into the comprehensive framework of GA. The described isomorphisms ultimately support the argument defended in this text: GAAFs generalize the standard AFs specifically designed for each algebra, i.e., real, complex, and

Table 1: Multiplication table of $\mathcal{G}(\mathbb{R}^3)$ via the geometric product.

	1	γ_1	γ_2	γ_3	γ_{12}	γ_{23}	γ_{31}	I
1	1	γ_1	γ_2	γ_3	γ_{12}	γ_{23}	γ_{31}	I
γ_1	γ_1	1	γ_{12}	$-\gamma_{31}$	γ_2	I	$-\gamma_3$	γ_{23}
γ_2	γ_2	$-\gamma_{12}$	1	γ_{23}	$-\gamma_1$	γ_3	I	γ_{31}
γ_3	γ_3	γ_{31}	$-\gamma_{23}$	1	I	$-\gamma_2$	γ_1	γ_{12}
γ_{12}	γ_{12}	$-\gamma_2$	γ_1	I	-1	$-\gamma_{31}$	γ_{23}	$-\gamma_3$
γ_{23}	γ_{23}	I	$-\gamma_3$	γ_2	γ_{31}	-1	$-\gamma_{12}$	$-\gamma_1$
γ_{31}	γ_{31}	γ_3	I	$-\gamma_1$	$-\gamma_{23}$	γ_{12}	-1	$-\gamma_2$
I	I	γ_{23}	γ_{31}	γ_{12}	$-\gamma_3$	$-\gamma_1$	$-\gamma_2$	-1

quaternions (Chapter 5).

3.3.1 Complete Geometric Algebra of $\mathbb{R}^3$

The basis of $\mathcal{G}(\mathbb{R}^3)$ is given by (3.12). Squaring each of the elements in (3.12) results in

$$\begin{aligned}
1^2 &= 1, \underbrace{\ (\gamma_1)^2 = 1, \ (\gamma_2)^2 = 1, \ (\gamma_3)^2 = 1}_{\text{From the algebra signature}} \\
(\gamma_{12})^2 &= \gamma_1\gamma_2\gamma_1\gamma_2 = -\gamma_1 \underbrace{(\gamma_2\gamma_2)}_{=1} \gamma_1 = -\gamma_1\gamma_1 = -1 \\
(\gamma_{23})^2 &= \gamma_2\gamma_3\gamma_2\gamma_3 =\therefore= -1 \\
(\gamma_{31})^2 &= \gamma_3\gamma_1\gamma_3\gamma_1 =\therefore= -1 \\
I^2 &= (\gamma_{123})^2 = \gamma_1\gamma_2\gamma_3\gamma_1\gamma_2\gamma_3 =\therefore= -1,
\end{aligned} \tag{3.13}$$

which enables to construct the multiplication table of $\mathcal{G}(\mathbb{R}^3)$ (Table 1). This helps to visualize any subalgebra of $\mathcal{G}(\mathbb{R}^3)$. A special group of subalgebras, the so-called *even-grade subalgebras*, will be necessary during the development of the GAAFs.

Definition 10 (Even-Grade Algebra)**.** A (sub)algebra is said to be even-grade (or simply even), and denoted $\mathcal{G}^+$, if it is composed only by even-grade elements, i.e., scalars ($g = 0$), bivectors ($g = 2$), 4-vectors ($g = 4$), and so on. For instance, a multivector A in the even subalgebra $\mathcal{G}^+(\mathbb{R}^3)$ has the general form

$$A = \langle A\rangle_0 + \langle A\rangle_2, \text{ where } \langle A\rangle_1 = \langle A\rangle_3 = 0. \tag{3.14}$$

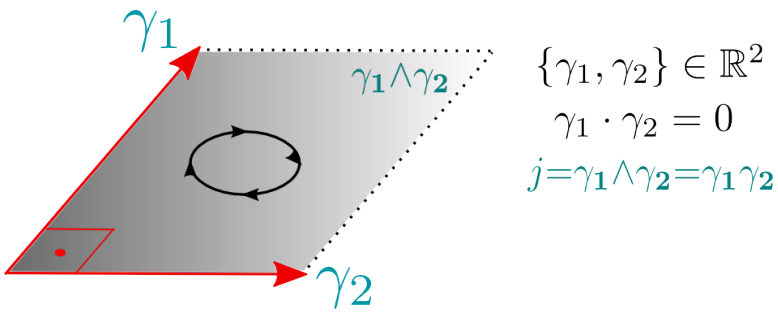


Figure 6: Visualization of the isomorphism with complex algebra.

The even subalgebra $\mathcal{G}^+(\mathbb{R}^n)$ is known as the *algebra of rotors*, i.e., its elements are able to apply n-dimensional rotations to vectors in $\mathbb{R}^n$. □

Remark 3. Similarly, the odd-grade part of an algebra is composed only by odd-grade elements and denoted $\mathcal{G}^-$. For A in $\mathcal{G}^-(\mathbb{R}^3)$, $A = \langle A \rangle_1 + \langle A \rangle_3$, where $\langle A \rangle_0 = \langle A \rangle_2 = 0$. This way, $\mathcal{G}(\mathbb{R}^3) = \mathcal{G}^+(\mathbb{R}^3) + \mathcal{G}^-(\mathbb{R}^3)$. Note that, differently from $\mathcal{G}^+$, $\mathcal{G}^-$ *is not* a subalgebra since it is not closed under the geometric product – it is only a subspace. □

In the sequel, it is shown how the complex numbers and quaternions algebras are obtained from even subalgebras (rotor algebras) of $\mathcal{G}(\mathbb{R}^n)$.

3.3.2 Rotor Algebra of $\mathbb{R}^2$ (Complex Numbers)

The complex-numbers algebra is isomorphic to the even subalgebra $\mathcal{G}^+(\mathbb{R}^2)$, which has basis

$$\{1, \gamma_{12}\}. \tag{3.15}$$

Thus, it is clear that $\mathcal{G}^+(\mathbb{R}^2)$ is also a subalgebra of $\mathcal{G}^+(\mathbb{R}^3)$ (with basis given by (3.12)).

Figure 6 shows the oriented area (bivector) created by the geometric multiplication between γ_1 and γ_2. That area is the visual representation of the pseudovector of $\mathcal{G}^+(\mathbb{R}^2)$, namely γ_{12}. The isomorphism to the complex algebra is established by identifying the imaginary unit j with the pseudovector,

$j = \gamma_{12} = \gamma_1\gamma_2 = \gamma_1 \wedge \gamma_2$. From Table 1 it is known that $(\gamma_{12})^2 = -1$. Then, due to the isomorphism, $j^2 = -1$.

Section 7.2.3 resorts to this isomorphism to test the performance of a GA-based AF which is equivalent to the Complex LMS (CLMS) [32].

3.3.3 Rotor Algebra of $\mathbb{R}^3$ (Quaternions)

The even subalgebra $\mathcal{G}^+(\mathbb{R}^3)$ has basis

$$\{1, \gamma_{12}, \gamma_{23}, \gamma_{31}\}. \tag{3.16}$$

By adopting the following correspondences, $\mathcal{G}^+(\mathbb{R}^3)$ is shown to be isomorphic to quaternion algebra [5, 33]:

$$i \leftrightarrow -\gamma_{12} \qquad j \leftrightarrow -\gamma_{23} \qquad k \leftrightarrow -\gamma_{31}, \tag{3.17}$$

where $\{i, j, k\}$ are the three imaginary unities of quaternion algebra. The minus signs are necessary to make the product between two bivectors equal to the third one and not minus the third, e.g. $(-\gamma_{12})(-\gamma_{23}) = \gamma_{13} = -\gamma_{31}$, just like in quaternion algebra, i.e. $ij = k$, $jk = i$, and $ki = j$ [5]. Again, from Table 1 it is known that $(\gamma_{12})^2 = -1 = i^2$, $(\gamma_{23})^2 = -1 = j^2$, and $(\gamma_{31})^2 = -1 = k^2$.

This algebra is particularly useful in the development of GAAFs for pose estimation (Chapter 6) which are applied in the registration of 3D point clouds. To this end, the rotation operator is defined:

Definition 11 (Rotation operator)**.** Given the vector $x \in \mathbb{R}^n$, a rotated version can be obtained by applying the GA rotation operator $r(\cdot)\widetilde{r}$ to it,

$$x \rightarrow \underbrace{rx\widetilde{r}}_{rotated}, \tag{3.18}$$

where $r \in \mathcal{G}^+(\mathbb{R}^n)$, $\widetilde{r}$ is its reverse[3], and $r\widetilde{r} = 1$, i.e., r is a unit rotor. □

[3]The proper definition of reverse of a (rotor) multivector is given further ahead in (3.24).

The unity constraint is necessary to avoid the rotation operator to scale the vector x, i.e., to avoid changing its norm. A lower case letter was adopted to represent the rotor r (an exception to the convention used in this text – refer to Section 2.1) to avoid ambiguity with rotation matrices, usually represented as R (uppercase).

A rotor $r \in \mathcal{G}^+(\mathbb{R}^n)$ can be generated from the geometric multiplication of two unit vectors in $\mathbb{R}^n$. Given $\{a, b\} \in \mathbb{R}^n$, $|a| = |b| = 1$, with an angle θ between them, and using the geometric product (Definition 5), a rotor can be defined as [3, p. 107]

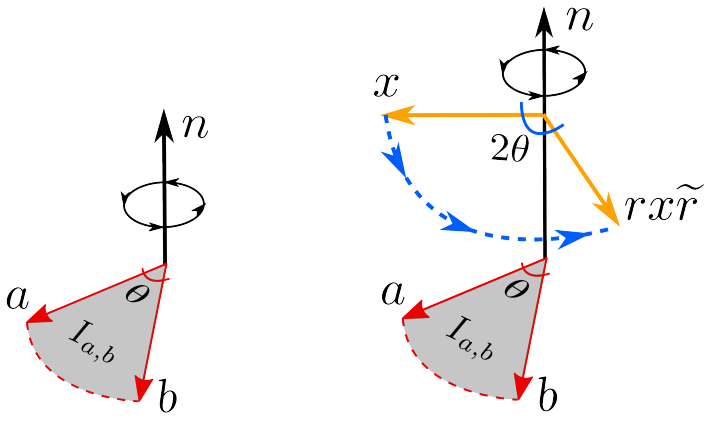
$$\begin{aligned} r &= ab = a \cdot b + a \wedge b \\ &= |a||b|cos\theta + I_{a,b}|a||b|sin\theta \\ &= cos\theta + I_{a,b}sin\theta \\ &= e^{I_{a,b}\theta}, \end{aligned} \tag{3.19}$$

where the definitions of inner product (Definition 3) and outer product (Definition 4) of vectors was used. The result is *the exponential form of a rotor*. Applying (3.19) into (3.18) (see Figure 7), it is possible to show that x is rotated by an angle of 2θ about the normal of the oriented area $I_{a,b}$ (rotation axis) [4]. This way, the structure of a rotor highlights the rotation angle and axis. Similarly, quaternions can be represented in an exponential shape [33, 34].

The rotor r can also be expressed in terms of its coefficients. For the 3D case, $r \in \mathcal{G}^+(\mathbb{R}^3)$, and

$$r = \langle r \rangle + \langle r \rangle_2 = r_0 + r_1\gamma_{12} + r_2\gamma_{23} + r_3\gamma_{31}, \tag{3.20}$$

in which $\{r_0, r_1, r_2, r_3\}$ are the coefficients of r. Note that quaternions, which can also represent rotations in three-dimensional space, have four coefficients as well.



(a) (b)

Figure 7: (a) A rotor can be generated from the geometric multiplication of two unit vectors in $\mathbb{R}^n$. (b) Applying the rotation operator: the vector x is rotated by an angle of 2θ about the normal n of the oriented area $I_{a,b}$.

3.4 Useful Definitions and Properties

This section lists a number of extra definitions and properties of GA which are used throughout the text. They are provided as a consulting list (which is referred to when necessary), and can be skipped at a first reading.

Definition 12 (Inner Product of p-vectors)**.** The inner product of a p-vector $A_p = \langle A \rangle_p$ with a q-vector $B_q = \langle B \rangle_q$ is

$$A_p \cdot B_q = B_q \cdot A_p \triangleq \langle A_p B_q \rangle_{|p-q|}. \tag{3.21}$$

For example, the inner product between a p-vector B and a vector a is $B \cdot a = \langle Ba \rangle_{|p-1|}$. Thus, multiplying a multivector B by a vector reduces its grade by 1. □

Definition 13 (Outer Product of p-vectors)**.** The outer product of a p-vector $A_p = \langle A \rangle_p$ with a q-vector $B_q = \langle B \rangle_q$ is

$$A_p \wedge B_q \triangleq \langle A_p B_q \rangle_{p+q}. \tag{3.22}$$

For example, the outer product between a p-vector B and a vector a is $B \wedge a =$

$\langle Ba\rangle_{|p+1|}$. Thus, multiplying a multivector B by a vector increases its grade by 1. Note that $A_p \wedge B_q \neq B_q \wedge A_p$, i.e., *the outer product is non-commutative.* □

Remark 4. The outer product of a vector a with itself is $a \wedge a \triangleq 0$. Thus, $aa \equiv a \cdot a$. □

Definition 14 (Properties)**.**

$$
\begin{aligned}
&\text{Addition is commutative,}\\
&A + B = B + A.\\
&\text{Multiplication is non-commutative for general multivectors,}\\
&AB \neq BA.\\
&\text{Addition and multiplication are associative,}\\
&(A + B) + C = A + (B + C),\\
&(AB)C = A(BC).\\
&\text{There exist unique additive and multiplicative identities 0 and 1,}\\
&A + 0 = A,\\
&1A = A.\\
&\text{Every mutivector has a unique additive inverse } -A,\\
&A + (-A) = 0.
\end{aligned}
\tag{3.23}
$$

□

Definition 15 (Reversion)**.** The reverse of a multivector A is defined as

$$\widetilde{A} \triangleq \sum_{g=0}^{n} (-1)^{g(g-1)/2} \langle A\rangle_g. \tag{3.24}$$

For example, the reverse of a 2-vector $A = \langle A\rangle_0 + \langle A\rangle_1 + \langle A\rangle_2$ is $\widetilde{A} = \widetilde{\langle A\rangle_0} + \widetilde{\langle A\rangle_1} + \widetilde{\langle A\rangle_2} = \langle A\rangle_0 + \langle A\rangle_1 - \langle A\rangle_2$. The reversion operation of GA is the extension of the complex conjugate in linear algebra. □

Remark 5. Note that since the 0-grade of a multivector is not affected by reversion, mutually reverse multivectors, say A and $\widetilde{A}$, have the same 0-grade,

$\langle A\rangle_0 = \langle \widetilde{A}\rangle_0$.

Definition 16 (Scalar Product)**.** The scalar product between two multivectors is

$$A * B \triangleq \langle AB\rangle, \tag{3.25}$$

i.e., it is the scalar part (0-grade) of the geometric multiplication between A and B. For the special case of vectors, $a * b = \langle ab\rangle = a \cdot b$. □

Definition 17 (Magnitude)**.**

$$|A| \triangleq \sqrt{A * \widetilde{A}} = \sqrt{\sum_g |\langle A\rangle_g|^2}. \tag{3.26}$$

□

Definition 18 (Cyclic reordering)**.** The scalar part of a product of two multivectors is order invariant. This way,

$$\langle AB\rangle = \langle BA\rangle \Rightarrow \langle AB\cdots C\rangle = \langle B\cdots CA\rangle. \tag{3.27}$$

□

Remark 6. From that it follows that the scalar product is commutative, $A * B = \langle AB\rangle = \langle BA\rangle = B * A$. □

Definition 19 (Pseudoscalar)**.** The pseudoscalar I is the highest grade of an algebra $\mathcal{G}$. In $3D$ Euclidean space,

$$I \triangleq a \wedge b \wedge c, \tag{3.28}$$

in which $\{a, b, c\}$ are linearly-independent vectors in $\mathcal{G}$. I commutes with any multivector in $\mathcal{G}$, hence the name pseudoscalar. □

Definition 20 (Inversion)**.** Every nonzero vector a has a multiplicative inverse defined as [4],

$$a^{-1} \triangleq \frac{a}{a^2}, \quad a^2 \neq 0 \Rightarrow aa^{-1} = a\frac{a}{a^2} = 1. \tag{3.29}$$

Definition 21 (Versor)**.** A multivector A that can be factored into a product of n vectors

$$A = a_1 a_2 \cdots a_n, \tag{3.30}$$

is called *versor*. Moreover, if the vectors $\{a_1, a_2, \cdots, a_n\}$ are invertible it is possible to show that A has a multiplicative inverse $A^{-1} \triangleq a_n^{-1} \cdots a_2^{-1} a_1^{-1}$. For a detailed explanation, please refer to [5, Eq.(25)] and [3, pp.103]. □

Definition 22 (Frame and Reciprocal Frame)**.** A set of vectors $\{a_1, a_2, \cdots, a_n\}$ defining a geometric algebra $\mathcal{G}$ is said to be a *frame* if and only if $A_n = a_1 \wedge a_2 \wedge \cdots \wedge a_n \neq 0$. This is equivalent to saying $\{a_1, a_2, \cdots, a_n\}$ are *linearly independent*.

Given a frame $\{a_1, a_2, \cdots, a_n\}$, it is possible to obtain a *reciprocal frame* [3] $\{a^1, a^2, \cdots, a^n\}$ via the equations $a_k a^j = \delta_{k,j},\ j, k = 1, 2, \cdots, n,$ where $\delta_{k,j} = 1,$ for $j = k$, is the *Kronecker delta*. □

Definition 23 (Decomposition into Grades)**.** According to Definition 8, given the frame $\{a_1, a_2, \cdots, a_n\}$, a basis for the geometric algebra $\mathcal{G}(I)$,[4] in which $I = a_1 \wedge a_2 \wedge \cdots \wedge a_n$ is its pseudovector, can be constructed by geometrically multiplying the elements of the frame. The resulting 2^n members (multivectors) of the basis are grouped like $\{\alpha_1, \alpha_2, \cdots, \alpha_{2^n}\}$, where $\alpha_{2^n} = I$. The same procedure can be adopted for a reciprocal frame $\{a^1, a^2, \cdots, a^n\}$, originating a reciprocal basis for $\mathcal{G}(I)$, $\{\alpha^1, \alpha^2, \cdots, \alpha^{2^n}\}$, where $\alpha^{2^n} = I$. From that, any multivector $B \in \mathcal{G}(I)$ can be decomposed into its grades like [3]

$$B = \sum_K \alpha^K (\alpha_K * B) = \sum_K \alpha^K \langle \alpha_K B \rangle, \quad K = 1, \cdots, 2^n. \tag{3.31}$$

This procedure will be very useful when performing geometric calculus operations.

□

[4] $\mathcal{G}(I)$ is adopted in the literature to denote *the geometric algebra whose pseudovector is* I. In fact, since I results from the geometric multiplication of the elements in the basis of the underlying vector space $\mathcal{V}$, the forms $\mathcal{G}(\mathcal{V})$ and $\mathcal{G}(I)$ are equivalent. See [3, p.19].

3.5 Geometric Calculus

The Geometric Calculus generalizes the standard concepts of calculus to encompass the GA theory. In the sequel, some basic relations are defined in order to be promptly used in the design of the AFs. For a detailed discussion on the subject, please refer to [3, 35].

Definition 24 (Differential operator)**.** The differential operator ∂ (also used throughout this work in the form ∇) has the algebraic properties of any other multivector in $\mathcal{G}(I)$ [3]. Thus, it can be decomposed into its grades by applying Definition 23

$$\partial = \sum_K a^K (a_K * \partial). \tag{3.32}$$

Whenever necessary, the differential operator will present a subscript indicating the variable (multivector) with respect to the derivation is performed. For instance, ∂_X is a derivative with respect to the multivector X. □

Definition 25 (Differential or A-derivative)**.** Let $F = F(X)$ be a function defined on $\mathcal{G}(I)$,

$$F : X \in \mathcal{G}(I) \to F(X) \in \mathcal{G}(I),$$

where $I = \langle I \rangle_n$ is a unit pseudovector (i.e. unit magnitude). Then the *differential* or *A-derivative* is defined by

$$A * \partial_X F(X) = (A * \partial_X) F(X) \triangleq \partial_\tau F(X + \tau A)|_{\tau=0} = \lim_{\tau \to 0} \frac{F(X + \tau A) - F(X)}{\tau}, \tag{3.33}$$

in which $A * \partial_X$ is called *scalar differential operator.* □

Definition 26 (Differential and overdot notation)**.** Let $F = F(X) \in \mathcal{G}(I)$. Given the product of two general multivectors AX, in which $A = F(X)$, the following notation

$$\dot{\partial_X}(A\dot{X}) \tag{3.34}$$

indicates that only X is to be differentiated [5, 35]. This is particularly useful to

circumvent the limitations imposed by the noncommutativity of GA: note that since the differential operator has the algebraic properties of a multivector in $\mathcal{G}(I)$, one *cannot simply assume* $\partial_X AX = A\partial_X X$. Recall that, in general, $\partial_X A \neq A\partial_X$ (Definition 14). Thus, the overdot notation provides a way to comply with the noncommutativity of multivectors with respect to the geometric product. □

Proposition 1 (Basic Multivector Differential)**.** *Given two multivectors X and A, it holds [16]*

$$(A * \partial_X)X = \dot{\partial}_X(\dot{X} * A) = \dot{\partial}_X\langle \dot{X}A\rangle = A. \tag{3.35}$$

Proof.

$$\begin{aligned}(A * \partial_X)X &= \lim_{\tau\to 0}\frac{(X+\tau A)-X}{\tau}\\ &= \lim_{\tau\to 0}\frac{\tau A}{\tau}\\ &= A.\end{aligned} \tag{3.36}$$

□

Remark 7. By similar means, it can be shown that the following relation holds: $(A * \partial_X)\widetilde{X} = \dot{\partial}_X\langle \dot{\widetilde{X}}A\rangle = \widetilde{A}$.

Definition 27 (Laplacian)**.** The *Laplacian* (second derivative) is defined as,

$$\partial_X^2 = \partial_X * \partial_X. \tag{3.37}$$

Thus, it is a scalar differential operator (See Definition 25), $(A * \partial_X)$, where $A = \partial_X$. □

Definition 28 (Product Rule)**.** Given two general multivectors A and B, the multivector derivative of the product AB is defined by the following rule [35, Eq.5.12]

$$\partial(AB) = \dot{\partial}\dot{A}B + \dot{\partial}A\dot{B}. \tag{3.38}$$

□

Proposition 2 (Doran's Relation)**.**

$$\dot{\partial}_\Omega \langle A \dot{\widetilde{\Omega}} \rangle = -\widetilde{\Omega} A \widetilde{\Omega}, \tag{3.39}$$

where A is a general multivector and Ω is a unit rotor.

Proof. Given that the scalar part (0-grade) of a multivector is not affected by rotation, and using the product rule (Definition 28), one can write

$$\partial_\Omega \langle \Omega A \widetilde{\Omega} \rangle = A \widetilde{\Omega} + \dot{\partial}_\Omega \langle \Omega A \dot{\widetilde{\Omega}} \rangle = 0. \tag{3.40}$$

Using the scalar product (Definition 16) and Proposition 1,

$$\Rightarrow \dot{\partial}_\Omega \langle \Omega A \dot{\widetilde{\Omega}} \rangle = \dot{\partial}_\Omega (\dot{\widetilde{\Omega}} * \Omega A) = (\Omega A) * \partial_\Omega \widetilde{\Omega}. \tag{3.41}$$

Plugging back into (3.40) and multiplying by $\widetilde{\Omega}$ from the left,

$$\begin{aligned} (\Omega A) * \partial_\Omega \widetilde{\Omega} &= -A \widetilde{\Omega}, \\ \underbrace{(\widetilde{\Omega} \Omega)}_{=1} A * \partial_\Omega \widetilde{\Omega} &= -\widetilde{\Omega} A \widetilde{\Omega}, \\ \dot{\partial}_\Omega (A * \dot{\widetilde{\Omega}}) &= -\widetilde{\Omega} A \widetilde{\Omega}, \\ \dot{\partial}_\Omega \langle A \dot{\widetilde{\Omega}} \rangle &= -\widetilde{\Omega} A \widetilde{\Omega}, \end{aligned} \tag{3.42}$$

in which Proposition 1 was employed once more. This relation was first presented in [13] with no clear proof. □

4 LINEAR ESTIMATION IN GA

This chapter shows how one can use GA to address linear estimation. Key differences between GA-based and LA-based formulations are highlighted. In particular, the concepts of *random multivectors* and *array of multivectors* are introduced in order to support the derivation and performance analysis of the GAAFs (Chapters 5 and 6).

The following LA minimization (least-squares) problem will be utilized to motivate the transition from LA to GA,

$$\min \left\| d - \hat{d} \right\|^2, \tag{4.1}$$

in which $\{d, \hat{d}\} \in \mathbb{R}^n, n = \{1, 2, \cdots\}$ and $\hat{d}$ is the estimate for d.

To formulate (4.1) in the GA framework, the concepts of multivectors (Definition 29) and arrays of multivectors (Definition 30) are used. This way, as shown further ahead, the GA version of (4.1) offers a way to extend that minimization problem for hypercomplex quantities.

Two especial cases of (4.1) are studied regarding the way d and $\hat{d}$ are defined:

1. In this case, d is defined according to (2.1) and $\hat{d} = u^* w$, in which u and w are $M \times 1$ arrays of multivectors, the regressor and the weight arrays, respectively, and * denotes the reverse array (see ahead Definition 31). The estimate for d is obtained from a collection of M input samples (regressor). Such a way of defining $\hat{d}$ is widely employed across adaptive filtering

literature [1, 6];

2. In this case, $d \in \mathbb{R}^n$ is the resulting vector after applying an unknown rigid geometric transformation (rotation and translation) to $x \in \mathbb{R}^n$. Thus, $\hat{d} = Rx+t$, where R represents an $n\times n$ rotation matrix, t an $n\times 1$ translation vector, provides an estimate of the actual rotation and translation applied to x.

From the above cases, two GA-based minimization cost functions (CFs) are generated: one for estimating the coefficients of w, which generates an estimate for d (called from here on the *standard cost function*); and one for estimating the rigid transformation (rotation and translation) that should be applied to x in order to align it with d (*pose estimation cost function*). Each of those forms is better suited for a specific type of application: the standard CF is connected to the system identification problem (see Section 7.2) and the pose estimation CF is related to the 3D registration of point clouds (see Section 7.3).

4.1 Useful Definitions

Some definitions are necessary before stating the general GA cost function. In the first one, the concept of random variable is simply extrapolated to allow for hypercomplex random quantities,

Definition 29 (Random Multivectors)**.** A *random multivector* is defined as a multivector whose grade values are random variables. Take for instance the following random multivector in $\mathcal{G}(\mathbb{R}^3)$ (the GA formed by the vector space $\mathbb{R}^3$)

$$\begin{aligned} \boldsymbol{A} &= \langle \boldsymbol{A} \rangle_0 + \langle \boldsymbol{A} \rangle_1 + \langle \boldsymbol{A} \rangle_2 + \langle \boldsymbol{A} \rangle_3 \\ &= \boldsymbol{a}_0 + \boldsymbol{a}_1\gamma_1 + \boldsymbol{a}_2\gamma_2 + \boldsymbol{a}_3\gamma_3 + \boldsymbol{a}_4\gamma_{12} + \boldsymbol{a}_5\gamma_{23} + \boldsymbol{a}_6\gamma_{31} + \boldsymbol{a}_7 I. \end{aligned} \tag{4.2}$$

The terms $\boldsymbol{a}_0, \cdots, \boldsymbol{a}_7$ are *real-valued* random variables, i.e., they are drawn from a stochastic process described by a certain probability density function with a

mean and a variance ([1, Chapter A]). Note that random multivectors/variables are denoted in boldface letters throughout the whole text. □

The next definition introduces the concept of arrays of multivectors,

Definition 30 (Arrays of Multivectors)**.** An *array of multivectors* is a collection of general multivectors. Given M multivectors $\{U_1, U_2, \cdots, U_M\}$ in $\mathcal{G}(\mathbb{R}^3)$, the $M \times 1$ array collects them as follows

$$u = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_M \end{bmatrix} = \begin{bmatrix} u_{10} + u_{11}\gamma_1 + u_{12}\gamma_2 + u_{13}\gamma_3 + u_{14}\gamma_{12} + u_{15}\gamma_{23} + u_{16}\gamma_{31} + u_{17}I \\ u_{20} + u_{21}\gamma_1 + u_{22}\gamma_2 + u_{23}\gamma_3 + u_{24}\gamma_{12} + u_{25}\gamma_{23} + u_{26}\gamma_{31} + u_{27}I \\ \vdots \\ u_{M0} + u_{M1}\gamma_1 + u_{M2}\gamma_2 + u_{M3}\gamma_3 + u_{M4}\gamma_{12} + u_{M5}\gamma_{23} + u_{M6}\gamma_{31} + u_{M7}I \end{bmatrix}. \tag{4.3}$$

The array is denoted using lower case letters, the same as scalars and vectors (1-vectors). However, the meaning of the symbol will be evident from the context. Also, the name *array* was chosen to avoid confusion with vectors (1-vectors) in $\mathbb{R}^n$, which in this text have the usual meaning of *collection of real numbers*. In this sense, an array of multivectors can be interpreted as a "vector" that allows for hypercomplex entries.

Array u in (4.3) can be rewritten to highlight its grades,

$$u = \begin{bmatrix} u_{10} \\ u_{20} \\ \vdots \\ u_{M0} \end{bmatrix} + \begin{bmatrix} u_{11} \\ u_{21} \\ \vdots \\ u_{M1} \end{bmatrix} \gamma_1 + \cdots + \begin{bmatrix} u_{17} \\ u_{27} \\ \vdots \\ u_{M7} \end{bmatrix} I. \tag{4.4}$$

Finally, there are also arrays of random multivectors,

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_2 \\ \vdots \\ \boldsymbol{U}_M \end{bmatrix}, \tag{4.5}$$

which of course are denoted using boldface type. □

Next, the reverse array is defined,

Definition 31 (Reverse Array)**.** The *reverse array* is the extension of the reverse operation of multivectors to include arrays of multivectors. Given the array u in (4.3), its reverse version, denoted by the symbol *, is

$$u^* = \left[\widetilde{U}_1 \; \widetilde{U}_2 \; \cdots \; \widetilde{U}_M\right]. \tag{4.6}$$

Note that the entries in u^* are the reverse counterparts of the entries in u. □

Now the product between arrays is defined,

Definition 32 (Array Product)**.** Given two $M \times 1$ arrays of multivectors, u and w, the product between them is defined as

$$u^T w = U_1 W_1 + U_2 W_2 + \cdots U_M W_M, \tag{4.7}$$

in which ^{T} represents the transpose array. The underlying product in each of the terms $U_j W_j$, $j = \{1, \cdots, M\}$, is the geometric product. Thus, the array product $u^T w$ results in the general multivector $\sum_{j=1}^{M} U_j W_j$. In a similar fashion,

$$u^* w = \sum_{j=1}^{M} \widetilde{U}_j W_j, \tag{4.8}$$

where * represents the reverse array.

Observe that due to the noncommutativity of the geometric product, $u^T w \neq w^T u$ in general. □

Remark 8. This text adopts the following notation to represent a product between an array and itself: given the array u, $\|u\|^2 \triangleq u^* u$. Note this is the same notation employed to denote the squared norm of a vector in $\mathbb{R}^n$ in linear algebra. However, here $\|u\|^2$ *is a general multivector*, i.e., *it is not a pure scalar value* which in linear algebra provides a measure of distance. In GA, the distance

metric is given by the *magnitude of a multivector* (see Definition 17), which is indeed a scalar value. Thus, for an array u and a multivector U,

$$\begin{aligned} \|u\|^2 &= u^* u : \text{is a multivector} \\ |U|^2 &= U * \widetilde{U} = \widetilde{U} * U = \textstyle\sum_g |\langle U \rangle_g|^2 : \text{is a scalar.} \end{aligned} \tag{4.9}$$

Finally, note that $\widetilde{\|u\|^2} = \widetilde{(u^* u)} = u^* u = \|u\|^2$, i.e., $\|u\|^2$ is equal to its own reverse. □

Definition 33 (Product Between Multivector and Array)**.** Here the multivector U is simply geometricaly multiplicated with each entry of the array w. Due to the noncommutativity of the geometric product, two cases have to be considered. The first is Uw,

$$Uw = U \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_M \end{bmatrix} = \begin{bmatrix} UW_1 \\ UW_2 \\ \vdots \\ UW_M \end{bmatrix}, \tag{4.10}$$

and the second is wU,

$$wU = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_M \end{bmatrix} U = \begin{bmatrix} W_1 U \\ W_2 U \\ \vdots \\ W_M U \end{bmatrix}. \tag{4.11}$$

□

With the previous definitions, the general GA cost function can be formulated.

4.2 General Cost Function in GA

Following the guidelines in [3, p.64 and p.121], one can formulate a minimization problem by defining a general CF in GA. The following CF is a "mother"

cost function, able to encompass the two cases aforementioned (standard form and pose estimation),

$$J(D, A_k, X, B_k) = \left| D - \sum_{k=1}^{M} A_k X B_k \right|^2 , \tag{4.12}$$

where D, X, A_k, B_k are general multivectors. The term $\sum_{k=1}^{M} A_k X B_k$ represents the canonical form of a linear transformation applied to the multivector X ([3, p.64 and p.121]). For the two applications of interest in this text (system identification and pose estimation), the goal is to change the variables A_k, B_k and X in order to minimize the squared magnitude (see Definition 17) of the error $D - \sum_{k=1}^{M} A_k X B_k$.

In the sequel, it will be shown how to retrieve the standard and pose estimation CFs from (4.12).

4.2.1 The Standard Shape

The standard cost function (least-squares) J_s is obtained from (4.12) by making $D = d$ (a general multivector), $X = 1$, $A_k = \widetilde{U}_k$, $B_k = W_k$,

$$J_s(w) = \left| d - \sum_{k=1}^{M} \widetilde{U}_k W_k \right|^2 = \left| d - u^* w \right|^2 , \tag{4.13}$$

where M is the system order (the number of taps in the filter), and the definition of array product (4.8) was employed to make $\sum_{k=1}^{M} \widetilde{U}_k W_k = u^* w$. Note that a lower case letter was adopted to represent the general multivector d (an exception to the convention used in this text). This is done to emphasize the shape similarity to the usual cost function $\left\| d - u^H w \right\|^2$ used in system identification applications in terms of a scalar d and vectors u and w, with ^{H} denoting the Hermitian conjugate [1,6]. Similarly to its linear-algebra counterpart, d is estimated as a linear combination of the entries of the regressor u, which are random multivectors. Thus, the error

quantity to be minimized is defined as

$$e = d - u^* w. \tag{4.14}$$

The performance analysis of the GAAFs (Section 5.3) requires the *least mean-squares* counterpart of (4.13),

$$\boxed{J_s(w) = \mathrm{E}\,|\boldsymbol{e}|^2 = \mathrm{E}\,|\boldsymbol{d} - \boldsymbol{u}^* w|^2}, \tag{4.15}$$

in which $\boldsymbol{e}$ and $\boldsymbol{d}$ are random multivectors, $\boldsymbol{u}^*$ is an $M \times 1$ array of multivectors, and E is the expectation operator. Notice that (4.15) has the exact same shape as the least-mean squares cost function used in linear algebra-based adaptive filtering [1, 6].

It will be shown in Chapter 5 how to devise the *standard* GAAFs, which are able to minimize (4.15). In fact, the GAAFs are derived from the *steepest-descent recursion*, which iteratively minimizes (4.15) providing the instantaneous cost

$$\boxed{J_s(i) = \mathrm{E}\,|\boldsymbol{d} - \boldsymbol{u}^* w_{i-1}|^2}. \tag{4.16}$$

$J_s(i) \triangleq J_s(w_{i-1})$ is the *learning curve* associated to the cost function $J_s(w)$ (see [1, Chapter 9]).

4.2.2 The Pose-Estimation Shape

The pose-estimation cost function J_p is obtained from (4.12) by making $M = 1$, $D = y$, $X = x$, $A_k = r$, $B_k = \widetilde{r}$,

$$J_p(r) = |y - rx\widetilde{r}|^2, \text{ subject to } r\widetilde{r} = \widetilde{r}r = 1, \tag{4.17}$$

where y and x are vectors in $\mathbb{R}^n$, and r is a rotor in $\mathcal{G}(\mathbb{R}^n)$, with $\widetilde{r}$ denoting its reversed version (see Definition 15). As pointed out in Definition 11, the term $rx\widetilde{r}$ is a rotated version of vector x, and the constraint $r\widetilde{r} = \widetilde{r}r = 1$ means r is a

unit rotor. Thus, the error quantity to be minimized is defined as

$$e = y - rx\widetilde{r}. \tag{4.18}$$

The least-mean squares counterpart of (4.17) is

$$\boxed{J_p(r) = \mathrm{E}\left|\boldsymbol{e}\right|^2 = \mathrm{E}\left|\boldsymbol{y} - r\boldsymbol{x}\widetilde{r}\right|^2, \text{ subject to } r\widetilde{r} = \widetilde{r}r = 1}, \tag{4.19}$$

where $\boldsymbol{e}$, $\boldsymbol{y}$ and $\boldsymbol{x}$ are random vectors and E is the expectation operator.

The minimization of cost function (4.19) consists in finding the rotor r that rotates x in order to align it with y. This is one of the requirements in the process of estimating rigid transformations in a given vector space. In particular, the 3D registration of point clouds is formulated in GA by resorting to (4.19). Chapter 6 shows how to devise the *pose-estimation* GAAFs, which are able to minimize (4.19).

Similarly to what is done in Section 4.2.1, the learning curve associated to (4.19) is defined as $J_p(i) \triangleq J_p(r_{i-1})$,

$$\boxed{J_p(i) = \mathrm{E}\left|\boldsymbol{y} - r_{i-1}\boldsymbol{x}\widetilde{r}_{i-1}\right|^2, \text{ subject to } r_{i-1}\widetilde{r}_{i-1} = \widetilde{r}_{i-1}r_{i-1} = 1}. \tag{4.20}$$

5 GEOMETRIC-ALGEBRA ADAPTIVE FILTERS (STANDARD)

In this chapter, the GAAFs are motivated following a least-squares approach, deriving the GA-LMS to minimize the cost function (4.13) in an adaptive manner. In the sequel, by modeling the observed data $d(i)$ and u_i as stochastic processes, a mean-square analysis (steady-state) is performed.

The GAAFs to be designed must provide an estimate for the array of multivectors w via a recursive rule of the form,

$$w_i = w_{i-1} + \mu G, \tag{5.1}$$

where i is the (time) iteration, μ is the AF step size, and G is a multivector valued quantity related to the estimation error (4.14).

A proper selection of G is required to make $J(w_i) < J(w_{i-1})$ at each iteration. This chapter adopts the steepest-descent rule [1, 6] and the analytical guidelines of [36], in which the AF is designed to follow the opposite direction of the gradient of the cost function, namely $\partial_w J(w_{i-1})$. This way, G is proportional to $\partial_w J(w_{i-1})$,

$$G \triangleq -B\partial_w J(w_{i-1}), \tag{5.2}$$

what yields the general form of an AF,

$$w_i = w_{i-1} - \mu B\partial_w J(w_{i-1}), \tag{5.3}$$

in which B is a general multivector, in contrast with the standard case in which

B would be a matrix [1]. Choosing B appropriately is a requirement to define the type of adaptive algorithm, what is detailed in the following subsections.

5.1 GA Least-Mean Squares (GA-LMS)

The GA-LMS is supposed to adaptively minimize the cost function (4.13), reproduced below for ease of reference

$$J_s(w_{i-1}) = |d(i) - u_i^* w_{i-1}|^2 = |e(i)|^2 . \tag{5.4}$$

Writing (5.4) *in terms of its grades* allows for applying GC in order to derive the GAAFs further ahead. This way

$$\begin{aligned} J(w_{i-1}) = |e(i)|^2 = e(i) * \widetilde{e}(i) \quad &= \textstyle\sum_{A=1}^{2^n} \gamma_A e_A * \sum_{A=1}^{2^n} e_A \widetilde{\gamma}_A \\ &= \textstyle\sum_{A=1}^{2^n} e_A^2, \end{aligned} \tag{5.5}$$

where

$$e_A = d_A - \hat{d}_A. \tag{5.6}$$

To move on to the calculation of the gradient of $J(w_{i-1})$ (required to obtain the GA-LMS AF), it is necessary to find an expression for $\hat{d}_A$ as a function of the grades of $u_i^* w_{i-1}$. Defining $\hat{d}(i) \triangleq u_i^* w_{i-1}$ (a multivector resultant from an array product) and using (3.31), $\hat{d}(i)$ can be written as

$$\hat{d}(i) = u_i^* w_{i-1} = \sum_{A=1}^{2^n} \gamma_A \langle \widetilde{\gamma}_A (u_i^* w_{i-1}) \rangle. \tag{5.7}$$

Since u_i and w_{i-1} are arrays with M multivector (Clifford numbers) entries, they can be written in terms of 2^n grades of M-dimensional arrays with real entries

$$u_i^* = \sum_{A=1}^{2^n} \langle u_i^* \gamma_A \rangle \widetilde{\gamma}_A = \sum_{A=1}^{2^n} u_{i,A}^T \widetilde{\gamma}_A, \tag{5.8}$$

and

$$w_{i-1} = \sum_{A=1}^{2^n} \gamma_A \langle \widetilde{\gamma}_A w_{i-1} \rangle = \sum_{A=1}^{2^n} \gamma_A w_{i-1,A}, \tag{5.9}$$

where $u_{i,A}^T$ and $w_{i-1,A}$ are respectively $1 \times M$ and $M \times 1$ arrays with real entries. Also, (3.31) was utilized once more. Plugging (5.8) and (5.9) back into (5.7)[1]

$$\begin{aligned} \hat{d}(i) = u_i^* w_{i-1} &= \textstyle\sum_{A=1}^{2^n} \gamma_A \langle \widetilde{\gamma}_A (u_i^* w_{i-1}) \rangle \\ &= \textstyle\sum_{A=1}^{2^n} \gamma_A \langle \widetilde{\gamma}_A (\sum_{B=1}^{2^n} u_B^T \widetilde{\gamma}_B \sum_{C=1}^{2^n} \gamma_C w_C) \rangle \\ &= \textstyle\sum_{A=1}^{2^n} \gamma_A \sum_{B,C=1}^{2^n} \langle \widetilde{\gamma}_A (u_B^T \widetilde{\gamma}_B \gamma_C w_C) \rangle \\ &= \textstyle\sum_{A=1}^{2^n} \gamma_A \sum_{B,C=1}^{2^n} \langle \widetilde{\gamma}_A \widetilde{\gamma}_B \gamma_C \rangle (u_B^T \cdot w_C) \\ &= \textstyle\sum_{A=1}^{2^n} \gamma_A \hat{d}_A, \end{aligned} \tag{5.10}$$

in which

$$\hat{d}_A = \sum_{B,C=1}^{2^n} \langle \widetilde{\gamma}_A \widetilde{\gamma}_B \gamma_C \rangle (u_B^T \cdot w_C), \ A = 1, \cdots, 2^n \tag{5.11}$$

is the expression of $\hat{d}_A$ as a function of the grades of $u_i^* w_{i-1}$.

The last step before performing the actual gradient calculation is to define the *multivector derivative* with respect to w in terms of its grades (see Definition 24)

$$\partial_w \triangleq \sum_{A=1}^{2^n} \gamma_A \langle \widetilde{\gamma}_A \partial_w \rangle = \sum_{A=1}^{2^n} \gamma_A \partial_{w,A}. \tag{5.12}$$

This is the case since the differential operator has the algebraic properties of a multivector in $\mathcal{G}(\mathbb{R}^n)$ ([3, p.45]).

With all the previous quantities (multivectors and arrays) described in terms of their GA grades, the gradient calculation is performed as follows

$$\partial_w J(w_{i-1}) = \left(\sum_{D=1}^{2^n} \gamma_D \partial_{w,D} \right) \left(\sum_{A=1}^{2^n} e_A^2 \right) = \sum_{A,D=1}^{2^n} \gamma_D \partial_{w,D} e_A^2, \tag{5.13}$$

[1] From now on, the iteration subscripts i and $i-1$ are omitted from $u_{i,A}$ and $w_{i-1,A}$ for clarity purposes.

in which

$$\begin{aligned}\partial_{w,D}e_A^2 = 2e_A(\partial_{w,D}e_A) &= 2e_A(\partial_{w,D}(d_A - \hat{d}_A)) \\ &= -2e_A(\partial_{w,D}\hat{d}_A),\end{aligned} \tag{5.14}$$

where $\partial_{w,D}d_A = 0$ since d_A does not depend on the weight vector w. Plugging (5.14) into (5.13) results in

$$\partial_w J(w_{i-1}) = -2\sum_{A,D=1}^{2^n} \gamma_D e_A(\partial_{w,D}\hat{d}_A). \tag{5.15}$$

Using (5.11) to rewrite $\partial_{w,D}\hat{d}_A$ yields

$$\begin{aligned}\partial_{w,D}\hat{d}_A &= \partial_{w,D}\left[\textstyle\sum_{B,C=1}^{2^n}\langle\widetilde{\gamma}_A\widetilde{\gamma}_B\gamma_C\rangle(u_B^T\cdot w_C)\right] \\ &= \textstyle\sum_{B,C=1}^{2^n}\langle\widetilde{\gamma}_A\widetilde{\gamma}_B\gamma_C\rangle\partial_{w,D}(u_B^T\cdot w_C).\end{aligned} \tag{5.16}$$

Now it is important to notice that the term $\partial_{w,D}(u_B^T\cdot w_C)$ will be different from zero only when $D = C$, i.e., when $\partial_{w,D}$ and w_C are of same grade – recall that ∂_w has the same algebraic properties as a multivector in $\mathcal{G}(\mathbb{R}^n)$. This way, $\partial_{w,D}(u_B^T\cdot w_C) = u_B$ for $D = C$, or adopting the *Kronecker delta function* δ_{BC} [36]

$$\partial_{w,D}(u_B^T\cdot w_C) = \delta_{CD}u_B^T. \tag{5.17}$$

Plugging it back into (5.16) results in

$$\partial_{w,D}\hat{d}_A = \sum_{B,C=1}^{2^n}\langle\widetilde{\gamma}_A\widetilde{\gamma}_B\gamma_C\rangle\delta_{CD}u_B^T. \tag{5.18}$$

Finally, substituting (5.18) into (5.15), the gradient is obtained

$$\begin{aligned}\partial_w J(w_{i-1}) &= -2\textstyle\sum_{A,D=1}^{2^n}\gamma_D e_A\sum_{B,C=1}^{2^n}\langle\widetilde{\gamma}_A\widetilde{\gamma}_B\gamma_C\rangle\delta_{CD}u_B^T \\ &= -2\textstyle\sum_{A,D=1}^{2^n}e_A\sum_{B=1}^{2^n}\gamma_D\langle\widetilde{\gamma}_A\widetilde{\gamma}_B\gamma_D\rangle u_B^T \\ &= -2\textstyle\sum_{A,D=1}^{2^n}e_A\gamma_D\langle\widetilde{\gamma}_A u_i^*\gamma_D\rangle \\ &= -2\textstyle\sum_{A,D=1}^{2^n}e_A\gamma_D\langle\widetilde{\gamma}_D u_i\gamma_A\rangle \\ &= -2\textstyle\sum_{A=1}^{2^n}e_A u_i\gamma_A = \boxed{-2u_i e(i)}.\end{aligned} \tag{5.19}$$

In the AF literature, setting B equal to the identity matrix in (5.3) (the general form of an AF) results in the steepest-descent update rule ([1, Eq.8-19]). In GA though, the multiplicative identity is the multivector (scalar) 1 (see Definition 14). This way, substituting (5.19) into (5.3) and setting $B = 1$ yields the GA-LMS update rule

$$\boxed{w_i = w_{i-1} + \mu u_i e(i)}, \tag{5.20}$$

where the 2 in (5.19) was absorbed by the step size μ.

Note that the GA-LMS (5.20) has the same shape of the regular LMS AFs [1, 6], namely the *real-valued* LMS (u and w have real-valued entries) and the *complex-valued* LMS (u and w have complex-valued entries). However, in the previous derivation, no constraints were put on the entries of the arrays u and w – they can be any kind of multivector. This way, the update rule (5.20) is valid for any u and w whose *entries are general multivectors* in $\mathcal{G}(\mathbb{R}^n)$. In other words, the update rule (5.20) generalizes the standard LMS AF for several types of u and w entries: general multivectors, rotors, quaternions, complex numbers, real numbers – *any subalgebra* of $\mathcal{G}(\mathbb{R}^n)$.

This is a very interesting result, accomplished due to the comprehensive analytic tools provided by Geometric Calculus. Recall that, in adaptive filtering theory, the transition from real-valued AFs to complex-valued AFs requires one to abide by the rules of differentiation with respect to a complex variable, represented by the *Cauchy-Riemann conditions* (see [1, Chapter C, p.25]). Similarly, quaternion-valued AFs require further differentiation rules that are captured by the *Hamilton-real* (HR) calculus [26,27,29] and its generalized version (GHR) [31]. Although those approaches are successful, note that each time the underlying algebra is changed, the analytic tools need an update as well. This is not the case if one resorts to GA and GC to address the minimization problem. In this sense,

GC proves itself as an extremely versatile analytic tool, providing a simple and unique way to perform calculus in any subalgebra of $\mathcal{G}(\mathbb{R}^n)$.

5.2 Data Model in GA

In order to carry on performance analyses of the GAAFs, this work adopts an specific data model. The goal of the analysis is to derive an expression for the mean-square error (MSE) in steady-state of standard GAAFs via energy conservation relations (ECR) [1].

Definition 34 (Steady-State MSE in GA)**.** As in linear algebra, the steady-state MSE in GA must be *scalar-valued*. To this end, the MSE is defined as

$$\text{MSE} = \xi \triangleq \lim_{i\to\infty} \text{E}\Big\langle \|\boldsymbol{e}(i)\|^2 \Big\rangle = \lim_{i\to\infty} \Big\langle \text{E}\,\|\boldsymbol{e}(i)\|^2 \Big\rangle, \tag{5.21}$$

i.e., it involves the calculation of the scalar part (0-grade) of the multivector $\|\boldsymbol{e}(i)\|^2 = \widetilde{\boldsymbol{e}}(i)\boldsymbol{e}(i)$. □

The ECR technique is an energy balance in terms of the following error quantities

$$\begin{cases} \Delta w_{i-1} \triangleq (w^o - w_{i-1}) \text{ weight-error array} \\ \boldsymbol{e}_a(i) = \boldsymbol{u}_i^* \Delta w_{i-1} \ \textit{a priori} \text{ estimation error} \\ \boldsymbol{e}_p(i) = \boldsymbol{u}_i^* \Delta w_i \ \textit{a posteriori} \text{ estimation error} \end{cases} \tag{5.22}$$

together with the AF's recursion.

This way, since adaptive filters are non-linear, time-varying, and stochastic, it is necessary to adopt a set of assumptions (stationary data model) ([1, p.231]),

Definition 35 (Stationary Data Model)**.**

(1) There exists an array of multivectors w^o such that $\boldsymbol{d}(i) = \boldsymbol{u}_i^* w^o + \boldsymbol{v}(i)$;

(2) The noise sequence $\{\boldsymbol{v}(i)\}$ is independent and identically distributed (i.i.d.) with constant variance $\mathrm{E}\widetilde{\boldsymbol{v}}(i)\boldsymbol{v}(i) = \mathrm{E}\,\|\boldsymbol{v}(i)\|^2$;

(3) The noise sequence $\{\boldsymbol{v}(i)\}$ is independent of $\boldsymbol{u}_j$ for all $i, j,$ and all other data;

(4) The initial condition w_{-1} is independent of all $\{\boldsymbol{d}(i), \boldsymbol{u}_i, \boldsymbol{v}(i)\}$;

(5) The expectation of $\boldsymbol{u}_i^* \boldsymbol{u}_i$ is denoted by $\mathrm{E}\boldsymbol{u}_i^* \boldsymbol{u}_i > 0$;

(6) The random quantities $\{\boldsymbol{d}(i), \boldsymbol{u}_i, \boldsymbol{v}(i)\}$ are zero mean.

□

Similarly to the definition of d, a lower case letter was adopted to represent the general multivector v (another exception to the convention used in this text).

The steady-state *excess mean-square error* (EMSE) is defined from the *a priori* estimation error $\boldsymbol{e}_a(i)$,

Definition 36 (Steady-State EMSE)**.**

$$\mathrm{EMSE} = \zeta \triangleq \lim_{i\to\infty} \mathrm{E}\Big\langle \|\boldsymbol{e}_a(i)\|^2 \Big\rangle = \lim_{i\to\infty} \Big\langle \mathrm{E}\,\|\boldsymbol{e}_a(i)\|^2 \Big\rangle. \tag{5.23}$$

Similar to (5.21), it involves the calculation of the scalar part (0-grade) of the multivector $\|\boldsymbol{e}_a(i)\|^2 = \widetilde{\boldsymbol{e}}_a(i)\boldsymbol{e}_a(i)$. □

As will be seen ahead, the analysis procedure requires the expectation of $\widetilde{\boldsymbol{v}}(i)\boldsymbol{v}(i)$ to be calculated.

Definition 37 (Expectation of $\widetilde{\boldsymbol{v}}\boldsymbol{v}$)**.** Given a random multivector $\boldsymbol{v} \in \mathcal{G}(\mathbb{R}^3)$ (see (4.2)),

$$\begin{aligned} \boldsymbol{v} &= \langle \boldsymbol{v} \rangle_0 + \langle \boldsymbol{v} \rangle_1 + \langle \boldsymbol{v} \rangle_2 + \langle \boldsymbol{v} \rangle_3 \\ &= \boldsymbol{v}_0 + \boldsymbol{v}_1\gamma_1 + \boldsymbol{v}_2\gamma_2 + \boldsymbol{v}_3\gamma_3 + \boldsymbol{v}_4\gamma_{12} + \boldsymbol{v}_5\gamma_{23} + \boldsymbol{v}_6\gamma_{31} + \boldsymbol{v}_7 I, \end{aligned} \tag{5.24}$$

where each coefficient $\boldsymbol{v}_k$, with $k = 0, \cdots, 7$ is an i.i.d. random variable.

The geometric product $\widetilde{\boldsymbol{v}}\boldsymbol{v}$ is

$$\begin{aligned}\widetilde{\boldsymbol{v}}\boldsymbol{v} = \quad & \boldsymbol{v}_0^2 + \boldsymbol{v}_0\boldsymbol{v}_1\gamma_1 + \boldsymbol{v}_0\boldsymbol{v}_2\gamma_2 + \boldsymbol{v}_0\boldsymbol{v}_3\gamma_3 - \boldsymbol{v}_0\boldsymbol{v}_4\gamma_{12} - \boldsymbol{v}_0\boldsymbol{v}_5\gamma_{23} - \boldsymbol{v}_0\boldsymbol{v}_6\gamma_{31} - \boldsymbol{v}_0\boldsymbol{v}_7 I + \\ & \boldsymbol{v}_1^2 + \boldsymbol{v}_0\boldsymbol{v}_1\gamma_1 - \boldsymbol{v}_1\boldsymbol{v}_4\gamma_2 + \boldsymbol{v}_1\boldsymbol{v}_6\gamma_3 + \boldsymbol{v}_1\boldsymbol{v}_2\gamma_{12} - \boldsymbol{v}_1\boldsymbol{v}_7\gamma_{23} - \boldsymbol{v}_1\boldsymbol{v}_3\gamma_{31} - \boldsymbol{v}_1\boldsymbol{v}_5 I + \\ & \qquad\qquad \vdots \\ & \boldsymbol{v}_7^2 + \boldsymbol{v}_7\boldsymbol{v}_5\gamma_1 + \boldsymbol{v}_7\boldsymbol{v}_6\gamma_2 + \boldsymbol{v}_7\boldsymbol{v}_4\gamma_3 + \boldsymbol{v}_7\boldsymbol{v}_3\gamma_{12} + \boldsymbol{v}_7\boldsymbol{v}_1\gamma_{23} - \boldsymbol{v}_7\boldsymbol{v}_2\gamma_{31} + \boldsymbol{v}_7\boldsymbol{v}_0 I.\end{aligned} \tag{5.25}$$

Thus, applying the expectation operator to (5.25) results in

$$\mathrm{E}\widetilde{\boldsymbol{v}}\boldsymbol{v} = \mathrm{E}\boldsymbol{v}_0^2 + \mathrm{E}\boldsymbol{v}_1^2 + \mathrm{E}\boldsymbol{v}_2^2 + \mathrm{E}\boldsymbol{v}_3^2 + \mathrm{E}\boldsymbol{v}_4^2 + \mathrm{E}\boldsymbol{v}_5^2 + \mathrm{E}\boldsymbol{v}_6^2 + \mathrm{E}\boldsymbol{v}_7^2, \tag{5.26}$$

since the expectations of the cross-terms are zero. Each term $\mathrm{E}\boldsymbol{v}_k^2$, $k = 0, \cdots, 7$, is said to be the variance of $\boldsymbol{v}_k$ and denoted $\mathrm{E}\boldsymbol{v}_k^2 \triangleq \sigma_v^2$. This way, (5.26) becomes

$$\boxed{\mathrm{E}\widetilde{\boldsymbol{v}}\boldsymbol{v} = 8\sigma_v^2}. \tag{5.27}$$

□

Remark 9. Note that in general $\mathrm{E}\widetilde{\boldsymbol{v}}\boldsymbol{v} = \dim\{\mathcal{G}^g(\mathbb{R}^n)\}\sigma_v^2$ for $\boldsymbol{v} \in \mathcal{G}^g(\mathbb{R}^n)$, in which $\mathcal{G}^g(\mathbb{R}^n)$ can be *any subspace* of $\mathcal{G}(\mathbb{R}^n)$ (see Definition 9). When the complete geometric algebra is used, i.e., $\boldsymbol{v} \in \mathcal{G}(\mathbb{R}^n)$, $\dim\{\mathcal{G}(\mathbb{R}^n)\}$ is given by (3.10), and thus $\mathrm{E}\widetilde{\boldsymbol{v}}\boldsymbol{v} = 2^n\sigma_v^2$. The expectation (5.27) is the particular case when $n = 3$, i.e., when the complete geometric algebra of $\mathbb{R}^3$, namely $\mathcal{G}(\mathbb{R}^3)$, is adopted (Section 3.3.1).

□

Similarly, the expectation of $\boldsymbol{u}_i^*\boldsymbol{u}_i$ is necessary during the analysis.

Definition 38 (Expectation of $\boldsymbol{u}^*\boldsymbol{u}$)**.** Using (4.8),

$$\mathrm{E}\boldsymbol{u}^*\boldsymbol{u} = \mathrm{E}\widetilde{\boldsymbol{U}}_1\boldsymbol{U}_1 + \mathrm{E}\widetilde{\boldsymbol{U}}_2\boldsymbol{U}_2 + \cdots + \mathrm{E}\widetilde{\boldsymbol{U}}_M\boldsymbol{U}_M, \tag{5.28}$$

where $\boldsymbol{U}_j$, $j = 1, \cdots, M$ is a general multivector.

The terms $\widetilde{\boldsymbol{U}}_j\boldsymbol{U}_j$ are geometric products. For the case $\boldsymbol{U}_j \in \mathcal{G}(\mathbb{R}^3)$,

$$\begin{aligned}\widetilde{\boldsymbol{U}}_j\boldsymbol{U}_j = \;& \boldsymbol{u}_{j0}^2 + \boldsymbol{u}_{j0}\boldsymbol{u}_{j1}\gamma_1 + \boldsymbol{u}_{j0}\boldsymbol{u}_{j2}\gamma_2 + \cdots + \boldsymbol{u}_{j0}\boldsymbol{u}_{j6}\gamma_{31} - \boldsymbol{u}_{j0}\boldsymbol{u}_{j7}I + \\ & \boldsymbol{u}_{j1}^2 + \boldsymbol{u}_{j0}\boldsymbol{u}_{j1}\gamma_1 - \boldsymbol{u}_{j1}\boldsymbol{u}_{j4}\gamma_2 + \cdots + \boldsymbol{u}_{j1}\boldsymbol{u}_{j3}\gamma_{31} - \boldsymbol{u}_{j1}\boldsymbol{u}_{j5}I + \\ & \vdots \\ & \boldsymbol{u}_{j7}^2 + \boldsymbol{u}_{j7}\boldsymbol{u}_{j5}\gamma_1 + \boldsymbol{u}_{j7}\boldsymbol{u}_{j6}\gamma_2 + \cdots + \boldsymbol{u}_{j7}\boldsymbol{u}_{j2}\gamma_{31} + \boldsymbol{u}_{j7}\boldsymbol{u}_{j0}I,\end{aligned} \tag{5.29}$$

where each coefficient $\boldsymbol{u}_{jk}$, with $k = 0, \cdots, 7$ is drawn from a white Gaussian noise process. Thus,

$$\mathrm{E}\widetilde{\boldsymbol{U}}_j\boldsymbol{U}_j = \mathrm{E}\,\boldsymbol{u}_{j0}^2 + \mathrm{E}\,\boldsymbol{u}_{j1}^2 + \mathrm{E}\,\boldsymbol{u}_{j2}^2 + \mathrm{E}\,\boldsymbol{u}_{j3}^2 + \mathrm{E}\,\boldsymbol{u}_{j4}^2 + \mathrm{E}\,\boldsymbol{u}_{j5}^2 + \mathrm{E}\,\boldsymbol{u}_{j6}^2 + \mathrm{E}\,\boldsymbol{u}_{j7}^2, \tag{5.30}$$

since the expectations of the cross-terms are zero. Each term $\mathrm{E}\boldsymbol{u}_{jk}^2$, with $j = 1, \cdots, M$ and $k = 0, \cdots, 7$, is said to be the variance of $\boldsymbol{u}_{jk}$ and denoted $\mathrm{E}\boldsymbol{u}_{jk}^2 \triangleq \sigma_u^2$. Note that this result is also obtained if it is assumed $\boldsymbol{U}_j$, $j = 1, \cdots, M$, is a circular Gaussian random multivector. This assumption [2] considers that the grades of a random multivector are independent Gaussian random variables.

This way, (5.30) becomes

$$\mathrm{E}\widetilde{\boldsymbol{U}}_j\boldsymbol{U}_j = 8\sigma_u^2, \text{ for } j = 1, \cdots, M \tag{5.31}$$

which substituted in (5.28) yields

$$\boxed{\mathrm{E}\boldsymbol{u}^*\boldsymbol{u} = M(8\sigma_u^2)}, \tag{5.32}$$

where M is the number of taps in the filter, as pointed out in Definition 30). □

Remark 10. Note that in general $\mathrm{E}\boldsymbol{u}^*\boldsymbol{u} = M(\dim\{\mathcal{G}^g(\mathbb{R}^n)\}\sigma_u^2)$ for $\boldsymbol{u}$ with entries belonging to $\mathcal{G}^g(\mathbb{R}^n)$, in which $\mathcal{G}^g(\mathbb{R}^n)$ can be *any subspace* of $\mathcal{G}(\mathbb{R}^n)$ (see Definition 9). When the complete geometric algebra is used, $\dim\{\mathcal{G}(\mathbb{R}^n)\}$ is given by (3.10), and thus $\mathrm{E}\boldsymbol{u}^*\boldsymbol{u} = M(2^n\sigma_u^2)$. The expectation (5.32) is the particular

[2] Here the circularity condition – explained in [1, p. 8] for complex-valued random variables – is extended to encompass random multivectors.

case when $n = 3$, i.e., when the complete geometric algebra of $\mathbb{R}^3$, namely $\mathcal{G}(\mathbb{R}^3)$, is adopted (Section 3.3.1). □

From the stationary linear data model (Definition 35),

$$\begin{aligned} \boldsymbol{e}(i) &= \boldsymbol{d}(i) - \boldsymbol{u}_i^* w_{i-1} \\ &= \boldsymbol{u}_i^*(w^o - w_{i-1}) + \boldsymbol{v}(i) \\ &= \boldsymbol{e}_a(i) + \boldsymbol{v}(i). \end{aligned} \tag{5.33}$$

Thus,

$$\mathrm{E}\widetilde{\boldsymbol{e}}(i)\boldsymbol{e}(i) = \mathrm{E}\widetilde{\boldsymbol{e}}_a(i)\boldsymbol{e}_a(i) + \mathrm{E}\widetilde{\boldsymbol{v}}(i)\boldsymbol{v}(i), \tag{5.34}$$

and using (5.21) and (5.23),

$$\mathrm{MSE} = \mathrm{EMSE} + \mathrm{E}\widetilde{\boldsymbol{v}}(i)\boldsymbol{v}(i). \tag{5.35}$$

The resulting energy equation leads to a variance relation from which the MSE and the EMSE can be derived.

5.3 Steady-State Analysis

In this section, the ECR technique [1] is applied step-by-step in order to obtain an expression for the EMSE of any GAAF whose update rule has the following general shape

$$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} + \mu \boldsymbol{u}_i f(\boldsymbol{e}(i)), \tag{5.36}$$

where $f(\cdot)$ is a multivector-valued function of the estimation error $\boldsymbol{e}(i)$. Depending on the type of the GAAF (LMS, NLMS etc), $f(\cdot)$ assumes a specific value.

The ECR technique performs an interplay between the energies of the weight array $\boldsymbol{w}$ and the error $\boldsymbol{e}$ at two successive time instants, say $i-1$ and i. Quantities related to time instant $i-1$ are labeled *a priori*, and those related to i are named *a posteriori*. Equating *a priori* and *a posteriori* quantities allows for studying

the mean-square performance of adaptive filters – transient, steady-state, and tracking (this work focuses on steady-state only). As a result, an expression for the *variance relation* is obtained, which is then particularized for each AF of interest. For details on the ECR procedure, please refer to [1, p.228].

Subtracting (5.36) from the optimal weight vector w^o yields

$$\Delta \boldsymbol{w}_i = \Delta \boldsymbol{w}_{i-1} - \mu \boldsymbol{u}_i f(\boldsymbol{e}(i)), \tag{5.37}$$

in which $\Delta \boldsymbol{w}_i = w^o - \boldsymbol{w}_i$. Multiplying from the left by $\boldsymbol{u}_i^*$ (array product),

$$\begin{aligned} \boldsymbol{u}_i^* \Delta \boldsymbol{w}_i &= \boldsymbol{u}_i^* \left[\Delta \boldsymbol{w}_{i-1} - \mu \boldsymbol{u}_i f(\boldsymbol{e}(i)) \right] \\ \boldsymbol{e}_p(i) &= \boldsymbol{e}_a(i) - \mu \left\| \boldsymbol{u}_i \right\|^2 f(\boldsymbol{e}(i)), \end{aligned} \tag{5.38}$$

where $\boldsymbol{e}_p(i) = \boldsymbol{u}_i^* \Delta \boldsymbol{w}_i$ is the *a posteriori* error, $\boldsymbol{e}_a(i) = \boldsymbol{u}_i^* \Delta \boldsymbol{w}_{i-1}$ is the *a priori* error (See (5.22)), and in the last equation $\left\| \boldsymbol{u}_i \right\|^2 = \boldsymbol{u}_i^* \boldsymbol{u}_i$ (See (4.9)).

Assuming that the multivector $\left\| \boldsymbol{u}_i \right\|^2$ is a *versor* composed by the product of invertible vectors (see Definition 21), and $\boldsymbol{u}_i \neq 0$, then it has a multiplicative inverse $\boldsymbol{\Gamma}(i) \triangleq \left(\left\| \boldsymbol{u}_i \right\|^2 \right)^{-1}$. This allows for solving (5.38) for $f(\boldsymbol{e}(i))$

$$f(\boldsymbol{e}(i)) = \frac{1}{\mu} \boldsymbol{\Gamma}(i) \left[\boldsymbol{e}_a(i) - \boldsymbol{e}_p(i) \right], \tag{5.39}$$

and plugging back into (5.37) results in

$$\Delta \boldsymbol{w}_i = \Delta \boldsymbol{w}_{i-1} - \boldsymbol{u}_i \boldsymbol{\Gamma}(i) \left[\boldsymbol{e}_a(i) - \boldsymbol{e}_p(i) \right], \tag{5.40}$$

which can be rearranged as

$$\Delta \boldsymbol{w}_i + \boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i) = \Delta \boldsymbol{w}_{i-1} + \boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_p(i). \tag{5.41}$$

Taking the squared magnitude of both sides,

$$\underbrace{\left| \Delta \boldsymbol{w}_i + \boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i) \right|^2}_{\text{LHS}} = \underbrace{\left| \Delta \boldsymbol{w}_{i-1} + \boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_p(i) \right|^2}_{\text{RHS}}. \tag{5.42}$$

The left-hand side (LHS) is expanded as

$$\text{LHS} = \left(\Delta \boldsymbol{w}_i + \boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)\right) * \widetilde{\left(\Delta \boldsymbol{w}_i + \boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)\right)} \tag{5.43}$$

in which $*$ is the GA scalar product and $\widetilde{\ }$ is the reverse. Further expansion gives

$$\begin{aligned} \text{LHS} \;\; &= |\Delta \boldsymbol{w}_i|^2 + |\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)|^2 \\ &\quad + \underbrace{\Delta \boldsymbol{w}_i * \big(\widetilde{\boldsymbol{e}}_a(i) \boldsymbol{\Gamma}(i) \boldsymbol{u}_i^*\big) + \big(\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)\big) * \widetilde{\Delta \boldsymbol{w}}_i}_{\text{Sum of 3rd and 4th terms}}, \end{aligned} \tag{5.44}$$

in which $\boldsymbol{\Gamma}(i) = \widetilde{\boldsymbol{\Gamma}}(i)$ since $\widetilde{\|\boldsymbol{u}_i\|^2} = \|\boldsymbol{u}_i\|^2$ holds (See Remark 8). Applying the definition of GA scalar product and observing that the third and fourth terms of (5.44) are each other's reverse[3], their sum can be written as

$$2\left\langle \Delta \boldsymbol{w}_i \widetilde{\boldsymbol{e}}_a(i) \boldsymbol{\Gamma}(i) \boldsymbol{u}_i^* \right\rangle \Rightarrow 2\left\langle \widetilde{\boldsymbol{e}}_a(i) \boldsymbol{\Gamma}(i) \boldsymbol{u}_i^* \Delta \boldsymbol{w}_i \right\rangle, \tag{5.45}$$

where the cyclic reordering property (3.27) for the 0-grade operator was used. Note that the term $\boldsymbol{u}_i^* \Delta \boldsymbol{w}_i$ is the definition of the *a posteriori* error $\boldsymbol{e}_p(i)$ (refer to (5.38)). This way, (5.43) assumes the form

$$|\Delta \boldsymbol{w}_i|^2 + 2\left\langle \widetilde{\boldsymbol{e}}_a(i) \boldsymbol{\Gamma}(i) \boldsymbol{e}_p(i) \right\rangle + |\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)|^2 . \tag{5.46}$$

Similar procedures allow to expand the right-hand side (RHS) of (5.42) as

$$|\Delta \boldsymbol{w}_{i-1}|^2 + 2\left\langle \widetilde{\boldsymbol{e}}_p(i) \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i) \right\rangle + |\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_p(i)|^2 . \tag{5.47}$$

Substituting (5.46) (LHS) and (5.47) (RHS) into (5.42) yields the following energy relation

$$|\Delta \boldsymbol{w}_i|^2 + |\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)|^2 = |\Delta \boldsymbol{w}_{i-1}|^2 + |\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_p(i)|^2 . \tag{5.48}$$

which balances out *a priori* and *a posteriori* terms. Note that the terms enclosed

[3]Given mutually reverse multivectors A and $\widetilde{A}$, the following relation holds $\langle A \rangle = \langle \widetilde{A} \rangle$ (See Remark 5). Thus, $\langle A \rangle + \langle \widetilde{A} \rangle = 2\langle A \rangle$.

by the 0-grade operator (in (5.46) and (5.47)) are each others reverse. Thus, since their 0-grade are exactly the same (see Remark 5), they are mutually cancelled.

Taking the expectation of the terms of (5.48) with respect to the random quantities $\boldsymbol{d}(i)$ and $\boldsymbol{u}_i$ results in

$$\mathrm{E}|\Delta \boldsymbol{w}_i|^2 + \mathrm{E}\,|\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)|^2 = \mathrm{E}|\Delta \boldsymbol{w}_{i-1}|^2 + \mathrm{E}\,|\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_p(i)|^2. \tag{5.49}$$

Calculating the limit of (5.49) when $i \to \infty$ gives

$$\mathrm{E}\,|\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)|^2 = \mathrm{E}\,|\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_p(i)|^2, i \to \infty, \tag{5.50}$$

in which the steady-state condition $\mathrm{E}|\Delta \boldsymbol{w}_i|^2 = \mathrm{E}|\Delta \boldsymbol{w}_{i-1}|^2 = \textit{constant}$ as $i \to \infty$ was employed [1, p.237].

Plugging (5.38) into (5.50) results in

$$\mathrm{E}\,|\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)|^2 = \mathrm{E}\,|\boldsymbol{u}_i \boldsymbol{\Gamma}(i) (\boldsymbol{e}_a(i) - \mu \left\|\boldsymbol{u}_i\right\|^2 f)|^2, i \to \infty. \tag{5.51}$$

The right-hand side of (5.51) is expanded as

$$\mathrm{E}\Big|\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)\Big|^2 - 2\mu \mathrm{E}\Big\langle \boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i) \widetilde{f} \boldsymbol{u}_i^* \Big\rangle + \mu^2 \mathrm{E}\Big|\boldsymbol{u}_i f\Big|^2. \tag{5.52}$$

Plugging (5.52) back into (5.51) and cancelling out the term $\mathrm{E}\Big|\boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i)\Big|^2$ on both sides results in

$$2\mu \mathrm{E}\Big\langle \boldsymbol{u}_i \boldsymbol{\Gamma}(i) \boldsymbol{e}_a(i) \widetilde{f} \boldsymbol{u}_i^* \Big\rangle = \mu^2 \mathrm{E}\Big|\boldsymbol{u}_i f\Big|^2. \tag{5.53}$$

Using the cyclic reordering property on the left-hand side of (5.53) to make $\boldsymbol{u}_i^* \boldsymbol{u}_i \boldsymbol{\Gamma}(i) = 1$, the so-called *variance relation* is obtained

$$\boxed{2\mathrm{E}\Big\langle \boldsymbol{e}_a(i) \widetilde{f} \Big\rangle = \mu \mathrm{E}\Big|\boldsymbol{u}_i f\Big|^2}. \tag{5.54}$$

5.3.1 GA-LMS

For the GA-LMS, function f is given by $f(\boldsymbol{e}(i)) = \boldsymbol{e}(i) = \boldsymbol{e}_a(i) + \boldsymbol{v}(i)$ (see (5.33)). Substituting into (5.54)

$$\underbrace{2\mathrm{E}\Big\langle \boldsymbol{e}_a(i)\Big(\widetilde{\boldsymbol{e}}_a(i) + \widetilde{\boldsymbol{v}}(i)\Big)\Big\rangle}_{\text{LHS (5.55)}} = \underbrace{\mu\mathrm{E}\Big|\boldsymbol{u}_i\Big(\boldsymbol{e}_a(i) + \boldsymbol{v}(i)\Big)\Big|^2}_{\text{RHS (5.55)}}. \tag{5.55}$$

The left-hand side of (5.55) becomes

$$\begin{aligned} \text{LHS (5.55)} &= 2\mathrm{E}\big\langle \boldsymbol{e}_a(i)\widetilde{\boldsymbol{e}}_a(i)\big\rangle + 2\mathrm{E}\big\langle \boldsymbol{e}_a(i)\widetilde{\boldsymbol{v}}(i)\big\rangle \\ &= 2\mathrm{E}|\boldsymbol{e}_a(i)|^2 + 2\left(\mathrm{E}\boldsymbol{e}_a(i) * \mathrm{E}\widetilde{\boldsymbol{v}}(i)\right) \\ &= 2\mathrm{E}|\boldsymbol{e}_a(i)|^2, \end{aligned} \tag{5.56}$$

where the fact that $\boldsymbol{v}(i)$ is independent of any other random quantity was used. Additionally, it is assumed here that the entries of $\boldsymbol{v}(i)$ (and $\widetilde{\boldsymbol{v}}(i)$) are drawn from a zero-mean white Gaussian process, thus $\mathrm{E}\widetilde{\boldsymbol{v}}(i) = \mathrm{E}\boldsymbol{v}(i) = 0$.

The right-hand side of (5.55) is expanded as

$$\begin{aligned} \text{RHS (5.55)} &= \mu\mathrm{E}\left[\boldsymbol{u}_i(\boldsymbol{e}_a(i) + \boldsymbol{v}(i)) * (\widetilde{\boldsymbol{e}}_a(i) + \widetilde{\boldsymbol{v}}(i))\boldsymbol{u}_i^*\right] \\ &= \therefore = \mu\mathrm{E}\big\langle \|\boldsymbol{u}_i\|^2 \|\boldsymbol{e}_a(i)\|^2 \big\rangle + 2\mu\mathrm{E}\big\langle \|\boldsymbol{u}_i\|^2 \boldsymbol{e}_a(i)\widetilde{\boldsymbol{v}}(i)\big\rangle \\ &\qquad + \mu\mathrm{E}\big\langle \|\boldsymbol{u}_i\|^2 \|\boldsymbol{v}(i)\|^2 \big\rangle. \end{aligned} \tag{5.57}$$

Since $\boldsymbol{v}(i)$ is statistically independent from other quantities and $\mathrm{E}\widetilde{\boldsymbol{v}}(i) = \mathrm{E}\boldsymbol{v}(i) = 0$ (Definition 35), the term $2\mathrm{E}\langle \|\boldsymbol{u}_i\|^2 \boldsymbol{e}_a(i)\widetilde{\boldsymbol{v}}(i)\rangle = 2\mathrm{E}\big(\|\boldsymbol{u}_i\|^2 \boldsymbol{e}_a(i)\big) * \mathrm{E}\widetilde{\boldsymbol{v}}(i) = 0$. This way,

$$\text{RHS (5.55)} = \mu\mathrm{E}\big\langle \|\boldsymbol{u}_i\|^2 \|\boldsymbol{e}_a(i)\|^2 \big\rangle + \mu\mathrm{E}\big\langle \|\boldsymbol{u}_i\|^2 \|\boldsymbol{v}(i)\|^2 \big\rangle. \tag{5.58}$$

Substituting (5.56) and (5.58) into (5.55) gives

$$2\mathrm{E}|\boldsymbol{e}_a(i)|^2 = \mu\mathrm{E}\big\langle \|\boldsymbol{u}_i\|^2 \|\boldsymbol{e}_a(i)\|^2 \big\rangle + \mu\mathrm{E}\big\langle \|\boldsymbol{u}_i\|^2 \|\boldsymbol{v}(i)\|^2 \big\rangle. \tag{5.59}$$

Observing that $2\mathrm{E}|\boldsymbol{e}_a(i)|^2 = 2\mathrm{E}\big\langle \|\boldsymbol{e}_a(i)\|^2 \big\rangle$, (5.59) can be rewritten as

$$\mathrm{E}\big\langle (2-\mu\|\boldsymbol{u}_i\|^2)\,\|\boldsymbol{e}_a(i)\|^2 \big\rangle = \mu\mathrm{E}\big\langle \|\boldsymbol{u}_i\|^2\,\|\boldsymbol{v}(i)\|^2 \big\rangle. \tag{5.60}$$

Adopting the *separation principle* (see [1, p.245]), i.e., in steady state $\|\boldsymbol{u}_i\|^2$ is independent of $\boldsymbol{e}(i)$ (and consequently of $\boldsymbol{e}_a(i)$), (5.60) becomes

$$\big\langle (2-\mu\mathrm{E}\,\|\boldsymbol{u}_i\|^2)\mathrm{E}\,\|\boldsymbol{e}_a(i)\|^2 \big\rangle = \mu\big\langle \mathrm{E}\,\|\boldsymbol{u}_i\|^2\,\mathrm{E}\,\|\boldsymbol{v}(i)\|^2 \big\rangle. \tag{5.61}$$

In the most general case, i.e., when all the multivectors belong to the complete algebra $\mathcal{G}(\mathbb{R}^n)$, the terms $\mathrm{E}\,\|\boldsymbol{v}(i)\|^2$ and $\mathrm{E}\,\|\boldsymbol{u}_i\|^2$ are calculated as described in Remarks 9 and 10, namely, $\mathrm{E}\,\|\boldsymbol{v}(i)\|^2 = 2^n\sigma_v^2$ and $\mathrm{E}\,\|\boldsymbol{u}_i\|^2 = M(2^n\sigma_u^2)$. Substituting into (5.61) yields

$$(2-\mu M(2^n\sigma_u^2))\big\langle \mathrm{E}\,\|\boldsymbol{e}_a(i)\|^2 \big\rangle = \mu M(2^n\sigma_u^2)(2^n\sigma_v^2). \tag{5.62}$$

It is important to notice that since Remark 10 is obtained considering inputs (regressor entries) drawn from a circular Gaussian process (see [1, p. 8]), the present analysis holds only for that kind of input.

Finally, the expression for the GA-LMS steady-state EMSE using the complete algebra $\mathcal{G}(\mathbb{R}^n)$ is given by

$$\boxed{\underset{\text{LMS}}{\zeta} \;=\; \frac{\mu M 4^n \sigma_u^2 \sigma_v^2}{2-\mu M 2^n \sigma_u^2},\;\; i\to\infty}. \tag{5.63}$$

For the special case $n=3$, (5.63) becomes

$$\boxed{\underset{\text{LMS }(n\,=\,3)}{\zeta} \;=\; \frac{32\mu M \sigma_u^2 \sigma_v^2}{1-4\mu M\sigma_u^2},\;\; i\to\infty}. \tag{5.64}$$

Table 2 summarizes the EMSE theoretical values for several algebras. Those are useful in Section 7.2, where GAAFs with entries in $\mathcal{G}(\mathbb{R}^3), \mathcal{G}^+(\mathbb{R}^3), \mathcal{G}^+(\mathbb{R}^2)$, and $\mathcal{G}^+(\mathbb{R})$ have their steady-state performance assessed. Notice that for $\mathcal{G}^+(\mathbb{R})$

Table 2: Steady-state EMSE of GA-LMS for several algebras and subspaces of interest.

Complete GA of $\mathbb{R}^n$ $\mathcal{G}(\mathbb{R}^n)$	$\dfrac{\mu M 4^n \sigma_u^2 \sigma_v^2}{2 - \mu M 2^n \sigma_u^2}$
Any subspace g $\mathcal{G}^g(\mathbb{R}^n)$	$\dfrac{\mu M \binom{n}{g}^2 \sigma_u^2 \sigma_v^2}{2 - \mu M \binom{n}{g} \sigma_u^2}$
Even Algebras $\mathcal{G}^+(\mathbb{R}^n)$	$\dfrac{\mu M \left[\sum\limits_k \binom{n}{2k}\right]^2 \sigma_u^2 \sigma_v^2}{2 - \mu M \sigma_u^2 \sum\limits_k \binom{n}{2k}}$, for $k = 0, 1, 2, 3, \cdots$
Complete GA of $\mathbb{R}^3$ $\mathcal{G}(\mathbb{R}^3)$	$\dfrac{32 \mu M \sigma_u^2 \sigma_v^2}{1 - 4 \mu M \sigma_u^2}$
Rotor GA of $\mathbb{R}^3$ (Quaternions) $\mathcal{G}^+(\mathbb{R}^3)$	$\dfrac{\mu M \left[\binom{3}{0} + \binom{3}{2}\right]^2 \sigma_u^2 \sigma_v^2}{2 - \mu M \sigma_u^2 \left[\binom{3}{0} + \binom{3}{2}\right]}$
Rotor GA of $\mathbb{R}^2$ (Complex) $\mathcal{G}^+(\mathbb{R}^2)$	$\dfrac{\mu M \left[\binom{2}{0} + \binom{2}{2}\right]^2 \sigma_u^2 \sigma_v^2}{2 - \mu M \sigma_u^2 \left[\binom{2}{0} + \binom{2}{2}\right]}$
Rotor GA of $\mathbb{R}$ (Real) $\mathcal{G}^+(\mathbb{R})$	$\dfrac{\mu M \sigma_u^2 \sigma_v^2}{2 - \mu M \sigma_u^2}$

the EMSE for the LMS with real-valued entries is recovered (compare to Equation 16.10 in [1, p.246] for white Gaussian inputs).

To obtain the respective MSE, one should add $\mathrm{E}\,\|\boldsymbol{v}(i)\|^2$ (Definition 37 and Remark 9) to the EMSE expression, as pointed out in (5.35).

6 GEOMETRIC-ALGEBRA ADAPTIVE FILTERS (POSE ESTIMATION)

In this chapter, a GA-based AF capable of estimating rotations in a 3-dimensional space is devised. To evaluate its performance, the 3D registration of point clouds (Section 2.3) is adopted.

At first, it is provided an overview on how the 3D registration problem is usually posed. Then it is recast in the GA framework.

6.1 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-to-1 correspondence, in which at least a part of X is a translated and rotated version of Y. Each PCD has K points, $\{y_k\} \in$ Y and $\{x_k\} \in$ X, $k = 1...K$, obtained via a local-shape feature-based matching system [10]. In the registration process, the goal is to find a rigid transformation to align Y and X. In other words, what is the rotation and the translation that should be applied to each point in X to map it onto Y?

This question can be posed as a least-squares problem [37, 38], in which one should minimize the following cost function

$$\mathcal{F}(R,t) = \frac{1}{K}\sum_{k=1}^{K}\left\|y_k^{'} - Rx_k^{'} - t\right\|_2^2, \tag{6.1}$$

where $y_k^{'}$ and $x_k^{'}$ are 3×1 vectors representing (presumably) *corresponding* points

in the target and source PCDs, respectively; R is a 3×3 rotation matrix; and t is a 3×1 translation vector.

Defining the centroids of the target and the source PCDs as $\bar{y}$ and $\bar{x}$ respectively, the coordinate of each point in a PCD with respect to its centroid is given by

$$\begin{aligned} y_k &= y_k^{'} - \bar{y} \\ x_k &= x_k^{'} - \bar{x}. \end{aligned} \tag{6.2}$$

Substituting (6.2) into (6.1) and setting $t = \bar{y} - R\bar{x}$ (translating the source PCD's centroid to match that of the target PCD), the cost function (6.1) can be rewritten as a constrained least-squares problem in terms of R,

$$\mathcal{F}(R) = \frac{1}{K} \sum_{k=1}^{K} \left\| y_k - R x_k \right\|^2, \text{ subject to } R^* R = R R^* = I_d, \tag{6.3}$$

in which * denotes the conjugate transpose, and I_d is the identity matrix.

In the registration process, one needs to determine the *linear operator*, i.e., the 3×3 rotation matrix R ([39], p.320), that maps X onto Y. Finding the matrix R that minimizes (6.3) is known as the *orthogonal procrustes problem.* Methods available in the literature rely on a standard least-squares estimator which is typically based on the singular value decomposition (SVD) algorithm of the PCDs cross-covariance matrix [37, 38, 40–42]. After estimating R, the translation is recovered by $t = \bar{y} - R\bar{x}$.

To estimate a (transformation) matrix, one may consider using Kronecker products and vectorization [1]. 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 [34]. 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 [33, p. 30] and [43]). Nevertheless, performing standard vector calculus in quaternion algebra (to calculate the gradient of the error vector) incur a cumbersome analytic derivation [27, 29, 30]. To circumvent that, (6.3) is recast in GA (which encompasses quaternion algebra) by using *rotors*. 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 for pose estimation 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]).

6.2 The Rotation Estimation Problem in GA

The problem (6.3) may be posed in GA as follows [44]. The rotation matrix R in the error vector $y_k - Rx_k$ is substituted by the rotation operator comprised by the bivector r (written in lower-case letter, an exception to the convention used in this text – refer to Section 2.1) and its *reversed* version $\widetilde{r}$ [4],

$$e_k = y_k - rx_k\widetilde{r}, \text{ subject to } r\widetilde{r} = \widetilde{r}r = |r|^2 = 1. \tag{6.4}$$

Note that error e_k has the same shape of the error in (4.18). However, in this case, r is a *unit rotor* in $\mathcal{G}(\mathbb{R}^3)$, which makes (6.4) a particular case of (4.18) for $n = 3$. This way, the term $rx_k\widetilde{r}$ is simply a rotated version of the vector x_k ([5, Eq.54]).

This rotation description is similar to the one provided by quaternion algebra. Recall from Section 3.3.3 that the subalgebra of $\mathcal{G}(\mathbb{R}^3)$ containing only the multivectors with even grades (rotors) is in fact *isomorphic* to quaternions. 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 [3, 35].

Hypercomplex AFs available in the literature make use of quaternion algebra [26, 28] and even GA theory [36]. However, the error vector therein has the form $e = y - rx$, which is similar to the *standard form* in (4.14) with $d = y$ and $u^* = r$. While this is the proper error function shape to derive *standard* AFs (Chapter 5), it is definitely not appropriate to model rotation error since it lacks $\widetilde{r}$ multiplying x from the right, like in (4.18) and (6.4).

This way, (6.3) is rewritten using (6.4), generating the GA-based cost function,

$$J(r) = \frac{1}{K}\sum_{k=1}^{K} |y_k - rx_k\widetilde{r}|^2 = \frac{1}{K}\sum_{k=1}^{K} |e_k|^2 = \frac{1}{K}\sum_{k=1}^{K} e_k * \widetilde{e}_k = \frac{1}{K}\sum_{k=1}^{K} \langle e_k\widetilde{e}_k\rangle, \quad (6.5)$$

subject to $r\widetilde{r} = \widetilde{r}r = |r|^2 = 1$. Note that (6.5) is the least-squares counterpart of (4.19).

6.3 Deriving the GAAFs

The GAAFs designed in the sequel should make $rx_k\widetilde{r}$ as close as possible to y_k in order to minimize (6.5). Applying the same reasoning adopted in Chapter 5 (see (5.1)), the AFs are designed to provide an estimate for the bivector r via a recursive rule of the form,

$$r_i = r_{i-1} + \mu G, \quad (6.6)$$

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 (6.4) (analogous to the standard formulation in [1], p.143).

Once more, the steepest-descent rule is adopted, designing the AF to fol-

low 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 $\widetilde{\nabla} J(r_{i-1})$,

$$G \triangleq -B\widetilde{\nabla} J(r_{i-1}), \tag{6.7}$$

in which B is a general multivector.

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

$$\begin{aligned} J(r) &= \sum_{k=1}^{K} \left(y_k - r x_k \widetilde{r}\right) * \widetilde{\left(y_k - r x_k \widetilde{r}\right)} \\ &= \sum_{k=1}^{K} \left[y_k * \widetilde{y}_k - y_k * (r\widetilde{x}\widetilde{r}) - (r x \widetilde{r}) * \widetilde{y}_k + (r x \widetilde{r}) * \widetilde{(r x \widetilde{r})} \right] \\ &= \sum_{k=1}^{K} |y_k|^2 + |x_k|^2 - 2\langle y_k r x_k \widetilde{r} \rangle, \end{aligned} \tag{6.8}$$

where the reversion operation (3.24) was used to conclude that $y_k = \widetilde{y}_k$, $x_k = \widetilde{x}_k$ (they are vectors), and $r\widetilde{r} = \widetilde{r}r = 1$.

Using Geometric Calculus techniques (refer to Section 3.5), the gradient of $J(r)$ is calculated from (6.8),

$$\begin{aligned} \nabla J(r) = \partial_r J(r) &= -2\partial_r \sum_{k=1}^{K} \langle y_k r x_k \widetilde{r} \rangle \\ &= -2 \left[\sum_{k=1}^{K} \partial_r \langle \dot{r} M_k \rangle + \partial_r \langle T_k \dot{\widetilde{r}} \rangle \right], \end{aligned} \tag{6.9}$$

in which the product rule (Definition 28) was used and the overdots emphasize which quantity is being differentiated by ∂_r (Definition 26). The terms $M_k = x_k \widetilde{r} y_k$ and $T_k = y_k r x_k$ are obtained applying the *cyclic reordering property* $\langle AD \cdots C \rangle = \langle D \cdots CA \rangle = \langle CAD \cdots \rangle$ (3.27). The first term on the right-hand side of (6.9) is $\partial_r \langle \dot{r} M_k \rangle = M_k$ ([35], Eq. 7.10), and the second term is $\partial_r \langle T_k \dot{\widetilde{r}} \rangle = -\widetilde{r} T_k \widetilde{r} = -\widetilde{r}(y_k r x_k)\widetilde{r}$ (see (3.39)). Plugging back into (6.9), the GA-form of the

gradient of $J(r)$ is obtained

$$\begin{aligned}\partial_r J(r) &= -2\sum_{k=1}^{K} x_k\widetilde{r}y_k - \widetilde{r}(y_k r x_k)\widetilde{r} \\ &= -2\widetilde{r}\sum_{k=1}^{K}(rx_k\widetilde{r})y_k - y_k(rx_k\widetilde{r}) = 4\widetilde{r}\sum_{k=1}^{K} y_k \wedge (rx_k\widetilde{r}),\end{aligned} \tag{6.10}$$

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

In [16], the GA framework to handle linear transformations is applied for mapping (6.10) 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 (6.10) into (5.2) (with $B = 1$) and explicitly showing the term $1/K$ results in

$$G = \frac{4}{K}\left[\sum_{k=1}^{K} y_k \wedge (r_{i-1}x_k\widetilde{r}_{i-1})\right] r_{i-1}, \tag{6.11}$$

which upon plugging into (5.1) yields

$$\boxed{r_i = r_{i-1} + \mu\frac{4}{m}\left[\sum_{k=1}^{m} y_k \wedge (r_{i-1}x_k\widetilde{r}_{i-1})\right] r_{i-1}}, \tag{6.12}$$

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, 6]. If $m = K$, (6.12) uses all the available points, originating the *geometric-algebra steepest-descent* algorithm for pose estimation. This work focuses on the case $m = 1$ (one pair per iteration) which is equivalent to approximating $\widetilde{\nabla}J(r)$ by its *current value* in (6.11) [1],

$$\frac{4}{K}\left[\sum_{k=1}^{K} y_k \wedge (r_{i-1}x_k\widetilde{r}_{i-1})\right] r_{i-1} \approx 4\left[y_i \wedge (r_{i-1}x_i\widetilde{r}_{i-1})\right] r_{i-1}. \tag{6.13}$$

In the sequel, the GA-LMS is obtained from (6.12). Given the proper calculation

of the Laplacian of (6.8), other types of filters, e.g., NLMS and RLS, can be derived as well. Those were recently developed (see Appendix A) and are still in the implementation stage.

6.3.1 GA Least-Mean Squares (GA-LMS)

Plugging (6.13) into (6.12) results in the GA-LMS update rule,

$$\boxed{r_i = r_{i-1} + \mu \left[y_i \wedge (r_{i-1} x_i \widetilde{r}_{i-1}) \right] r_{i-1}}, \tag{6.14}$$

in which the factor 4 was absorbed by μ. Note that (6.14) was obtained without restrictions to the dimension of the vector space containing $\{y_k, x_k\}$.

Adopting (6.13) 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} (see Figure 8). Thus, the GA-LMS uses *less information per iteration* when compared to methods in the literature [37, 38, 40–42, 45] 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 (6.5)) and avoid scaling the PCD points, the estimate r_i in (6.14) is normalized at each iteration when implementing the GA-LMS.

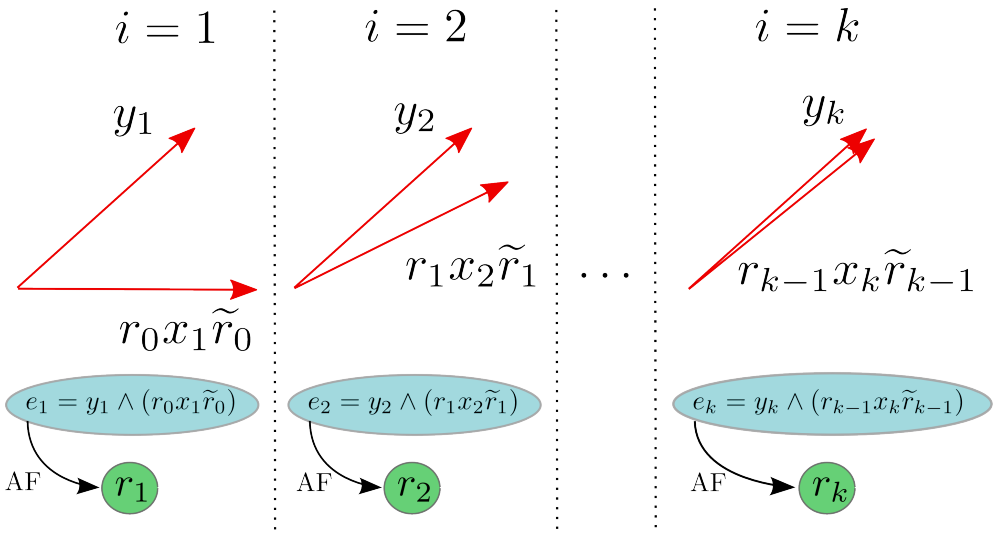


Figure 8: Step-by-step: this figure provides a visual representation of what happens at each GA-LMS (pose estimation) iteration. A different pair $\{y_k, x_k\}$ is selected at each step and the *a priori* estimate of r is applied to it. Given a proper selection of the step-size value, after a number of iterations, the vectors y_k and $r_{k-1}x_k\widetilde{r}_{k-1}$ are (almost) aligned. Any misalignment is due to the existence of outliers which arise during the feature matching stage (refer to [46] for techniques to minimize the influence of outliers). Once the AF has converged, the final estimate of r is applied to all the points in X (Source PCD), aligning it with Y (Target PCD).

6.4 Algorithm Performance

6.4.1 Computational Complexity

The computational cost is calculated by breaking (6.14) 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 (i.e., data reprocessing), typical in CV, is hindered due to real-time constraints.

6.4.2 Step-size Bounds

Selecting the step size within an appropriate range of values prevents the AF from diverging and enables us to take full advantage of the GA-LMS capabilities. This way, a simple formula is devised, refined by empirical data, for the step-size bounds as a function of the PCDs dimensions and the point correspondences.

Embedding $1/K$ into $J(r)$ in (6.5), (6.12) is plugged into (6.5),

$$\begin{aligned} J(r_i) &= \sum_{k=1}^{K} |y_k|^2 + |x_k|^2 - 2\langle y_k r_i x_k \widetilde{r_i} \rangle \\ &= c - 2 \sum_{k=1}^{K} \left\langle y_k \Big(r_{i-1} - \mu \widetilde{\nabla} J \Big) x_k \Big(r_{i-1} - \mu \widetilde{\nabla} J \Big)^{\widetilde{}} \right\rangle, \end{aligned} \tag{6.15}$$

where $c = \sum_{k=1}^{K} |y_k|^2 + |x_k|^2$. Notice, the index $i-1$ of $\widetilde{\nabla} J$ was omitted for the sake of readability and $\langle ab \rangle = a \cdot b$, $a, b \in \mathbb{R}^3$ is used (Definition 16).

The second term of the right-hand side of (6.8) can be expanded as

$$\begin{aligned} 2 \sum_{k=1}^{K} \; & \langle y_k r_{i-1} x_k \widetilde{r}_{i-1} - \mu y_k r_{i-1} x_k \nabla J \\ & - \mu y_k \widetilde{\nabla} J x_k \widetilde{r}_{i-1} + \mu^2 y_k \widetilde{\nabla} J x_k \nabla J \rangle. \end{aligned} \tag{6.16}$$

Thus, a recursive equation for the cost function value is obtained

$$\begin{aligned} J(r_i) = J(r_{i-1}) \; & + 2\mu \sum_{k=1}^{K} \langle y_k (r_{i-1} x_k \nabla J + \widetilde{\nabla} J x_k \widetilde{r}_{i-1}) \rangle \\ & - \mu^2 \sum_{k=1}^{K} \langle y_k \widetilde{\nabla} J x_k \nabla J \rangle, \end{aligned} \tag{6.17}$$

in which $J(r_{i-1}) = c - \sum_{k=1}^{K} 2\langle y_k r_{i-1} x_k \widetilde{r}_{i-1} \rangle$.

In order to make $J(r_i) < J(r_{i-1})$ at each iteration [1], one should select the step size within the interval bounded by the roots of the following second-order

equation in μ,

$$2\mu\sum_{k=1}^{K}\langle y_k(r_{i-1}x_k\nabla J+\widetilde{\nabla}Jx_k\widetilde{r}_{i-1})\rangle-\mu^2\sum_{k=1}^{K}\langle y_k\widetilde{\nabla}Jx_k\nabla J\rangle=0, \tag{6.18}$$

which is comprised by the second and third terms on the right-hand side of (6.17). Note that one of the roots of (6.18) is $\mu=0$. The second root is calculated via

$$\mu(i)=4\sum_{k=1}^{K}\Big\langle y_k r_{i-1}x_k\nabla J\Big\rangle\Big/\sum_{k=1}^{K}\Big\langle y_k\widetilde{\nabla}Jx_k\nabla J\Big\rangle, \tag{6.19}$$

where the use of the *cyclic reordering property* (3.27) and the relation $\langle A\rangle=\langle\widetilde{A}\rangle$ (which holds for any multivector A – see Remark 5) allows for rewriting the first term of (6.18) as $2\sum\limits_{k=1}^{K}\Big\langle y_k\Big[r_{i-1}x_k\nabla J+\widetilde{\nabla}Jx_k\widetilde{r}_{i-1}\Big]\Big\rangle=4\sum\limits_{k=1}^{K}\langle y_k r_{i-1}x_k\nabla J\rangle$. Note that μ is iteration dependent (it is a function of the rotor r_{i-1}).

To represent μ solely as a function of the Target (Y) and Source (X) PCDs dimensions and their point correspondences, one can simplify (6.19) by removing the dependence on r_{i-1}. The underlying idea is to consider r_{i-1} as a "dummy" rotation, i.e., make $r_{i-1}=1$ in (6.19). As a consequence, (6.10) becomes

$$\nabla J{=}4\sum_{k=1}^{K}y_k\wedge x_k. \tag{6.20}$$

The intuition is that it will provide the necessary μ value for the next iteration given that the PCDs were not rotated in the previous iteration. Applying this to (6.19) allows us to express μ as

$$\mu(Y,X)=\sum_{k=1}^{K}\langle y_k x_k Q\rangle\Big/\sum_{k=1}^{K}\langle y_k\widetilde{Q}x_k Q\rangle, \tag{6.21}$$

in which $Q=\sum\limits_{k=1}^{K}y_k\wedge x_k$. As an example, using (6.21) to compute the step-size value for the Stanford Bunny dataset [47] results in $\mu=0.51$.

As aforementioned, (6.21) is obtained to make $J(r_i)<J(r_{i-1})$ at each iteration. However, this approach is very conservative, prioritizing monotonic convergence at the expense of speed (the step-size upper limit depicted by (6.21)

is very small, making the filter adaptation very slow). Indeed, the simulations in [44] show that the GA-LMS is able to align the Stanford Bunny PCDs, without diverging, using $\mu = 8$, approximately fifteen times the conservative value $\mu = 0.51$. Using this fact, the constraint $J(r_i) < J(r_{i-1}) \ \forall i$ is relaxed and (6.21) is multiplied by $\rho \geq 1$ to obtain a simple rule for selecting μ,

$$\boxed{\mu(Y, X) = \rho \sum_{k=1}^{K} \langle y_k x_k Q \rangle \Big/ \sum_{k=1}^{K} \langle y_k \widetilde{Q} x_k Q \rangle}. \tag{6.22}$$

In this work $\rho = 15$ was adopted, an empirical value determined from experiments performed with the Stanford Bunny PCDs. Figure 9 shows the step size values calculated via (6.22) as a function of the PCDs greatest dimension. Note the inverse proportion between PCD size and μ, i.e., for big PCDs one should use small step sizes, and vice versa. This is the case because the input data $\{y, x\}$ provided by big PCDs has higher power (greater magnitudes $|y|, |x|$) than the one from small PCDs, resulting in higher values for the gradient term (6.10). Thus, to avoid divergence, one must compensate for it by making μ smaller.

Since the selection of μ is not required to be exact, one should think of Figure 9 as a chart of possible values for μ. For example, the greatest dimension in the PCDs of the Stanford Bunny dataset is $15cm$, which in Figure 9 corresponds to $\mu = 7.7$. One may select a slightly higher value to increase speed, however, this also increases the chances of divergence. Choosing a lower value is also possible, with convergence speed reduction as a side effect. Therefore, Figure 9 depicts the recommended *superior limit* for μ.

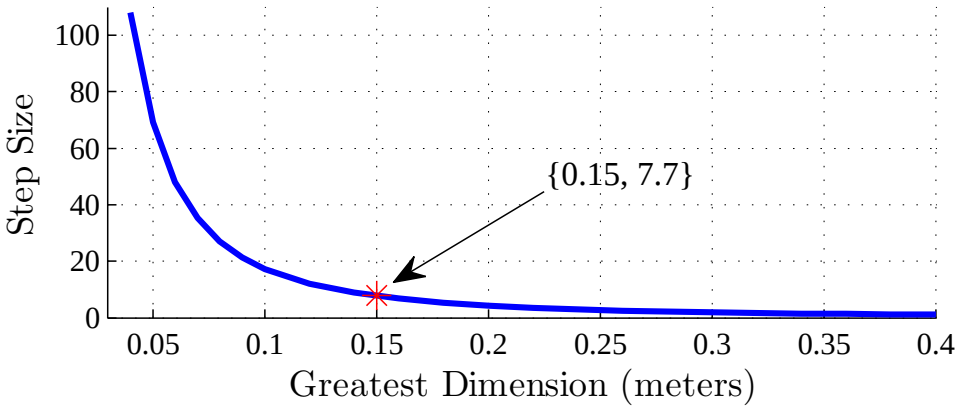


Figure 9: Simple rule for selecting μ.

7 APPLICATIONS OF GAAFS

This Chapter shows the performance of the computational implementation of GAAFs. Each type of GAAFs is tested in an specific application: standard GAAFs are studied in a system identification task while the GAAF for pose estimation is applied as the minimizer in a 3D point cloud alignment problem.

Before presenting the GAAFs learning curves, an overview is provided on how the AFs are implemented.

7.1 Implementation in C++

The computational implementation of outer and geometric products requires special libraries and/or toolboxes, depending on the computational language.

A number of GA libraries/toolboxes are freely available on the Internet. To select the most suitable for this work, two characteristics were prioritized: *speed* (to enable the use of AFs in real-time applications) and *openness* (one should be able to modify the core of the library in order to add specific features of AFs). In light of that, the *Geometric Algebra ALgorithms Expression Templates* (GAALET) [48], a C++ library for evaluation of GA expressions, was selected.

All the AFs studied in this text were implemented in C++ using GAALET. Several new functions were created and added to the source code of GAALET to enable the construction of arrays of multivectors, the multiplication between

two arrays (array product), between arrays and multivectors, and between arrays and scalars.

This work has supplementary material available at openga.org, a website created by the author to share all the source codes and scripts necessary to reproduce the experiments herein. For detailed information about the filter implementations, please refer to that website. Also, tutorials and videos are provided in order to help the reader to get started with GAAFs.

7.2 System Identification with standard GAAFs

This section employs the standard GAAFs in a system identification task (explained in Section 2.2). The *optimal weight array* w^o to be estimated has M multivector-valued entries (number of taps), namely W_j, $j = 1, \cdots, M$,

$$w^o = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_M \end{bmatrix}. \tag{7.1}$$

Each case studied in the sequel (multivector, rotor, complex, and real entries) adopts a different value for W_j (which will be highlighted in a timely manner).

As aforementioned, the measurement noise multivector $\boldsymbol{v}$ has each of its coefficients drawn from a white Gaussian stationary process with variance σ_v^2.

7.2.1 Multivector Entries

The underlying geometric algebra in this case is $\mathcal{G}(\mathbb{R}^n)$, with $n = 3$, i.e., the one whose multivectors are described by basis (3.12). The derivation of the GAAFs puts no restriction on the values the vector space dimension n can assume. However, setting $n = 3$ (generating a GA with dimension 8) provides

a didactic example that captures the core idea of this work: the GAAFs can estimate hypercomplex quantities which generalize real, complex, and quaternion entries.

Here the optimal weight array is,

$$w^o = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_M \end{bmatrix} = \begin{bmatrix} 0.55 + 0\gamma_1 + 1\gamma_2 + 2\gamma_3 + 0.71\gamma_{12} + 1.3\gamma_{23} + 4.5\gamma_{31} + 3I \\ 0.55 + 0\gamma_1 + 1\gamma_2 + 2\gamma_3 + 0.71\gamma_{12} + 1.3\gamma_{23} + 4.5\gamma_{31} + 3I \\ \vdots \\ 0.55 + 0\gamma_1 + 1\gamma_2 + 2\gamma_3 + 0.71\gamma_{12} + 1.3\gamma_{23} + 4.5\gamma_{31} + 3I \end{bmatrix}, \quad (7.2)$$

where all multivector entries are the same, namely $\{0.55 + 0\gamma_1 + 1\gamma_2 + 2\gamma_3 + 0.71\gamma_{12} + 1.3\gamma_{23} + 4.5\gamma_{31} + 3I\}$. Those values were selected in an aleatory manner. Note that the coefficient of γ_1 is zero. However, it was kept in (7.2) to emphasize the structure of the $\mathcal{G}(\mathbb{R}^3)$ basis.

Figure 10 shows several learning curves (MSE and EMSE) for the GA-LMS estimating the weight array (7.2) with $M = 10$. The step size value is $\mu = 0.005$ for all simulated curves. Notice the perfect agreement between the theoretical error levels (obtained with (5.64)) and the simulated steady-state error. Those experiments show that the GA-LMS is indeed *capable of estimating multivector-valued quantities*, supporting what was previously devised in Chapter 5.

Figure 11 depicts the steady-state error as a function of the system order (number of taps) M. Multiple curves are provided, each for a specific value of measurement noise σ_v^2. Theory and experimental values agree throughout the entire tested range $M = [1, 40]$.

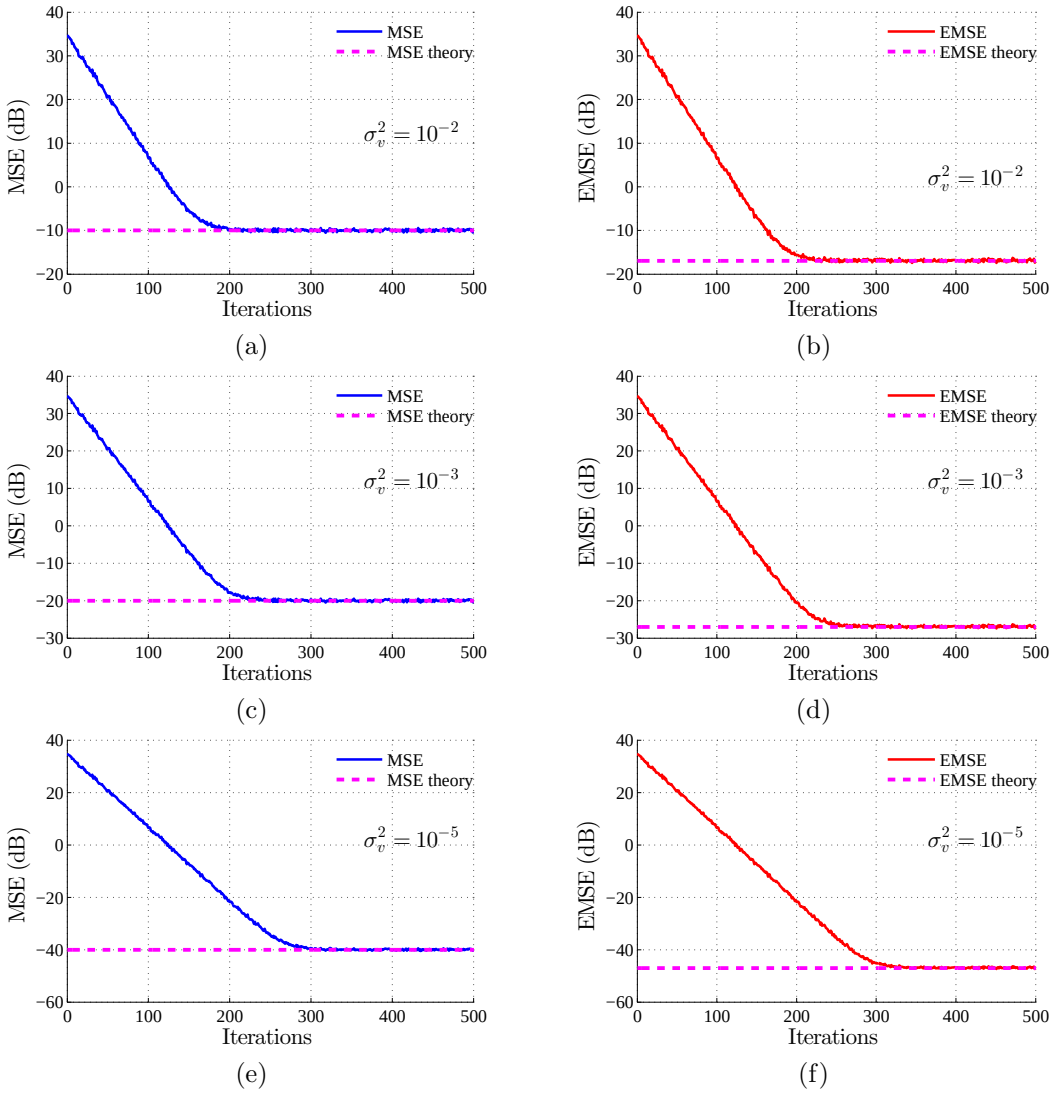
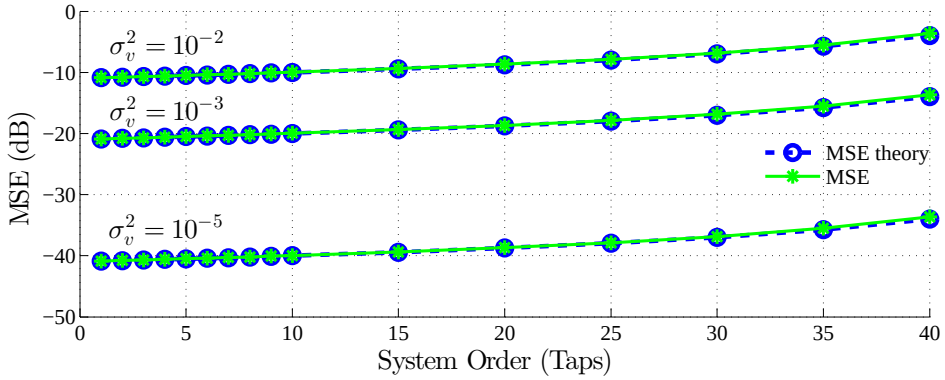
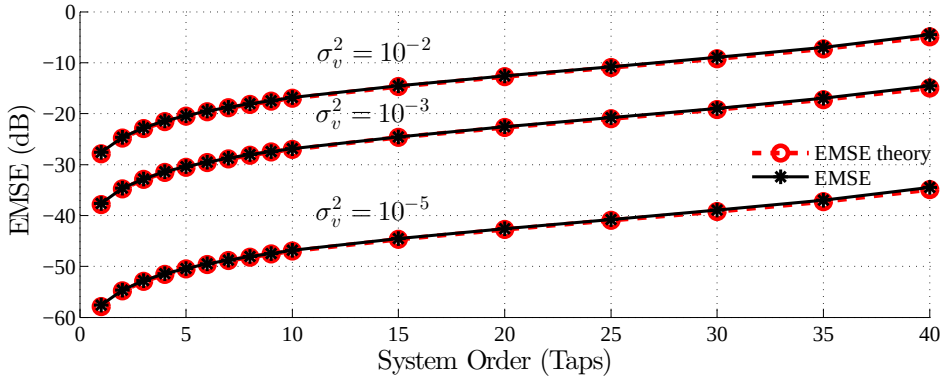


Figure 10: GA-LMS: MSE and EMSE learning curves for $M = 10$, $\mu = 0.005$, and $\sigma_v^2 = \{10^{-2}, 10^{-3}, 10^{-5}\}$. The curves are averaged over 100 experiments.



(a)



(b)

Figure 11: GA-LMS: steady-state MSE and EMSE as functions of the system order (number of taps) M for $\sigma_v^2 = \{10^{-2}, 10^{-3}, 10^{-5}\}$. The simulated steady-state value is obtained by averaging the last 200 points of the ensemble-average learning curve for each M. Notice how the simulated curves agree with the model.

7.2.2 Rotor Entries

In this case, the underlying geometric algebra is $\mathcal{G}^+(\mathbb{R}^3)$ (isomorphic to quaternions – see Section 3.3.3), and the optimal weight array is

$$w^o = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_M \end{bmatrix} = \begin{bmatrix} 0.55 + 0.71\gamma_{12} + 1.3\gamma_{23} + 4.5\gamma_{31} \\ 0.55 + 0.71\gamma_{12} + 1.3\gamma_{23} + 4.5\gamma_{31} \\ \vdots \\ 0.55 + 0.71\gamma_{12} + 1.3\gamma_{23} + 4.5\gamma_{31} \end{bmatrix}, \tag{7.3}$$

where all rotor entries are the same, namely $\{0.55 + 0.71\gamma_{12} + 1.3\gamma_{23} + 4.5\gamma_{31}\}$.

Differently from Section 7.2.1, it was chosen to show only one EMSE learning

curve to avoid an overwhelming amount of similar figures[1]. The EMSE learning curve is depicted in Figure 12 together with steady-state MSE and EMSE for several values of M. Note how the theoretical and experimental values agree.

All in all, the AF is shown to be capable of estimating a weight array with rotor-valued quantities. Thus, resorting to the isomorphism between $\mathcal{G}^+(\mathbb{R}^3)$ and quaternion algebra (see Section 3.3.3), that filter is naturally suited for estimating weight arrays whose entries are quaternions. This way, the GA-LMS becomes an alternative to the quaternion-LMS (QLMS) [26, 27, 29, 31].

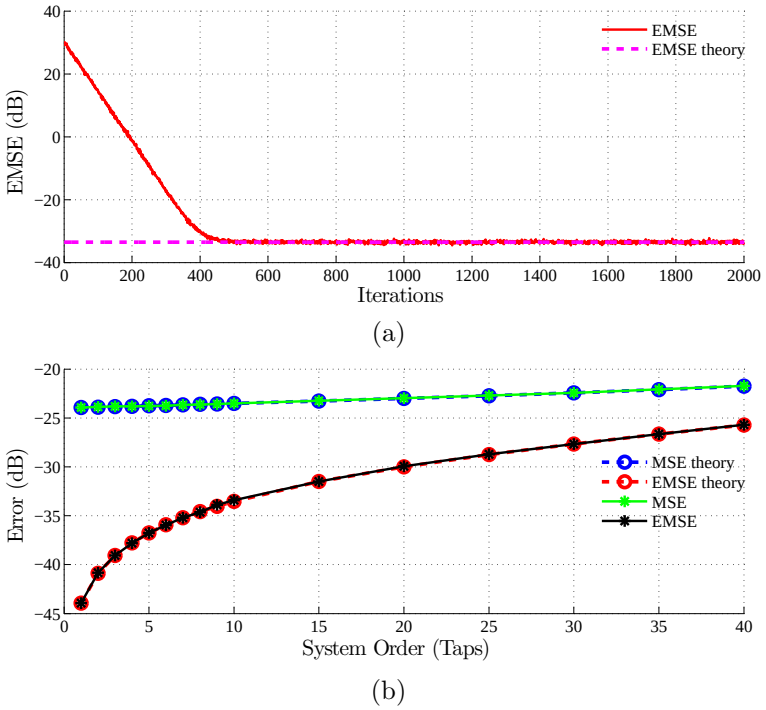


Figure 12: Rotor entries. (a) EMSE learning curve for $M = 10$, $\mu = 0.005$, and $\sigma_v^2 = 10^{-3}$ (100 experiments). (b) Steady-state MSE and EMSE versus the number of taps for $\mu = 0.005$ and $\sigma_v^2 = 10^{-3}$.

[1]All the C++ source codes, MATLAB® scripts, and instructions necessary to generate the learning curves are available on openga.org. The reader is encouraged to explore the online material in order to see the GAAFs performance in several different scenarios.

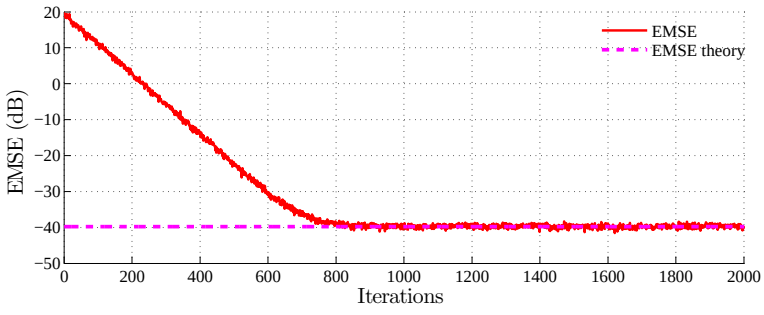
7.2.3 Complex Entries

The underlying geometric algebra in this section is $\mathcal{G}^+(\mathbb{R}^2)$ (isomorphic to complex numbers – see Section 3.3.2), and the optimal weight array is

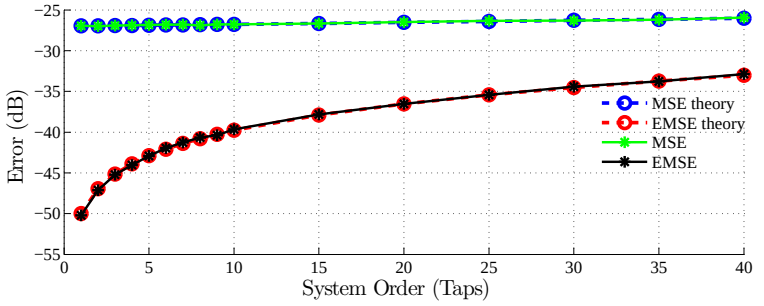
$$w^o = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_M \end{bmatrix} = \begin{bmatrix} 0.55 + 0.71\gamma_{12} \\ 0.55 + 0.71\gamma_{12} \\ \vdots \\ 0.55 + 0.71\gamma_{12} \end{bmatrix}, \tag{7.4}$$

where all complex entries are the same, namely $\{0.55 + 0.71\gamma_{12}\}$.

Figure 13 shows an EMSE learning curve for $M = 10$ and steady-state errors for several values of M. The GA-LMS is shown to be capable of estimating a weight array with complex-valued quantities and the experimental values corroborate the theory. Thus, the GA-LMS becomes an alternative to the complex-LMS (CLMS) [32].



(a)



(b)

Figure 13: Complex entries. (a) EMSE learning curve for $M = 10$, $\mu = 0.005$, and $\sigma_v^2 = 10^{-3}$ (100 experiments). (b) Steady-state MSE and EMSE versus the number of taps for $\mu = 0.005$ and $\sigma_v^2 = 10^{-3}$.

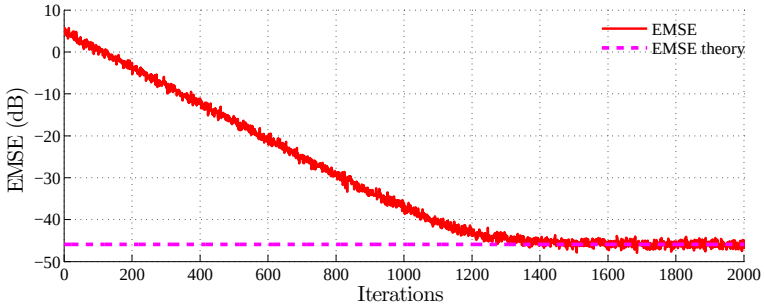
7.2.4 Real Entries

Finally, the most basic type of LMS, i.e., the one that estimates arrays with real-valued entries, is recovered via isomorphism with the geometric algebra $\mathcal{G}^+(\mathbb{R})$. Thus, the optimal weight array is

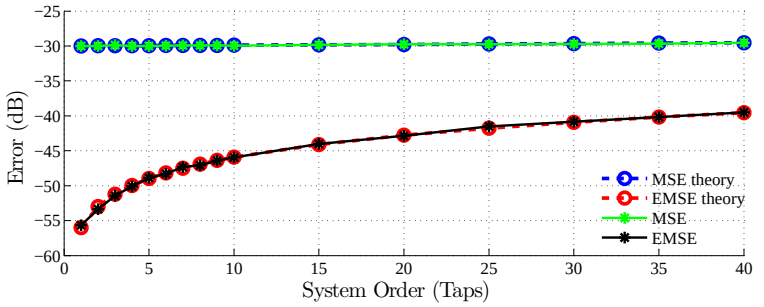
$$w^o = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_M \end{bmatrix} = \begin{bmatrix} 0.55 \\ 0.55 \\ \vdots \\ 0.55 \end{bmatrix}, \tag{7.5}$$

where all real entries are the same, namely 0.55 (recall that $\gamma_0 = 1$). The array w^o in (7.5) has the same shape of the so-called *weight vector* from regular adaptive filtering theory [1, 6].

Figure 14 shows an EMSE learning curve for $M = 10$ and steady-state errors for several values of M. Once again, theoretical and experimental values agree.



(a)



(b)

Figure 14: Real entries. (a) EMSE learning curve for $M = 10$, $\mu = 0.005$, and $\sigma_v^2 = 10^{-3}$ (100 experiments). (b) Steady-state MSE and EMSE versus the number of taps for $\mu = 0.005$ and $\sigma_v^2 = 10^{-3}$.

7.3 3D Registration of Point Clouds with GAAFs for Pose Estimation

This section uses the GA-LMS for pose estimation to align (register) a pair of 3D point clouds (explained in Section 2.3).

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 cubic PCDs. Then, the AF performance is further tested when registering two PCDs from the "Stanford Bunny", one of the most popular 3D datasets [47].

The GA-LMS is implemented using the GAALET C++ library [48] 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 $r = 0.5 + 0.5\gamma_{12} + 0.5\gamma_{23} + 0.5\gamma_{31}$ ($|r| = 1$).

7.3.1 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 $\sigma_v^2 \in \{0, 10^{-9}, 10^{-5}, 10^{-2}\}$.

Figure 15 shows curves of the excess mean-square error ($\text{EMSE}(i) = \text{E}|y_i - r_{i-1}x_i\widetilde{r}_{i-1}|^2$) averaged over 200 realizations. Figure 15 (top) depicts the typical trade-off between 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

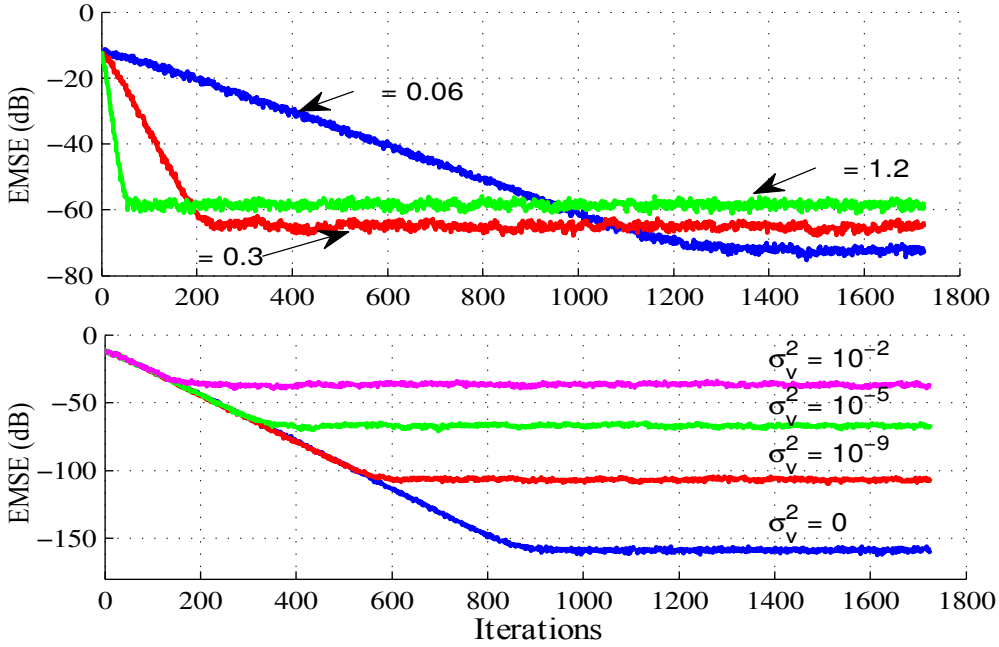
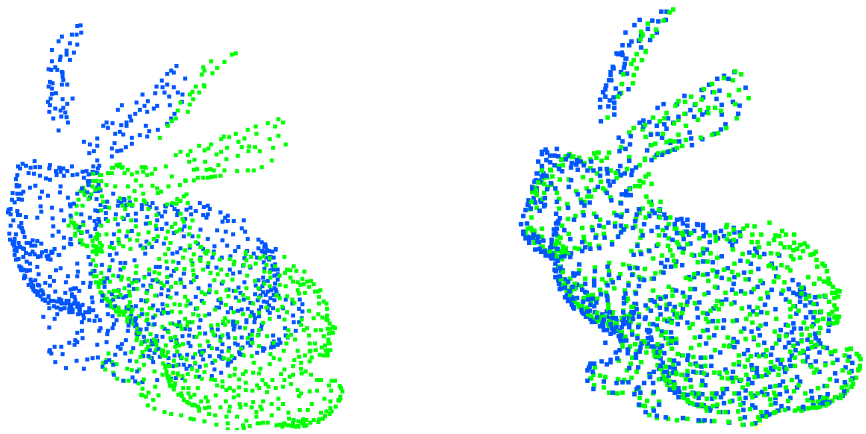


Figure 15: Cube set. (top) EMSE for $\sigma_v^2 = 10^{-5}$ and different values of μ. (bottom) EMSE for $\mu = 0.2$ and different noise variances σ_v^2. For all cases, the steady state is achieved using only part of the correspondence points. The curves are averaged over 200 realizations.

pairs. Figure 15 (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 Figure 15 (bottom), the SVD-based method [42] implemented by the Point Cloud Library (PCL) [49] achieves similar results except for $\sigma_v^2 = 0$, when SVD reaches $-128dB$ compared to $-158dB$ of GA-LMS.

7.3.2 Bunny registration

Two specific scans of the "Stanford Bunny" dataset [47] are selected (see Figure 16), 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 [10]. It suffices to say the point matching is not perfect and hence



(a) (b)

Figure 16: PCDs of the bunny set. (a) Unaligned, (b) after GA-LMS alignment.

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 Figure 17. It shows the curve (in blue) for the mean-square error (MSE), which is approximated by the instantaneous squared error ($\text{MSE}(i) \approx |d_i - r_{i-1}x_i\widetilde{r}_{i-1}|^2$), 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 (6.5) 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 [2].

[2] A video showing the alignment of the PCD sets is available on openga.org

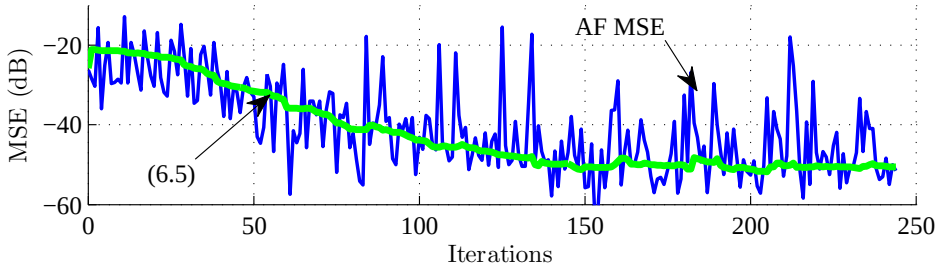


Figure 17: Bunny set, $\mu = 8$. The cost function (6.5) curve is plotted on top of the MSE to emphasize the minimization performed by the AF. The steady state is reached before using all the available correspondences.

8 CONCLUSION

The formulation of GA-based adaptive techniques is still in its infancy. The majority of AF algorithms available in the literature resorts to specific subalgebras of GA (real, complex numbers and quaternions). Each of them requires an specific set of tools in order to pose the problem and perform calculus. In this sense, the development of the GAAFs is an attempt to unify those different adaptive-filtering approaches under the same mathematical language. Additionally, as shown throughout the text, GAAFs have improved estimation capabilities since they are not limited to 1-vector estimation (like LA-based AFs). Instead, they can naturally estimate any kind of multivector. Also, given a type of GAAF, say GA-LMS, the shape of its update rule is invariant with respect to the multivector subalgebra. This is only possible due to the use of GA and GC.

On top of the theoretical contributions, the experimental validation provided in Chapter 7 shows that the GAAFs are ready for use in two key applications: system identification and 3D registration of PCDs. Nevertheless, it must be said that there might exist a number of different applications in need of a tool like GAAFs. It is expected that any estimation problem posed in terms of hyper-complex quantities could benefit from this work. For instance, GAAFs may be useful in data fusion, where different signals are supposed to be integrated in the same "package" and then processed. The multivector (and by extension the array of multivectors) can be interpreted as a fundamental information package that aggregates scalar, vector, bivector, and so on, quantities.

Besides storing all the codes and scripts to perform the experiments, the website openga.org is an interface between engineers and researchers interested in contributing for the development of GA-based algorithms (not necessarily AFs). Thus, the possibilities of new GAAF applications can be amplified by the network of contributors that will (hopefully) expand as time goes by. The reader is encouraged to download the source code, develop his own ideas, and communicate the results to openga.org. Also, feedback about the codes and scripts, as well as application ideas for GAAFs are welcomed and appreciated.

New types of GAAFs are currently under study, particularly the NLMS and RLS variants for system ID and the GA-NLMS and GA-RLS for pose estimation (Appendix A). Based on the achieved results, the use of GAAFs in computer vision is quite promising [44, 46] since the alignment of PCDs is a subtask present in many applications. However, the mean-square analysis of the GAAFs for pose estimation is rather challenging and only incipient results were achieved so far. This figures as the main topic to be explored in the list of future works.

Finally, the combination of GA and AF theories shows that the use of more comprehensive mathematics can indeed circumvent limitations of already-known tools, leading the way to pose new questions and formulate new problems.

APPENDIX A – GA-NLMS AND GA-RLS FOR POSE ESTIMATION

This appendix provides the derivation of two GAAFs for pose estimation, namely GA-NLMS and GA-RLS.

A.1 Laplacian of the Pose Estimation Cost Function

To calculate $\partial_r^2 J(r)$, the Laplacian operator (3.37) is applied to (6.8),

$$\partial_r^2 J(r) = -2 \sum_{k=1}^{K} (\partial_r * \partial_r) \langle y_k r x_k \widetilde{r} \rangle, \tag{A.1}$$

where

$$(\partial_r * \partial_r) \langle y_k r x_k \widetilde{r} \rangle = (\partial_r * \partial_r) \langle r M_k \rangle + (\partial_r * \partial_r) \langle T_k \widetilde{r} \rangle, \tag{A.2}$$

in which $M_k = x_k \widetilde{r} y_k$ and $T_k = y_k r x_k$. The terms on the right-hand side are calculated by applying the definition of multivector differential (Definition 25)

with $A = \partial_r$,

$$\begin{aligned}
(\partial_r * \partial_r)\langle rM_k\rangle &= \lim_{\tau\to 0}\frac{\Big\langle [r+\tau P(\partial_r)]\, M_k\Big\rangle - \langle rM_k\rangle}{\tau} \\
&= \lim_{\tau\to 0}\frac{\langle rM_k\rangle + \tau\langle P(\partial_r)M_k\rangle - \langle rM_k\rangle}{\tau} \\
&= \langle P(\partial_r)M_k\rangle = \langle \partial_r M_k\rangle \\
&= \left\langle \Big(\sum_J a^J a_J * \partial_r\Big) x_k \widetilde{r} y_k \right\rangle \\
&= \left\langle \Big(\underbrace{\sum_J a^J \widetilde{a}_J}_{=d}\Big) x_k y_k \right\rangle \\
&= d\langle x_k y_k\rangle,
\end{aligned} \tag{A.3}$$

where d is the dimension of the algebra where the multivectors are defined, and

$$\begin{aligned}
(\partial_r * \partial_r)\langle T_k r\rangle &= \lim_{\tau\to 0}\frac{\Big\langle T_k\,[r+\tau P(\partial_r)]\Big\rangle - \langle T_k\widetilde{r}\rangle}{\tau} \\
&= \therefore = d\langle x_k y_k\rangle.
\end{aligned} \tag{A.4}$$

Thus, (A.2) becomes

$$(\partial_r * \partial_r)\langle y_k r x_k \widetilde{r}\rangle = 2d\langle x_k y_k\rangle. \tag{A.5}$$

Plugging (A.5) back into (A.1) yields the Laplacian of the cost function $J(r)$,

$$\boxed{\partial_r^2 J(r) = -4d\sum_{k=1}^{K}\langle x_k y_k\rangle = -4d\sum_{k=1}^{K} y_k \cdot x_k}, \tag{A.6}$$

in terms of the inner product of the pair $\{y_k, x_k\}$.

A.2 GA-NLMS for Pose Estimation

Selecting $B = [\partial_r^2 J(r_{i-1})]^{-1} = \dfrac{-1}{4d}\left[\sum_{k=1}^{K} y_k \cdot x_k + \delta(i)\right]^{-1}$ in (6.7) and substituting in (6.6), the *regularized Newton's recursion* can be defined,

$$r_i = r_{i-1} + \mu\left[\sum_{k=1}^{K} y_k \cdot x_k + \delta(i)\right]^{-1}\left[\sum_{k=1}^{K}(r_{i-1}x_k\widetilde{r}_{i-1}) \wedge y_k\right] r_{i-1}, \tag{A.7}$$

in which the term $\frac{1}{4d}$ was absorbed by μ, the gradient (6.10) was used and the regularization term $\delta(i) \ll 1$ was introduced to avoid division by zero.

The instantaneous approximations are adopted for the Laplacian,

$$\sum_{k=1}^{K} y_k \cdot x_k \approx (y_i \cdot x_i)\,, \tag{A.8}$$

and for the reversed gradient,

$$\left[\sum_{k=1}^{K} (r_{i-1} x_k \widetilde{r}_{i-1}) \wedge y_k\right] r_{i-1} \approx \left[(r_{i-1} x_i \widetilde{r}_{i-1}) \wedge y_i\right] r_{i-1}. \tag{A.9}$$

Thus, substituting (A.8) and (A.9) in (A.7), and making $\delta(i) = \delta$ yields the GA-NLMS update rule,

$$\boxed{r_i = r_{i-1} + \frac{\mu}{(y_i \cdot x_i) + \delta} \left[(r_{i-1} x_i \widetilde{r}_{i-1}) \wedge y_i\right] r_{i-1}}. \tag{A.10}$$

A.3 GA-RLS for Pose Estimation

The GA-RLS adopts an *exponentially weighted average* approximation for the Laplacian [1, p. 198],

$$\sum_{k=1}^{K} y_k \cdot x_k \approx \frac{1}{i+1} \sum_{k=1}^{i} \eta^{i-k} \left(y_k \cdot x_k\right), \tag{A.11}$$

in which η is the GA-RLS *forgetting factor*.

The step-size value is chosen to gradually decrease with time

$$\mu \triangleq \mu(i) = \frac{1}{i+1}, \tag{A.12}$$

and the regularization term is defined as [1]

$$\delta(i) = \frac{\eta^{i+1}\delta}{i+1}. \tag{A.13}$$

Plugging (A.11), (A.12), and (A.13) into the regularized Newton's recur-

sion (A.7) yields

$$\begin{aligned} r_i &= r_{i-1} + \frac{1}{i+1}\left[\frac{1}{i+1}\sum_{k=1}^{i}\eta^{i-k}\left(y_k \cdot x_k\right) + \frac{\eta^{i+1}\delta}{i+1}\right]^{-1}\left[(rx_i\widetilde{r}) \wedge y_i\right] r \\ &= r_{i-1} + \underbrace{\left[\sum_{k=1}^{i}\eta^{i-k}\left(y_k \cdot x_k\right) + \eta^{i+1}\delta\right]}_{\Theta(i)}^{-1}\left[(rx_i\widetilde{r}) \wedge y_i\right] r, \end{aligned} \tag{A.14}$$

where the instantaneous approximations were used for the reversed gradient.

Note that $\Theta(i)$ is always scalar-valued and satisfies the following recursion

$$\Theta(i) = \eta\Theta(i-1) + y_i \cdot x_i, \quad \Theta(-1) = \delta. \tag{A.15}$$

Defining $\Phi \triangleq \Theta^{-1}$ and using the *matrix inversion lemma*[1] (see [1], p.78)

$$\Phi(i) = \eta^{-1}\left[\Phi(i-1) - \frac{\eta^{-1}\Phi(i-1)(y_i \cdot x_i)\Phi(i-1)}{1+\eta^{-1}\Phi(i-1)(y_i \cdot x_i)}\right]. \tag{A.16}$$

Finally, substituting (A.16) in (A.14) yields the GA-RLS update rule

$$\boxed{r_i = r_{i-1} + \Phi(i)\left[(rx_i\widetilde{r}) \wedge y_i\right] r}. \tag{A.17}$$

[1]Since Θ is always a scalar, the lemma assumes the simple form $(a+bcd)^{-1} = a^{-1} - a^{-1}b(c^{-1}+da^{-1}b)^{-1}da^{-1}$, with $a = \eta\Theta(i-1)$, $b = y_{i+1} \cdot x_{i+1}$, and $c = d = 1$.

REFERENCES

[1] A.H. Sayed, *Adaptive filters*, Wiley-IEEE Press, 2008.

[2] I. Kleiner, *A History of Abstract Algebra*, Birkhäuser Boston, 2007.

[3] D. Hestenes and G. Sobczyk, *Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics*, Fundamental Theories of Physics. Springer Netherlands, 1987.

[4] D. Hestenes, *New Foundations for Classical Mechanics*, Fundamental Theories of Physics. Springer, 1999.

[5] 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.

[6] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, Springer US, 4 edition, 2013.

[7] R. B. Rusu, *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*, Ph.D. thesis, Computer Science department, Technische Universität München, Germany, October 2009.

[8] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 1999, vol. 2, pp. 1150–1157 vol.2.

[9] G. Schroth, R. Huitl, D. Chen, M. Abu-Alqumsan, A. Al-Nuaimi, and E. Steinbach, "Mobile visual location recognition," *Signal Processing Magazine, IEEE*, vol. 28, no. 4, pp. 77–89, 2011.

[10] A. Al-Nuaimi, M. Piccolorovazzi, S. Gedikli, E. Steinbach, and G. Schroth, "Indoor location recognition using shape matching of kinectfusion scans to large-scale indoor point clouds.," in *Eurographics, Workshop on 3D Object Retrieval*, 2015.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[12] J. Vaz Jr. and R. da Rocha Jr., *Álgebras de Clifford e Espinores*, Livraria da Física, 2012.

[13] C. J. L. Doran, *Geometric Algebra and Its Application to Mathematical Physics*, Ph.D. thesis, University of Cambridge, 1994.

[14] C.J.L. Doran and A.N. Lasenby, *Geometric Algebra for Physicists*, Cambridge University Press, 2003.

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

[16] 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.

[17] C. Perwass, *Geometric Algebra with Applications in Engineering*, Geometry and Computing. Springer Berlin Heidelberg, 2009.

[18] D. Hildenbrand, *Foundations of Geometric Algebra Computing*, Geometry and Computing. Springer Berlin Heidelberg, 2012.

[19] L. Dorst, C. Doran, and J. Lasenby, *Applications of Geometric Algebra in Computer Science and Engineering*, Birkhäuser Boston, 2012.

[20] G. Sommer, *Geometric Computing with Clifford Algebras: Theoretical Foundations and Applications in Computer Vision and Robotics*, Springer Berlin Heidelberg, 2013.

[21] M.J. Crowe, *A History of Vector Analysis: The Evolution of the Idea of a Vectorial System*, Dover Books on Mathematics Series. Dover, 1967.

[22] G.C. Rota, *Indiscrete Thoughts*, Modern Birkhäuser Classics. Birkhäuser Boston, 2009.

[23] W. K. Clifford, *The Concepts of Space and Time: Their Structure and Their Development*, chapter On the Space-Theory of Matter, pp. 295–296, Springer Netherlands, Dordrecht, 1976.

[24] W. K. Clifford, *The Concepts of Space and Time: Their Structure and Their Development*, chapter On the Bending of Space, pp. 291–294, Springer Netherlands, Dordrecht, 1976.

[25] D. P. Mandic and V. S. L. Goh, *Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models*, John Wiley & Sons, 2009.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] M. Jiang, W. Liu, and Y. 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.

[31] D. P. Mandic D. Xu, Y. Xia, "Optimization in quaternion dynamic systems: Gradient, Hessian, and learning algorithms," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 249–261, Feb 2016.

[32] B. Widrow, J. McCool, and M. Ball, "The complex LMS algorithm," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 719–720, April 1975.

[33] E. B. Dam, M. Koch, and M. 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.

[34] P. R. Girard, *Quaternions, Clifford Algebras and Relativistic Physics*, Birkhäuser Basel, 2007.

[35] E. Hitzer, "Multivector differential calculus," *Advances in Applied Clifford Algebras*, vol. 12, no. 2, pp. 135–182, 2002.

[36] E. Hitzer, "Algebraic foundations of split hypercomplex nonlinear adaptive filtering," *Mathematical Methods in the Applied Sciences*, vol. 36, no. 9, pp. 1042–1055, 2013.

[37] 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.

[38] Z. 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.

[39] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, 2001.

[40] 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.

[41] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-d location parameters using dual number quaternions," *CVGIP: Image Underst.*, vol. 54, no. 3, pp. 358–367, Oct. 1991.

[42] 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.

[43] J.B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*, Princeton paperbacks. Princeton University Press, 1999.

[44] W. B. Lopes, A. Al-Nuaimi, and C. G. Lopes, "Geometric-algebra lms adaptive filter and its application to rotation estimation," *IEEE Signal Processing Letters*, vol. 23, no. 6, pp. 858–862, June 2016.

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

[46] A. Al-Nuaimi, W. B. Lopes, E. Steinbach, and C. G. Lopes, "6DOF point cloud alignment using geometric algebra-based adaptive filtering," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2016, pp. 1–9.

[47] G. Turk and M. 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.

[48] F. Seybold and U. Wössner, "Gaalet - a C++ expression template library for implementing geometric algebra," in *6th High-End Visualization Workshop*, 2010.

[49] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.