Geometric Algebra for Computer Science

Geometric Algebra for Computer Science

An Object-oriented Approach to Geometry

LEO DORST

DANIEL FONTIJNE

STEPHEN MANN

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann Publishers is an imprint of Elsevier

MORGAN KAUFMANN PUBLISHERS

Publishing Director	Denise Penrose
Senior Acquisitions Editor	Tim Cox
Publishing Services Manager	George Morrison
Senior Project Manager	Brandy Lilly
Editorial Assistant	Michelle Ward
Text Design	Multiscience Press, Inc.
Composition	diacriTech
Technical Illustration	diacriTech
Copyeditor	Multiscience Press, Inc.
Proofreader	Multiscience Press, Inc.
Indexer	Multiscience Press, Inc.
Interior printer	Hing Yip Printing Co., Ltd.
Cover printer	Hing Yip Printing Co., Ltd.

Morgan Kaufmann Publishers is an imprint of Elsevier.
500 Sansome Street, Suite 400, San Francisco, CA 94111

This book is printed on acid-free paper.

Library of Congress Cataloging-in-Publication Data
Application submitted

ISBN 13: 978-0-12-369465-2
ISBN10: 0-12-369465-5

For information on all Morgan Kaufmann publications,
visit our Web site at www.mkp.com or www.books.elsevier.com

Printed in China
07 08 09 10 11 5 4 3 2 1

Contents

List of Figures

List of Tables

List of Programming Examples

Preface

Geometric algebra is a powerful and practical framework for the representation and solution of geometrical problems. We believe it to be eminently suitable to those subfields of computer science in which such issues occur: computer graphics, robotics, and computer vision. We wrote this book to explain the basic structure of geometric algebra, and to help the reader become a practical user. We employ various tools to get there:

- Explanations that are not more mathematical than we deem necessary, connecting algebra and geometry at every step
- A large number of interactive illustrations to get the "object-oriented" feeling of constructions that are dependent only on the geometric elements in them (rather than on coordinates)
- Drills and structural exercises for almost every chapter
- Detailed programming examples on elements of practical applications
- An extensive section on the implementational aspects of geometric algebra (Part III of this book)

This is the first book on geometric algebra that has been written especially for the computer science audience. When reading it, you should remember that geometric algebra is fundamentally simple, and fundamentally simplifying. That simplicity will not always be clear; precisely because it is so fundamental, it does basic things in a slightly different way and in a different notation. This requires your full attention, notably in the beginning, when we only seem to go over familiar things in a perhaps irritatingly different manner. The patterns we uncover, and the coordinate-free way in which we encode them, will all pay off in the end in generally applicable quantitative geometrical operators and constructions.

We emphasize that this is not primarily a book on programming, and that the subtitle "An Object-oriented Approach to Geometry" should not be interpreted too literally. It is intended to convey that we finally achieve clean computational "objects" (in the sense of object-oriented programming) to correspond to the oriented elements and operators of geometry by identifying them with "oriented objects" of the algebra.

AUDIENCE

The book is aimed at a graduate level; we only assume basic linear algebra (and a bit of calculus in Chapter 8). No prior knowledge of the techniques of computer graphics or robotics is required, though if you are familiar with those fields you will appreciate how much easier things are in geometric algebra. The book should also be well suited for self-study at the post-graduate level; in fact, we tried to write the book that we would have wanted ourselves for this purpose. Depending on your level of interest, you may want to read it in different ways.

- If you are a seasoned user of geometry and well-versed in the techniques of casting geometry in linear algebra, but don't have much time, you will still find this book worthwhile. In a comfortable reading, you can absorb what is different in geometric algebra, and its structure will help you understand all those old tricks in your library. In our experience, it makes many arcane techniques comprehensible, and it helped us to learn from useful math books that we would otherwise never have dared to read. You may never actually use geometric algebra, but you will find it enlightening all the same. And who knows—you may come back for more.
- If you are currently writing code using the coordinate-based linear algebra, a background study of the techniques in this book will be helpful and constructive. The advantages for the previous category will apply to you as well. Moreover, you may find yourself doing derivations of formulas you need to program in the compact geometric algebra manner, and this will clarify and improve your implementations, even if you continue writing those in the old linear algebra vocabulary. In particular, the thinking behind your code will be more geometrical, less coordinate-based, and this will make it more transparent, more flexibly applicable (for instance, in higher dimensions), and ready to be translated into geometric algebra after the revolution.
- If you are starting out in geometric programming, take the time to absorb this book thoroughly. This geometric algebra way of thinking is quite natural, and we are rather envious that you can learn it from scratch, without having to unlearn old methods. With study and practice you will be able to write programs in geometric algebra rather fluently, and eventually contribute actively to its development.

Our style in this book is factual. We give you the necessary mathematics, but always relate the algebra to the geometry, so that you get the complete picture. Occasionally, there is a need for more extensive proofs to convince you of the consistency of aspects of the framework. When such a proof became too lengthy and did not further the arguments, it was relegated to an appendix. The derivations that remain in the text should be worth your time, since they are good practice in developing your skills. We have attempted to avoid the "pitfall of imprecision" in this somewhat narrative presentation style by providing the fundamental chapters with a summary of the essential results, for easy consultation via the index.

HISTORY

We do not constantly attribute all results, but that does not mean that we think that we developed all this ourselves. By its very nature, geometric algebra collates many partial results in a single framework, and the original sources become hard to trace in their original context. It is part of the pleasure of geometric algebra that it empowers the user; by mastering just a few techniques, you can usually easily rediscover the result you need.

Once you grasp its essence, geometric algebra will become so natural that you will wonder why we have not done geometry this way all along. The reason is a history of geometric (mis)representation, for almost all elements of geometric algebra are not new—in hindsight. Elements of the quantitative characterization of geometric constructions directly in terms of its elements are already present in the work of René Descartes (1595–1650); however, his followers thought it was easier to reduce his techniques to coordinate systems not related to the elements (nevertheless calling them Cartesian, in his honor). This gave us the mixed blessing of coordinates, and the tiresome custom of specifying geometry at the coordinate level (whereas coordinates should be relegated to the lowest implementational level, reserved for the actual computations). To have a more direct means of expression, Hermann Grassmann (1809–1877) developed a theory of extended quantities, allowing geometry to be based on more than points and vectors. Unfortunately, his ideas were ahead of their time, and his very compact notation made his work more obscure than it should have been. William Rowan Hamilton (1805–1865) developed quaternions for the algebra of rotations in 3D, and William Kingdon Clifford (1845–1879) defined a more general product between vectors that could incorporate general rigid body motions.

All these individual contributions pointed toward a geometric algebra, and at the end of the 19th century, there were various potentially useful systems to represent aspects of geometry. Gibbs (1839–1903) made a special selection of useful techniques for the 3D geometry of engineering, and this limited framework is basically what we have been using ever since in the geometrical applications of linear algebra. In a typical quote from his biography "using ideas of Grassmann, Gibbs produced a system much more easily applied to physics than that of Hamilton." In the process, we lost geometric algebra. Linear algebra and matrices, with their coordinate representations, became the mainstay of doing geometry, both in practice and in mathematical development. Matrices work, but in their usual form they only work on vectors, and this ignores Grassmann's insight that extended qualities can be elements of computation. (Tensors partially fix this, but in a cumbersome coordinate-based notation.)

With the arrival of quantum physics, convenient alternative representations for spatial motions were developed (notably for rotations), using complex numbers in "spinors." The complex nature of spinors was mistaken for an essential aspect of quantum mechanics, and the representations were not reapplied to everyday geometry. David Hestenes (1933–present) was perhaps the first to realize that the representational techniques in relativity and quantum mechanics were essentially manifestations of a fundamental

"algebra of spatial relationships" that needed to be explored. He rescued the half-forgotten geometric algebra (by now called Clifford algebra and developed in nongeometric directions), developed it into an alternative to the classical linear algebra–based representations, and started advocating its universal use. In the 1990s, his voice was heard, and with the implementation of geometric algebra into interactive computer programs its practical applicability is becoming more apparent.

We can now finally begin to pick up the thread of geometrical representation where it was left around 1900. Gibbs was wrong in assuming that computing with the geometry of 3D space requires only representations of 3D points, although he did give us a powerful system to compute with those. This book will demonstrate that allowing more extended quantities in higher-dimensional representational spaces provides a more convenient executable language for geometry. Maybe we could have had this all along; but perhaps we indeed needed to wait for the arrival of computers to appreciate the effectiveness of this approach.

SOFTWARE

There are three main software packages associated with this book, each written with a different goal in mind (interaction, efficiency and illustration of algorithms, respectively). All three were developed by us, and can be found on the web site:

http://www.geometricalgebra.net

for free downloading.

- `GAViewer` is an interactive program that we used to generate the majority of the figures in this book. It was originally developed as a teaching tool, and a web tutorial is available, using `GAViewer` to explain the basics of geometric algebra. You can use `GAViewer` when reading the book to type in algebraic formulas and have them act on geometrical elements interactively. This interaction should aid your understanding of the correspondence between geometry and algebra considerably. The (simplified) code of the figures provides a starting point for your own experimentation.
- `Gaigen 2` is geometric algebra implementation in C++ (and Java), intended for applications requiring more speed and efficiency than a simple tutorial. The GA sandbox source code package used for the programming examples and exercises in this book is built on top of `Gaigen 2`. To compile and run the programming examples in Part I and Part II, you only have to download the sandbox package from the web site.
- Our simplistic but educational "reference implementation" implements all algorithms and techniques discussed in Part III. It is written in Java and intended to show only the essential structure; we do not deem it usable for anything that is computationally intensive, since it can easily be 10 to 100 times slower than `Gaigen 2`.

If you are serious about implementing further applications, you can start with the GA sandbox package, or other available implementations of geometric algebra, or even write your own package.

ACKNOWLEDGEMENTS

Of those who have helped us develop this work, we especially thank David Hestenes, not only for reinvigorating geometric algebra, but also for giving Leo an early introduction to the conformal model at a half-year sabbatical at Arizona State University. We are grateful to Joan Lasenby of Cambridge University for her detailed comments on the early chapters, and for providing some of the applied examples. We are also indebted to Timaeus Bouma for his keen insights that allowed our software to be well-founded in mathematical fact.

We gratefully acknowledge the support of the University of Amsterdam, especially professor Frans Groen; NWO (Netherlands Organization for Scientific Research) in project 612.012.006; and NSERC (Natural Sciences and Engineering Research Council of Canada).

Ultimately, though, this book would have been impossible without the home front:

Leo Dorst's parents and his wife Phyllis have always utterly supported him in his quest to understand new aspects of math and life; he dedicates this book to them.

Daniel Fontijne owes many thanks to Yvonne for providing the fun and artistic reasons to study geometric algebra, and to Femke and Tijmen for the many refreshing breaks while working at home.

Stephen Mann would like to thank Mei and Lilly for their support during the writing of this book.

Geometric Algebra for Computer Science