

Geometry and Computing

Series Editors

Herbert Edelsbrunner
Leif Kobbelt
Konrad Polthier

Editorial Advisory Board

Jean-Daniel Boissonnat
Gunnar Carlsson
Bernard Chazelle
Xiao-Shan Gao
Craig Gotsman
Leo Guibas
Myung-Soo Kim
Takao Nishizeki
Helmut Pottmann
Roberto Scopigno
Hans-Peter Seidel
Steve Smale
Peter Schröder
Dietrich Stoyan

For further volumes:

<http://www.springer.com/series/7580>

Dietmar Hildenbrand

Foundations of Geometric Algebra Computing

Dr. Dietmar Hildenbrand
University of Technology Darmstadt
Darmstadt
Germany

ISSN 1866-6795 ISSN 1866-6809 (electronic)
ISBN 978-3-642-31793-4 ISBN 978-3-642-31794-1 (eBook)
DOI 10.1007/978-3-642-31794-1
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012953093

ACM Computing Classification (1998): B.8.2, C.1.4, D.1.3, G.1.2, G.4, I.3.5, F.2.2

Mathematics Subject Classification (2000): 65Dxx, 68U05

© Springer-Verlag Berlin Heidelberg 2013, Corrected printing 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

*To 150 years of Hermann G. Grassmann's
Ausdehnungslehre of 1862*

Foreword

If I have seen further it is by standing on the shoulders of giants.
– I. Newton, 15 Feb 1676.

Newton’s well-known quote to Hooke finds no better illustration than in the development of what we call Geometric Algebra. The primary set of shoulders in the Geometric Algebra epic belong to Hermann Günther Grassmann, who epitomized the mathematical mind, at least in process, if not in expository style. He systematically created algebras for geometric concepts by listing essential elements and operations and then reducing them to minimal axiomatic schemes. In his book *Die Lineale Ausdehnungslehre* he laid the foundation for scores of mathematical systems bearing names like projective and exterior geometry and Quaternion, Clifford, Gibbs, Cartan, and Boolean algebra. Unfortunately, his book was densely written with many religious allusions and it shifted without warning between projective and Euclidean spaces.

One of the fundamental ideas of Grassmann was that of different grade (intrinsic dimensions) elements; thus, in addition to scalar and vector elements, with which we are well familiarized today, he would have included bi-vectors, tri-vectors etc., not as combinations of vectors like the cross-product, but as axiomatic elements. His book introduced two fundamental operations between elements, the inner and outer products. One was simply a grade lowering and the other a grade elevating operation between elements. Between vectors, these operations became the familiar dot product (or projection) and the wedge product (or plane through the vectors). The import of these ideas becomes apparent when one considers what else can one say about the relation of two vectors? The grade lowering and grade elevating operators underpin any comprehensive vector algebra.

It was William Kingdon Clifford, who then recognized the sublime value of joining these two operations, inner and outer, into one product via a new algebraic “addition” of the two products. The geometric product of Clifford was cast as the fundamental operation with inner and outer products defined in terms of it. Its subtlety lies in the fact that it is not a binary operation as we are used to thinking of

additions; it does not replace two elements with a third “sum” of the same type of element. Instead, it is a prepositional operation that only makes sense in the overall algebraic structure. It is rather like adding real and imaginary parts in complex analysis.

The subsequent evolution of “Clifford” Algebra became a candidate for the mathematical model of nineteenth-century physics, along with the quaternions of Hamilton. The winner of this intellectual contest, however, was eventually another Grassmannian derivative, the vector algebra proposed by Josiah Gibbs, a thermodynamicist. Gibbs’ focus was three dimensions, appropriate to his area of research. As such he considered scalars and vectors to be sufficient; the bi-vector could be represented by the cross product, another vector. Hence there was no need for bi-vectors, and certainly not tri-vectors, nor anything higher. Two elements, the scalar and the vector, seem to satisfy Occam’s razor. This may well be true, but only for three dimensions. As physics evolved in higher dimensions, the inadequacies of the cross product have become apparent. Pauli and Dirac invented their own versions of spinors, for example, a concept which has since been shown to be fundamental in any comprehensive formulation of an axiomatic physics system.

Our current state of babel has different mathematical systems, which require sophisticated methods of translation between them. They are often so common to us now that we do not see the clumsiness of, say, rewriting complex analysis in terms of 2D vectors, or 3D vectors in terms of quaternions to rotate and rewriting back. Similar situations exist between exterior algebra, Pauli and linear algebras, and the list goes on. A single, comprehensive algebra is both pedagogically and operationally “a consummation devoutly to be wish’d”. That is the goal of Geometric Algebra.

In the last 50 years, Clifford Algebra found a strong proponent in physicist David Hestenes, who used it as the basis to describe electron theory and celestial mechanics among other very successful applications. His version of the Geometric Algebra was developed with a strong emphasis on the geometrically intuitive aspects of the algebra.

While Geometric Algebra is generally acknowledged to be a compelling and comprehensive system of mathematics (and it is beginning to find traction in many application areas) one major obstacle exists to its broader adoption, which is the very practical one. How do I compute with it? Without a Maple- or Mathematica-like facility, its usability is vastly limited in today’s modern research or engineering environment. The way forward is obvious and a number of researchers have addressed this problem with computer algebraic systems based on Geometric Algebra. Two of the most current and popular are CLUCalc and Gaalop, as described in this book.

Experience with these methods and the innate characteristics of Geometric Algebra now point to the next logical step in the evolution. It is the need to use modern parallel architecture to accelerate Geometric Algebra. We can not only increase the range of realizable applications through speed and efficiency, but it provides unique and valuable insights into the algebra itself. This is the natural

evolution and critical path to bringing this richer, more comprehensive system of mathematics, this Geometric Algebra, to the collective scientific consciousness.

This book by Hildenbrand is, in my opinion, the next, necessary and joyful twist in this elegant evolutionary thread stretching back to Grassmann and beyond. He gives a highly readable account of the development of Geometric Algebra. He is able to cook the subject matter like a good meal, and then, pulling all the pieces together, feeds us a very compelling solution for the next steps in creating the most advanced environment for learning, applying and enjoying the beauty of Geometric Algebra.

Bon appetit

Thuwal, Kingdom of Saudi Arabia
May 2012

Prof. Alyn Rockwood

Preface

Hermann G. Grassmann's *Ausdehnungslehre* of 1862 laid the foundations for Geometric Algebra as a mathematical language combining geometry and algebra. A hundred and fifty years later, this book is intended to lay the foundations for the widespread use of this mathematical system on various computing platforms.

Seventeen years after Grassmann's first more philosophically written *Ausdehnungslehre* of 1844, he admitted in the preface of his mathematical version of 1862, "I remain completely confident that the labor I have expended on the science presented here and which has demanded a significant part of my life, as well as the most strenuous application of my powers, will not be lost. It is true that I am aware that the form which I have given the science is imperfect and must be imperfect. But I know and feel obliged to state (though I run the risk of seeming arrogant) that even if this work should again remain unused for another seventeen years or even longer, without entering into the actual development of science, still that time will come when it will be brought forth from the dust of oblivion and when ideas now dormant will bring forth fruit." And he went on to say, "there will come a time when these ideas, perhaps in a new form, will arise anew and will enter into a living communication with contemporary developments."

The form that we give to Geometric Algebra in this book has the power to lead easily from the geometric intuition of solving an engineering application to its efficient implementation on current and future computing platforms. We show how easy it is to develop new algorithms in areas such as computer graphics, robotics, computer animation, and computer simulation. Owing to its geometric intuitiveness, compactness, and simplicity, algorithms based on Geometric Algebra can lead to enhanced quality, a reduction in development time and solutions that are more easily understandable and maintainable. Often, a clear structure and greater elegance result in lower runtime performance. However, based on our computing technology, Geometric Algebra implementations can even be faster and more robust than conventional ones.

I really do hope that this book can support the widespread use of Geometric Algebra Computing technology in many engineering fields.

Darmstadt, Germany
May 2012

Dr. Dietmar Hildenbrand

Acknowledgments

First of all, I would like to express my thanks to Professor Marc Alexa, who put me in touch with Geometric Algebra in 2002 and, as my advisor, gave me the chance to study for a Ph.D. dealing with the application of Geometric Algebra to computer graphics and robotics. Thanks to Marc, one focus of my dissertation was on the runtime performance of Geometric Algebra algorithms.

I am grateful to Professor Eduardo Bayro-Corrochano for good cooperation in past years, especially concerning the application of Geometric Algebra to robotics. Thanks to Eduardo, I got to know the Cliffordlib Maple library, created by Rafal Ablamowicz and Bertfried Fauser, which we could use advantageously for our first runtime optimization study.

Thanks to Dr. habil. Christian Perwass, I was able to use a very powerful and productive tool for the interactive and visual development and testing of Geometric Algebra algorithms; Christian's tool CLUCalc has accompanied my research into Geometric Algebra from its early beginnings to today. I am very grateful for his good cooperation over the last 10 years.

I would like to thank Professor Andreas Koch and his group for their very good cooperation on compiler technology. Thanks to funding from the DFG (Deutsche Forschungsgemeinschaft), we were able to cooperate on a common research project dealing with a Geometric Algebra compiler for FPGAs.

The molecular dynamics simulation described in this book was a joint work with the high-performance computing center at the University of Stuttgart with special thanks to Florian Seybold.

Many thanks to Professor Vaclav Skala for his GraVisMa initiative.

I would like to thank Professor Kanta Tachibana and Professor Yukio Kaneda for the chance to present my early Geometric Algebra Computing results in some tutorials at the Frontiers of Computational Science Center of Excellence in Nagoya, Japan, as well as for the honor of becoming a member of the advisory committee of this center.

Thanks are due for the enthusiastic work of my former students Holger Griesheimer, Stefan Rockensuess, Jun Zhao, Thomas Kalbe, Haidan Zhang, Yusheng Wang, Sebastian Hartte, Carsten Cibura, Elmar Brendel, Florian Woersdoerfer,

Christian Schwinn, Meike Becker, Roman Getto, Crispin Deul, Michael Burger, and Andreas Goerlitz. Thanks to Marco Hutter for proofreading parts of the manuscript. Special thanks to Joe Pitt for the first version of our Gaalop compiler, and to Patrick Charrier and Christian Steinmetz for the current version of Gaalop and the precompilers for C++ and OpenCL, as well as for proofreading of the manuscript.

Thanks to Professor David Hestenes, Professor Hongbo Li, Professor Alyn Rockwood, Prof. Eckhard Hitzer, Professor Rafal Ablamowicz, Professor Gerik Scheuermann, Professor Wolfgang Strasser, Dr. Joan Lasenby, Dr. Leo Dorst, Dr. Daniel Fontijne, Dr. Julio Zamora, Dr. Martin Horn, Professor Paul Drechsel, and Professor Hans-Joachim Petsche for a lot of very fruitful discussions. I would like to thank Prof. Simos and the “*European Society of Computational Methods in Sciences and Engineering*”. They awarded me in 2012 with the highest distinction of “Honorary Fellowship” for my work in the field of Applied Mathematics.

I would like to thank Professor Zoubir for his support and I hope that our LOEWE research priority program Cocoon will benefit a lot from this book.

Finally, I would like to thank my wife Carola for her interest shown in my work and her support, understanding and patience during recent months.

Contents

1	Introduction	1
1.1	The Benefits of Geometric Algebra	2
1.2	The Benefits of Geometric Algebra Computing	5
1.3	History of Geometric Algebra Computing	7
1.4	Overview.....	11
1.5	Outline	12
1.5.1	Part I.....	12
1.5.2	Part II.....	13
1.5.3	Part III.....	13
 Part I Mathematical Foundations		
2	Mathematical Introduction.....	17
2.1	The Basic Algebraic Elements of Geometric Algebra.....	17
2.2	The Products of Geometric Algebra.....	18
2.2.1	The Outer Product	18
2.2.2	The Inner Product.....	20
2.2.3	The Geometric Product.....	20
2.3	Euclidean Geometric Algebra	23
2.4	Projective Geometric Algebra	25
3	Conformal Geometric Algebra.....	27
3.1	The Basic Geometric Entities	28
3.1.1	Points.....	29
3.1.2	Spheres	30
3.1.3	Planes	30
3.1.4	Circles	30
3.1.5	Lines.....	31
3.1.6	Point Pairs.....	31
3.2	IPNS and OPNS	31
3.3	The Center of a Sphere, Circle, or Point Pair	32

3.4	Distances and Angles	33
3.4.1	Distances	34
3.4.2	Angles	38
3.5	Transformations	39
3.5.1	Rotation	39
3.5.2	Translation	40
3.6	Rigid-Body Motion	42
3.7	The Horizon Example	42
4	Maple and the Identification of Quaternions and Other Algebras	45
4.1	Using Maple for Symbolic Geometric Algebra Computing	45
4.2	Complex Numbers	47
4.3	Quaternions	48
4.3.1	The Imaginary Units	50
4.3.2	Pure Quaternions and Their Geometric Product	51
4.3.3	Rotations Based on Unit Quaternions	52
4.4	Plücker Coordinates	53
4.5	Dual Numbers	55
4.6	Dual Quaternions	56
5	Fitting of Planes or Spheres to Sets of Points	61
5.1	The Role of Infinity	61
5.1.1	Sphere of Infinite Radius	62
5.1.2	Point at Infinity	62
5.1.3	Plane at Infinite Distance from the Origin	63
5.1.4	Planes as a Limit of a Sphere	64
5.2	Distance Measure	65
5.3	Least-Squares Approach	65
5.4	Example	67

Part II Interactive and Visual Geometric Algebra Computing

6	A Tutorial on Geometric Algebra Using CLUCalc	71
6.1	Blades and Vectors	73
6.2	The Products of Geometric Algebra	74
6.2.1	The Outer Product and Parallelness	74
6.2.2	The Inner Product and Perpendicularity	77
6.2.3	The Geometric Product and Invertibility	78
6.3	Geometric Operations	81
6.3.1	Projection and Rejection	82
6.3.2	Reflection	83
6.3.3	Rotation in 2D	84
6.3.4	Rotation in 3D	86
6.4	Conformal Geometric Algebra	86
6.4.1	Vectors in CGA	87
6.4.2	Bivectors in CGA	89

6.4.3	Dual Vectors in CGA	91
6.4.4	Is a Point Inside or Outside the Circumcircle of a Triangle?	92
6.4.5	Intersections.....	93
6.4.6	Reflection	96
6.4.7	Projection	97
6.5	CLUCalc Implementation of the Horizon Example	98
6.6	CLUCalc Implementation of Motions.....	99
7	Inverse Kinematics of a Simple Robot	101
7.1	Computation of P_0	102
7.2	Computation of P_2	103
7.3	Computation of P_1	104
7.4	Computation of the Joint Angles	105
8	Robot Grasping an Object.....	107
8.1	The Geometric Algebra Algorithm	107
8.1.1	Computation of the Bounding Volume of the Object	107
8.1.2	Computation of the Grasping Circle Z_t	108
8.1.3	Gripper Circle.....	109
8.1.4	Estimation of Translation and Rotation.....	110
8.2	The Algorithm Using CLUCalc	111
8.3	Geometric Algebra Versus Conventional Mathematics.....	115
8.3.1	The Base Circle and Its Center.....	115
8.3.2	The Transformation of the Gripper	116
 Part III Runtime Performance of Geometric Algebra Computing		
9	Efficient Computer Animation Application in CGA	121
9.1	Optimizations Based on Quaternions.....	121
9.1.1	Direct Computation of Quaternions	122
9.1.2	Efficient Computation of Quaternions	123
9.2	The Inverse Kinematics Algorithm	123
9.2.1	Computation of the Swivel Plane	124
9.2.2	The Elbow Point P_e	125
9.2.3	Calculation of the Elbow Quaternion Q_e	126
9.2.4	Rotation to the Elbow Position.....	127
9.2.5	Rotation to the Wrist Location	128
9.3	Approaches to Runtime Optimization	129
9.3.1	Optimization with Gaigen 2	129
9.3.2	Optimization with Maple	134
9.4	Results	138

10 Using Gaalop for High-Performance Geometric Algebra Computing	141
10.1 The Horizon Example with Gaalop	141
10.2 The Geometric Algebra Computing Approach	142
10.3 Table-Based Compilation Approach	145
10.3.1 Multiplication Tables	146
10.3.2 Table-Based Multiplication Algorithm	148
10.3.3 Example	148
10.3.4 Cascading Multiplications	151
10.3.5 Linear Operation Tables	153
10.3.6 Multiplication Tables with a Non-Euclidean Metric	153
10.3.7 Additional Symbolic Optimizations Using Maxima	154
11 Collision Detection Using the Gaalop Precompiler	155
11.1 Basic Concept of Gaalop GPC	155
11.2 The Horizon Example Revisited in Gaalop GPC for C++	156
11.3 Collision Detection	158
12 The Gaalop Precompiler for GPUs	161
12.1 Strided Arrays	161
12.2 The Horizon Example on a GPU	162
12.2.1 OpenCL Implementation	162
12.2.2 CUDA Implementation	163
12.3 List of Multivector Functions	164
13 Molecular Dynamics Using Gaalop GPC for OpenCL	165
13.1 Molecular Dynamics in a Nutshell	165
13.2 Software Architecture	167
13.3 Initialization	168
13.4 Velocity Verlet Integration Step 1	170
13.5 Accumulation of Forces Per Atom	171
13.6 Velocity Verlet Integration Step 2	176
14 Geometric Algebra Computers	179
14.1 FPGA Implementation of Geometric Algebra Algorithms	179
14.2 Adaptation of Geometric Algebra to Current Computer Architectures	181
14.3 Geometric Algebra Parallelism Programs (GAPP)	182
14.3.1 Example	184
14.3.2 Parallelization Concepts Supported by GAPP	187
14.4 Geometric Algebra Computers Based on Gaalop	188
References	189
Index	195

List of Figures

Fig. 1.1	Two trends that are combined in Geometric Algebra Computing technology: increasingly parallel computing platforms and algebras with more and more geometric meaning	2
Fig. 1.2	The blades of CGA. Spheres and planes, for instance, are vectors. <i>Lines</i> and <i>circles</i> can be represented as bivectors. Other mathematical systems such as complex numbers and quaternions can be identified based on their imaginary units i, j, k . This is why transformations such as rotations can also be handled in the algebra	3
Fig. 1.3	Spheres and circles are basic entities of Geometric Algebra. Operations such as the intersection of two spheres are easily expressed	4
Fig. 1.4	Spheres and lines are basic entities of Geometric Algebra that one can compute with. Operations such as the intersection of these objects are easily expressed with the help of their outer product. In a ray-tracing application, for instance, the result of the intersection of a ray and a (bounding) sphere is another geometric entity: the point pair consisting of the two points where the line intersects the sphere. The sign of the <i>square</i> of the point pair indicates easily whether there is a real intersection or not	5
Fig. 1.5	Hermann Grassmann	7
Fig. 1.6	History of Geometric Algebra and Geometric Calculus [49]	8
Fig. 3.1	Why 5D Conformal Geometric Algebra for 3D world problems?.	28
Fig. 3.2	The inner product of a point and a sphere describes the square of the distance between the point and sphere according to (3.46)	37

Fig. 3.3	Translation of a sphere from the origin to the point P_t	41
Fig. 3.4	Horizon of an observer on a beach	43
Fig. 3.5	Calculation of the intersection circle (horizon)	44
Fig. 4.1	The blades of CGA and the geometric meaning of some of them	46
Fig. 4.2	Complex numbers in CGA. The imaginary unit can be identified in CGA as one of the 2-blades $e_1 \wedge e_2$, $e_1 \wedge e_3$, and $e_2 \wedge e_3$, each with a different geometric meaning	47
Fig. 4.3	Definition of quaternions in Maple	49
Fig. 4.4	Quaternions in CGA	49
Fig. 4.5	Computation of the representation of a line in Maple	50
Fig. 4.6	The product of pure quaternions in Maple	51
Fig. 4.7	Plücker coordinates in CGA	53
Fig. 4.8	Computations of Plücker coordinates in Maple.....	54
Fig. 4.9	The six Plücker coordinates of the line through a and b consist of the coordinates of the direction vector \mathbf{u} and the moment vector \mathbf{m}	55
Fig. 4.10	Dual numbers in CGA	56
Fig. 4.11	Dual quaternions are represented in CGA based on eight blades: the scalar, six 2-blades and one 4-blade	57
Fig. 4.12	Dual-quaternion computations in Maple	58
Fig. 4.13	Dual quaternion computations in Maple	58
Fig. 5.1	The point at infinity	63
Fig. 5.2	A sphere with a <i>center</i> \mathbf{s} (in the direction opposite to a normal vector \mathbf{n}) that goes to infinity (while the radius of the sphere changes accordingly), results finally in a plane with a normal vector \mathbf{n} and a distance d from the origin	64
Fig. 5.3	The inner product $P \cdot S$ of a point and a sphere on the one hand already describes the square of a distance, but on the other hand has to be squared again in the least-squares method, since the inner product can be positive or negative depending on whether (a) the point \mathbf{p} lies outside the sphere or (b) the point \mathbf{p} lies inside the sphere ..	67
Fig. 5.4	The constraint $s^T s = 1$ leads implicitly to a scaling of the distance measure such that it gets smaller with increasing radius; if the radius increases from the one in (a) via the radius in (b) and further to an infinite radius, the distance measure gets zero for a plane considered as a sphere of infinite radius	67
Fig. 5.5	Fitting a sphere to a set of five points	68
Fig. 5.6	Fitting a plane to a set of five points.....	68

Fig. 6.1	Interactive and visual development of algorithms using CLUCalc. In the editor window, the intersection of two spheres is defined, which is immediately visualized in the visualization window	72
Fig. 6.2	BasisElementsE3.clu	73
Fig. 6.3	bivectorE3.clu	75
Fig. 6.4	trivectorE3.clu	76
Fig. 6.5	innerProductE3.clu	77
Fig. 6.6	DualE3.clu	81
Fig. 6.7	ProjectE3.clu	82
Fig. 6.8	ReflectE3.clu	83
Fig. 6.9	Rotor2d.clu	84
Fig. 6.10	Rotate_EXP_E3.clu	85
Fig. 6.11	Rotor3d.clu	86
Fig. 6.12	OneSphereN3.clu	88
Fig. 6.13	PlaneN3.clu	89
Fig. 6.14	CircleN3.clu	90
Fig. 6.15	LineN3.clu	91
Fig. 6.16	DualSphereN3.clu	92
Fig. 6.17	PointInsideCircleN3.clu	93
Fig. 6.18	intersectSphereSphereN3.clu	94
Fig. 6.19	intersectSphereLineN3.clu	94
Fig. 6.20	intersectPlaneLineN3.clu	95
Fig. 6.21	ReflectN3.clu	96
Fig. 6.22	ProjectN3.clu	97
Fig. 6.23	Visualization of the horizon example	98
Fig. 6.24	Visualization of a CLUScript describing motion	99
Fig. 7.1	Kinematic chain of the example robot	102
Fig. 7.2	Target point and gripper plane	102
Fig. 7.3	Computation of P_0	103
Fig. 7.4	Computation of P_2	103
Fig. 7.5	Computation of P_1	104
Fig. 7.6	Visualization of step 4	105
Fig. 8.1	The robot Geometer grasping an object	108
Fig. 8.2	Assigning points for the bounding cylinder of the object to be grasped	108
Fig. 8.3	Grasping circle Z_t	109
Fig. 8.4	Gripper	109
Fig. 8.5	Gripper circle Z_h , grasping circle Z_t and their axes L_h and L_t	110
Fig. 8.6	Moving the gripper circle Z_h towards the grasping circle Z_t	111
Fig. 8.7	CLUCalc code of the grasping algorithm	112
Fig. 8.8	Input parameters of the CLUCalc algorithm	112

Fig. 8.9	Construction and visualization of the base circle z_b	112
Fig. 8.10	Base points	112
Fig. 8.11	Base circle	113
Fig. 8.12	Translation vector	113
Fig. 8.13	Target circle	113
Fig. 8.14	Gripper circle	114
Fig. 8.15	Final position	115
Fig. 9.1	Rotation based on the midline between two points through the origin	122
Fig. 9.2	Swivel plane	124
Fig. 9.3	Computation of the elbow point	125
Fig. 9.4	Using the elbow quaternion	126
Fig. 9.5	Rotation to the elbow position	127
Fig. 9.6	Rotation to the wrist location	128
Fig. 9.7	Computation of the elbow point	131
Fig. 10.1	The CLUCalc input code of the <i>horizon</i> example as requested by <i>Gaalop</i>	142
Fig. 10.2	Two alternative calculations of the horizon application according to Sect. 3.7, with the same optimized result	144
Fig. 10.3	The Geometric Algebra Computing architecture. Algorithms are compiled to an <i>intermediate</i> <i>representation</i> for compilation to different computing platforms..	145
Fig. 11.1	Gaalop GPC for C++	156
Fig. 11.2	Visualization of the horizon example	157
Fig. 11.3	Point–triangle intersection in CLUCalc. This picture shows the triangle, the plane it is embedded in, and its three boundary planes	159
Fig. 12.1	Gaalop GPC for OpenCL	162
Fig. 13.1	Screenshot of a molecular dynamics simulation using CGA	166
Fig. 13.2	The forces between all of the atoms in the molecules result in movement of the molecules	166
Fig. 13.3	The Lennard-Jones potential describes the dependence of the energy between two atoms and the distance between them (Image source: www.wikipedia.org)	166
Fig. 13.4	Code architecture of the molecular dynamics application	167
Fig. 14.1	Generation of optimized FPGA implementations from Geometric Algebra algorithms	180
Fig. 14.2	Pipeline schedule for the coefficient p_{ex} of a multivector. All of the computations specified by (14.1) for all of the pipeline stages can be done in parallel	180

Fig. 14.3	Parallel dot product of two n -dimensional vectors Vector0 and Vector1 (n parallel products followed by $\log(n)$ parallel addition steps)	182
Fig. 14.4	Selected blades of a multivector are stored in a five-dimensional vector in the form $E_0, -E_2, E_3, E_5,$ and $-E_4$	184
Fig. 14.5	Geometric Algebra computers based on Gaalop are able to use GAPP as instruction set for Geometric Algebra computations	188

List of Tables

Table 1.1	Multiplication table of 2D Geometric Algebra. This algebra consists of basic algebraic objects of grade (dimension) 0, the scalar, of grade 1, the two basis vectors e_1 and e_2 and of grade 2, the bivector $e_1 \wedge e_2$, which can be identified with the imaginary number i squaring to -1	2
Table 1.2	List of the basic geometric primitives provided by 5D Conformal Geometric Algebra. The bold characters represent 3D entities (\mathbf{x} is a 3D point, \mathbf{n} is a 3D normal vector, and \mathbf{x}^2 is a scalar product of the 3D vector \mathbf{x}). The two additional basis vectors e_0 and e_∞ represent the origin and infinity. Based on the outer product, circles and lines can be described as intersections of two spheres and of two planes, respectively. The parameter r represents the radius of the sphere, and the parameter d the distance from the origin to the plane	4
Table 2.1	List of the 8 blades of 3D Euclidean Geometric Algebra	18
Table 2.2	Notations for the geometric algebra products	19
Table 2.3	Properties of the outer product \wedge	19
Table 2.4	The 8 blades of 3D Euclidean Geometric Algebra.....	23
Table 2.5	The 16 blades of 4D projective Geometric Algebra	25
Table 3.1	The 32 blades of the 5D Conformal Geometric Algebra (CGA) (for simplicity reasons we use e_0 and e_∞ of (3.2) for this blade description while mathematically correct the basis vectors e_+ and e_- should be used)	28

Table 3.2	The two representations (IPNS and OPNS) of conformal geometric entities. The IPNS and OPNS representations are dual to each other, which is indicated by the asterisk symbol	29
Table 3.3	Geometric meaning of conformal vectors	33
Table 3.4	Geometric meaning of the inner product of two conformal vectors U and V	34
Table 4.1	Notation for Geometric Algebra operations in Maple	46
Table 6.1	List of the eight blades of 3D Euclidean Geometric Algebra	74
Table 6.2	Properties of the outer product	75
Table 6.3	Meanings of the coefficients of the two additional coordinates in CGA	87
Table 9.1	Input/output parameters of the inverse kinematics algorithm	124
Table 9.2	Input/output parameters of the inverse kinematics algorithm using Gaigen 2	130
Table 9.3	Computation of the shoulder quaternion	133
Table 9.4	Input/output parameters of the inverse kinematics algorithm using Maple	134
Table 10.1	The 32 blades of 5D CGA that compose a multivector. The entry in the first column is the index of the corresponding blade. The negated entries are needed for the selectors of the Geometric Algebra Parallelism Programs (GAPP) described in Chap. 14.....	143
Table 10.2	Multiplication table for the geometric product of 2D Geometric Algebra. This algebra consists of the following basic algebraic objects: an object of grade (dimension) 0, the scalar; objects of grade 1, the two basis vectors e_1 and e_2 ; and an object of grade 2, the bivector $e_1 \wedge e_2$	146
Table 10.3	Multiplication table of 2D Geometric Algebra in terms of its basis blades E_1 , E_2 , E_3 and E_4	146

Table 10.4	Multiplication table describing the geometric product of two multivectors $a = \sum a_i E_i$ and $b = \sum b_i E_i$ for 3D Euclidean Geometric Algebra. Each coefficient c_k of the product $c = ab$ can be computed by summing the products $\pm a_i * b_j$ based on the table entries for E_k ; for instance, $c_0 = a_0 * b_0 + a_1 * b_1 + a_2 * b_2 + a_3 * b_3 - a_4 * b_4 - a_5 * b_5 - a_6 * b_6 - a_7 * b_7$ for the table entries for E_0 . In other words, a particular blade E_k of the result multivector c is computed by summing the products $\pm a_i * b_j$ of the table entries marked by E_k . See Sect. 10.3.2 for details of the algorithmic steps for computation with multiplication tables	149
Table 10.5	Multiplication table describing the geometric product of two vectors $a = a_1 e_1 + a_2 e_2 + a_3 e_3$ and $b = b_1 e_1 + b_2 e_2 + b_3 e_3$ for 3D Euclidean Geometric Algebra. Note that all rows and columns for basis blades not needed for the vectors are set to zero	150
Table 10.6	Multiplication table describing the outer product of two general multivectors $a = \sum a_i E_i$ and $b = \sum b_i E_i$ for 3D Euclidean Geometric Algebra	151
Table 10.7	Part of the multiplication table in Table 10.6 describing the outer product of two specific multivectors $a = \sum_{i=1}^3 a_i E_i$ and $d = \sum_{i=0}^6 d_i E_i$ for 3D Euclidean Geometric Algebra	151
Table 10.8	A subset of the 5D geometric-product multiplication table of Geometric Algebra. This lists only 5 out of the 32 possible blades for each multivector. The expression E denotes the outer product $E = e_\infty \wedge e_0$	153
Table 12.1	Gaalop GPC functions for constructing and accessing multivectors	164
Table 14.1	The main commands of the Geometric Algebra Parallelism Programs (GAPP) language; a more detailed list can be found in [105]	183