

Geometric Algebra for Computer Science

Geometric Algebra for Computer Science

An Object-oriented Approach to Geometry

LEO DORST

DANIEL FONTIJNE

STEPHEN MANN



ELSEVIER

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

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

© 2007 by Elsevier Inc. All rights reserved.

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances in which Morgan Kaufmann Publishers is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, scanning, or otherwise—without prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone: (+44) 1865 843830, fax: (+44) 1865 853333, E-mail: permissions@elsevier.com. You may also complete your request on-line via the Elsevier homepage (<http://elsevier.com>), by selecting "Support & Contact" then "Copyright and Permission" and then "Obtaining Permissions."

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

Working together to grow
libraries in developing countries

www.elsevier.com | www.bookaid.org | www.sabre.org

ELSEVIER

BOOK AID
International

Sabre Foundation

Contents

LIST OF FIGURES	xx
LIST OF TABLES	xxvi
LIST OF PROGRAMMING EXAMPLES	xxviii
PREFACE	xxx
CHAPTER 1. WHY GEOMETRIC ALGEBRA?	1
1.1 An Example in Geometric Algebra	1
1.2 How It Works and How It's Different	7
1.2.1 Vector Spaces as Modeling Tools	8
1.2.2 Subspaces as Elements of Computation	9
1.2.3 Linear Transformations Extended	12
1.2.4 Universal Orthogonal Transformations	12
1.2.5 Objects are Operators	14
1.2.6 Closed-Form Interpolation and Perturbation	15
1.3 Programming Geometry	15
1.3.1 You Can Only Gain	15
1.3.2 Software Implementation	16
1.4 The Structure of This Book	17
1.4.1 Part I: Geometric Algebra	18
1.4.2 Part II: Models of Geometry	18
1.4.3 Part III: Implementation of Geometric Algebra	18
1.5 The Structure of the Chapters	19
PART I GEOMETRIC ALGEBRA	
<hr/>	
CHAPTER 2. SPANNING ORIENTED SUBSPACES	23
2.1 Vector Spaces	24

2.2	Oriented Line Elements	25
2.2.1	Properties of Homogeneous Lines	25
2.2.2	Visualizing Vectors	26
2.3	Oriented Area Elements	27
2.3.1	Properties of Planes	27
2.3.2	Introducing the Outer Product	28
2.3.3	Visualizing Bivectors	31
2.3.4	Visualizing Bivector Addition	32
2.4	Oriented Volume Elements	33
2.4.1	Properties of Volumes	33
2.4.2	Associativity of the Outer Product	35
2.4.3	Visualization of Trivectors	36
2.5	Quadvectors in 3-D Are Zero	37
2.6	Scalars Interpreted Geometrically	37
2.7	Applications	39
2.7.1	Solving Linear Equations	39
2.7.2	Intersecting Planar Lines	41
2.8	Homogeneous Subspace Representation	42
2.8.1	Parallelness	42
2.8.2	Direct Representation of Oriented Weighted Subspaces	43
2.8.3	Nonmetric Lengths, Areas, and Volumes	43
2.9	The Graded Algebra of Subspaces	44
2.9.1	Blades and Grades	44
2.9.2	The Ladder of Subspaces	45
2.9.3	k -Blades Versus k -Vectors	46
2.9.4	The Grassmann Algebra of Multivectors	47
2.9.5	Reversion and Grade Involution	49
2.10	Summary of Outer Product Properties	50
2.11	Further Reading	51
2.12	Exercises	51
2.12.1	Drills	51
2.12.2	Structural Exercises	52
2.13	Programming Examples and Exercises	53
2.13.1	Drawing Bivectors	57
2.13.2	Exercise: Hidden Surface Removal	57
2.13.3	Singularities in Vector Fields	60
CHAPTER 3. METRIC PRODUCTS OF SUBSPACES		65
3.1	Sizing Up Subspaces	66
3.1.1	Metrics, Norms, and Angles	66
3.1.2	Definition of the Scalar Product *	67

3.1.3	The Squared Norm of a Subspace	67
3.1.4	The Angle Between Subspaces	68
3.2	From Scalar Product to Contraction	71
3.2.1	Implicit Definition of Contraction \lrcorner	71
3.2.2	Computing the Contraction Explicitly	73
3.2.3	Algebraic Subtleties	74
3.3	Geometric Interpretation of the Contraction	75
3.4	The Other Contraction \llcorner	77
3.5	Orthogonality and Duality	78
3.5.1	Nonassociativity of the Contraction	78
3.5.2	The Inverse of a Blade	79
3.5.3	Orthogonal Complement and Duality	80
3.5.4	The Duality Relationships	82
3.5.5	Dual Representation of Subspaces	83
3.6	Orthogonal Projection of Subspaces	83
3.7	The 3-D Cross Product	86
3.7.1	Uses of the Cross Product	86
3.7.2	The Cross Product Incorporated	87
3.8	Application: Reciprocal Frames	89
3.9	Further Reading	91
3.10	Exercises	91
3.10.1	Drills	91
3.10.2	Structural Exercises	91
3.11	Programming Examples and Exercises	93
3.11.1	Orthonormalization	93
3.11.2	Exercise: Implementing the Cross Product	94
3.11.3	Reciprocal Frames	95
3.11.4	Color Space Conversion	95
CHAPTER 4. LINEAR TRANSFORMATIONS OF SUBSPACES		99
4.1	Linear Transformations of Vectors	100
4.2	Outermorphisms: Linear Transformations of Blades	101
4.2.1	Motivation of the Outermorphism	102
4.2.2	Examples of Outermorphisms	103
4.2.3	The Determinant of a Linear Transformation	106
4.3	Linear Transformation of the Metric Products	108
4.3.1	Linear Transformation of the Scalar Product	108
4.3.2	The Adjoint of a Linear Transformation	108
4.3.3	Linear Transformation of the Contraction	109
4.3.4	Orthogonal Transformations	110

4.3.5	Transforming a Dual Representation	111
4.3.6	Application: Linear Transformation of the Cross Product	112
4.4	Inverses of Outermorphisms	113
4.5	Matrix Representations	114
4.5.1	Matrices for Vector Transformations	114
4.5.2	Matrices for Outermorphisms	115
4.6	Summary	117
4.7	Suggestions for Further Reading	117
4.8	Structural Exercises	118
4.9	Programming Examples and Exercises	120
4.9.1	Orthogonal Projection	120
4.9.2	Orthogonal Projection, Matrix Representation	120
4.9.3	Transforming Normal Vectors	122
CHAPTER 5. INTERSECTION AND UNION OF SUBSPACES		125
5.1	The Phenomenology of Intersection	125
5.2	Intersection through Outer Factorization	127
5.3	Relationships Between Meet and Join	128
5.4	Using Meet and Join	129
5.5	Join and Meet are Mostly Linear	131
5.6	Quantitative Properties of the Meet	132
5.7	Linear Transformation of Meet and Join	133
5.8	Offset Subspaces	136
5.9	Further Reading	136
5.10	Exercises	137
5.10.1	Drills	137
5.10.2	Structural Exercises	137
5.11	Programming Examples and Exercises	138
5.11.1	The Meet and Join	138
5.11.2	Efficiency	140
5.11.3	Floating Point Issues	140
CHAPTER 6. THE FUNDAMENTAL PRODUCT OF GEOMETRIC ALGEBRA		141
6.1	The Geometric Product for Vectors	142
6.1.1	An Invertible Product for Geometry	142
6.1.2	Symmetry and Antisymmetry	143
6.1.3	Properties of the Geometric Product	143
6.1.4	The Geometric Product for Vectors on a Basis	144

	6.1.5	Dividing by a Vector	145
	6.1.6	Ratios of Vectors as Operators	146
6.2		The Geometric Product of Multivectors	147
	6.2.1	Algebraic Definition of the Geometric Product	148
	6.2.2	Evaluating the Geometric Product	149
	6.2.3	Grades and the Geometric Product	150
6.3		The Subspace Products Retrieved	151
	6.3.1	The Subspace Products from Symmetry	152
	6.3.2	The Subspace Products as Selected Grades	154
6.4		Geometric Division	154
	6.4.1	Inverses of Blades	155
	6.4.2	Decomposition: Projection to Subspaces	155
	6.4.3	The Other Division: Reflection	158
6.5		Further Reading	159
6.6		Exercises	159
	6.6.1	Drills	159
	6.6.2	Structural Exercises	160
6.7		Programming Examples and Exercises	161
	6.7.1	Exercise: Subspace Products Retrieved	161
	6.7.2	Gram-Schmidt Orthogonalization	162
CHAPTER 7. ORTHOGONAL TRANSFORMATIONS AS VERSORS			167
7.1		Reflections of Subspaces	168
7.2		Rotations of Subspaces	169
	7.2.1	3-D Rotors as Double Reflectors	170
	7.2.2	Rotors Perform Rotations	172
	7.2.3	A Sense of Rotation	174
7.3		Composition of Rotations	176
	7.3.1	Multiple Rotations in 2-D	177
	7.3.2	Real 2-D Rotors Subsume Complex Numbers	177
	7.3.3	Multiple Rotations in 3-D	179
	7.3.4	Visualizing 3-D Rotations	179
	7.3.5	Unit Quaternions Subsumed	181
7.4		The Exponential Representation of Rotors	182
	7.4.1	Pure Rotors as Exponentials of 2-Blades	183
	7.4.2	Trigonometric and Hyperbolic Functions	184
	7.4.3	Rotors as Exponentials of Bivectors	185
	7.4.4	Logarithms	187
7.5		Subspaces as Operators	188
	7.5.1	Reflection by Subspaces	188
	7.5.2	Subspace Projection as Sandwiching	190
	7.5.3	Transformations as Objects	190

7.6	Versors Generate Orthogonal Transformations	191
7.6.1	The Versor Product	192
7.6.2	Even and Odd Versors	193
7.6.3	Orthogonal Transformations are Versor Products	193
7.6.4	Versors, Blades, Rotors, and Spinors	195
7.7	The Product Structure of Geometric Algebra	196
7.7.1	The Products Summarized	196
7.7.2	Geometric Algebra versus Clifford Algebra	198
7.7.3	But—is it Efficient?	199
7.8	Further Reading	201
7.9	Exercises	202
7.9.1	Drills	202
7.9.2	Structural Exercises	202
7.10	Programming Examples and Exercises	204
7.10.1	Reflecting in Vectors	204
7.10.2	Two Reflections Equal One Rotation	204
7.10.3	Matrix-Rotor Conversion 1	204
7.10.4	Exercise: Matrix-Rotor Conversion 2	206
7.10.5	Julia Fractals	208
7.10.6	Extra Example: Rotations Used for Example User Interface	210
CHAPTER 8. GEOMETRIC DIFFERENTIATION		213
8.1	Geometrical Changes by Orthogonal Transformations	214
8.2	Transformational Changes	215
8.2.1	The Commutator Product	215
8.2.2	Rotor-Induced Changes	217
8.2.3	Multiple Rotor-Induced Changes	218
8.2.4	Transformation of a Change	219
8.2.5	Change of a Transformation	220
8.3	Parametric Differentiation	221
8.4	Scalar Differentiation	221
8.4.1	Application: Radius of Curvature of a Planar Curve	223
8.5	Directional Differentiation	224
8.5.1	Table of Elementary Results	225
8.5.2	Application: Tilting a Mirror	228
8.6	Vector Differentiation	230
8.6.1	Elementary Results of Vector Differentiation	232
8.6.2	Properties of Vector Differentiation	234
8.7	Multivector Differentiation	235
8.7.1	Definition	236
8.7.2	Application: Estimating Rotors Optimally	236
8.8	Further Reading	239

8.9	Exercises	240
	8.9.1 Drills	240
	8.9.2 Structural Exercises	240

PART II MODELS OF GEOMETRIES

CHAPTER 9. MODELING GEOMETRIES 245

CHAPTER 10. THE VECTOR SPACE MODEL: THE ALGEBRA OF DIRECTIONS 247

10.1	The Natural Model for Directions	248
10.2	Angular Relationships	248
	10.2.1 The Geometry of Planar Triangles	249
	10.2.2 Angular Relationships in 3-D	251
	10.2.3 Rotation Groups and Crystallography	254
10.3	Computing with 3-D Rotors	256
	10.3.1 Determining a Rotor from Rotation Plane and Angle	256
	10.3.2 Determining a Rotor from a Frame Rotation in 3-D	257
	10.3.3 The Logarithm of a 3-D Rotor	258
	10.3.4 Rotation Interpolation	259
10.4	Application: Estimation in the Vector Space Model	260
	10.4.1 Noisy Rotor Estimation	260
	10.4.2 External Camera Calibration	260
10.5	Convenient Abuse: Locations as Directions	263
10.6	Further Reading	264
10.7	Programming Examples and Exercises	265
	10.7.1 Interpolating Rotations	265
	10.7.2 Crystallography Implementation	267
	10.7.3 External Camera Calibration	268

CHAPTER 11. THE HOMOGENEOUS MODEL 271

11.1	Homogeneous Representation Space	272
11.2	All Points Are Vectors	274
	11.2.1 Finite Points	274
	11.2.2 Infinite Points and Attitudes	275
	11.2.3 Addition of Points	276
	11.2.4 Terminology: from Precise to Convenient	278
11.3	All Lines Are 2-Blades	278
	11.3.1 Finite Lines	278
	11.3.2 Lines at Infinity	282
	11.3.3 Don't Add Lines	282

11.4	All Planes Are 3-Blades	283
11.5	k-Flats as $(k + 1)$-Blades	285
11.5.1	Finite k -Flats	285
11.5.2	Infinite k -Flats	285
11.5.3	Parameters of k -Flats	285
11.5.4	The Number of Parameters of an Offset Flat	286
11.6	Direct and Dual Representations of Flats	286
11.6.1	Direct Representation	286
11.6.2	Dual Representation	288
11.7	Incidence Relationships	292
11.7.1	Examples of Incidence Computations	292
11.7.2	Relative Orientation	296
11.7.3	Relative Lengths: Distance Ratio and Cross Ratio	298
11.8	Linear Transformations: Motions, and More	302
11.8.1	Linear Transformations on Blades	302
11.8.2	Translations	303
11.8.3	Rotation Around the Origin	304
11.8.4	General Rotation	305
11.8.5	Rigid Body Motion	305
11.8.6	Constructing Elements Through Motions	305
11.8.7	Rigid Body Motion Outermorphisms as Matrices	306
11.8.8	Affine Transformations	306
11.8.9	Projective Transformations	308
11.9	Coordinate-Free Parameterized Constructions	309
11.10	Metric Products in the Homogeneous Model	312
11.10.1	Non-Euclidean Results	312
11.10.2	Nonmetric Orthogonal Projection	314
11.11	Further Reading	315
11.12	Exercises	316
11.12.1	Drills	316
11.12.2	Structural Exercises	316
11.13	Programming Examples and Exercises	320
11.13.1	Working with Points	320
11.13.2	Intersecting Primitives	322
11.13.3	Don't Add Lines	324
11.13.4	Perspective Projection	325
CHAPTER 12. APPLICATIONS OF THE HOMOGENEOUS MODEL		327
12.1	Homogeneous Plücker Coordinates in 3-D	328
12.1.1	Line Representation	328
12.1.2	The Elements in Coordinate Form	330
12.1.3	Combining Elements	332

	12.1.4	Matrices of Motions in Plücker Coordinates	334
	12.1.5	Sparse Usage of the 2^4 Dimensions	336
12.2		Imaging by Multiple Cameras	336
	12.2.1	The Pinhole Camera	337
	12.2.2	Homogeneous Coordinates as Imaging	339
	12.2.3	Cameras and Stereo Vision	340
	12.2.4	Line-based Stereo Vision	342
12.3		Further Reading	346
12.4		Exercises	347
	12.4.1	Structural Exercises	347
12.5		Programming Examples and Exercises	348
	12.5.1	Loading Transformations into OpenGL	348
	12.5.2	Transforming Primitives with OpenGL Matrices	349
	12.5.3	Marker Reconstruction in Optical Motion Capture	351
		CHAPTER 13. THE CONFORMAL MODEL: OPERATIONAL EUCLIDEAN GEOMETRY	355
13.1		The Conformal Model	356
	13.1.1	Representational Space and Metric	356
	13.1.2	Points as Null Vectors	359
	13.1.3	General Vectors Represent Dual Planes and Spheres	361
13.2		Euclidean Transformations as Versors	364
	13.2.1	Euclidean Versors	364
	13.2.2	Proper Euclidean Motions as Even Versors	365
	13.2.3	Covariant Preservation of Structure	367
	13.2.4	The Invariance of Properties	369
13.3		Flats and Directions	370
	13.3.1	The Direct Representation of Flats	370
	13.3.2	Correspondence with the Homogeneous Model	372
	13.3.3	Dual Representation of Flats	374
	13.3.4	Directions	376
13.4		Application: General Planar Reflection	377
13.5		Rigid Body Motions	379
	13.5.1	Algebraic Properties of Translations and Rotations	380
	13.5.2	Screw Motions	381
	13.5.3	Logarithm of a Rigid Body Motion	383
13.6		Application: Interpolation of Rigid Body Motions	385
13.7		Application: Differential Planar Reflections	386
13.8		Further Reading	388
13.9		Exercises	388
	13.9.1	Drills	388
	13.9.2	Structural Exercises	389

13.10	Programming Examples and Exercises	390
13.10.1	Metric Matters	390
13.10.2	Exercise: The Distance Between Points	392
13.10.3	Loading Transformations into OpenGL, Again	393
13.10.4	Interpolation of Rigid Body Motions	395
CHAPTER 14. NEW PRIMITIVES FOR EUCLIDEAN GEOMETRY		397
14.1	Rounds	398
14.1.1	Dual Rounds	398
14.1.2	Direct Rounds	400
14.1.3	Oriented Rounds	402
14.2	Tangents as Intersections of Touching Rounds	404
14.2.1	Euclid's Elements	406
14.2.2	From Blades to Parameters	408
14.3	A Visual Explanation of Rounds as Blades	410
14.3.1	Point Representation	410
14.3.2	Circle Representation	412
14.3.3	Euclidean Circles Intersect as Planes	414
14.4	Application: Voronoi Diagrams	415
14.5	Application: Fitting a Sphere to Points	417
14.5.1	The Inner Product Distance of Spheres	417
14.5.2	Fitting a Sphere to Points	419
14.6	Application: Kinematics	420
14.6.1	Forward Kinematics	420
14.6.2	Inverse Kinematics	423
14.7	Further Reading	426
14.8	Exercises	427
14.8.1	Drills	427
14.8.2	Structural Exercises	427
14.9	Programming Examples and Exercises	428
14.9.1	Voronoi Diagrams and Delaunay Triangulations	428
14.9.2	Exercise: Drawing Euclid's Elements	431
14.9.3	Conformal Primitives and Intersections	432
14.9.4	Fitting a Sphere to a Set of Points	434
CHAPTER 15. CONSTRUCTIONS IN EUCLIDEAN GEOMETRY		437
15.1	Euclidean Incidence and Coincidence	438
15.1.1	Incidence Revisited	438
15.1.2	Co-Incidence	438

	15.1.3	Real Meet or Plunge	440
	15.1.4	The Plunge of Flats	442
15.2		Euclidean Nuggets	444
	15.2.1	Tangents Without Differentiating	445
	15.2.2	Carriers, Tangent Flat	445
	15.2.3	Surrounds, Factorization of Rounds	446
	15.2.4	Affine Combinations	447
15.3		Euclidean Projections	449
15.4		Application: All Kinds of Vectors	451
15.5		Application: Analysis of a Voronoi Cell	455
	15.5.1	Edge Lines	455
	15.5.2	Edge Point	456
	15.5.3	Edge Length	457
	15.5.4	Conversion to Classical Formulas	458
15.6		Further Reading	460
15.7		Exercises	460
	15.7.1	Drills	460
	15.7.2	Structural Exercises	461
15.8		Programming Examples and Exercises	462
	15.8.1	The Plunge	462
	15.8.2	Affine Combinations of Points	463
	15.8.3	Euclidean Projections	464
		CHAPTER 16. CONFORMAL OPERATORS	465
16.1		Spherical Inversion	465
16.2		Applications of Inversion	468
	16.2.1	The Center of a Round	468
	16.2.2	Reflection in Spheres and Circles	468
16.3		Scaling	469
	16.3.1	The Positive Scaling Rotor	469
	16.3.2	Reflection in the Origin: Negative Scaling	471
	16.3.3	Positively Scaled Rigid Body Motions	472
	16.3.4	Logarithm of a Scaled Rigid Body Motion	473
16.4		Transversions	475
16.5		Transformations of the Standard Blades	477
16.6		General Conformal Transformations	477
	16.6.1	Loxodromes	477
	16.6.2	Circular Rotations	479
	16.6.3	Möbius Transformations	479
16.7		Non-Euclidean Geometries	480
	16.7.1	Hyperbolic Geometry	480
	16.7.2	Spherical Geometry	482

16.8	Further Reading	483
16.9	Exercises	483
	16.9.1 Drills	483
	16.9.2 Structural Exercises	484
16.10	Programming Examples and Exercises	486
	16.10.1 Homogeneous 4×4 Matrices to Conformal Versors	486
	16.10.2 Logarithm of Scaled Rigid Body Motion	493
	16.10.3 Interpolation of Scaled Rigid Body Motions	493
	16.10.4 The Seashell	494
CHAPTER 17. OPERATIONAL MODELS FOR GEOMETRIES		497
17.1	Algebras for Geometries	497
PART III IMPLEMENTING GEOMETRIC ALGEBRA		
CHAPTER 18. IMPLEMENTATION ISSUES		503
18.1	The Levels of Geometric Algebra Implementation	504
18.2	Who Should Read What	506
18.3	Alternative Implementation Approaches	506
	18.3.1 Isomorphic Matrix Algebras	506
	18.3.2 Irreducible Matrix Implementations	507
	18.3.3 Factored Representations	508
18.4	Structural Exercises	509
CHAPTER 19. BASIS BLADES AND OPERATIONS		511
19.1	Representing Unit Basis Blades with Bitmaps	512
19.2	The Outer Product of Basis Blades	513
19.3	The Geometric Product of Basis Blades in an Orthogonal Metric	515
19.4	The Geometric Product of Basis Blades in Nonorthogonal Metrics	516
19.5	The Inner Products of Basis Blades	518
19.6	Commutator Product of Basis Blades	518
19.7	Grade-Dependent Signs on Basis Blades	518

CHAPTER 20. THE LINEAR PRODUCTS AND OPERATIONS	521
20.1 A Linear Algebra Approach	522
20.1.1 Implementing the Linear Operations	522
20.1.2 Implementing the Linear Products	523
20.2 The List of Basis Blades Approach	526
20.3 Structural Exercises	527
CHAPTER 21. FUNDAMENTAL ALGORITHMS FOR NONLINEAR PRODUCTS	529
21.1 Inverse of Versors (and Blades)	529
21.2 Inverse of Multivectors	530
21.3 Exponential, Sine, and Cosine of Multivectors	531
21.4 Logarithm of Versors	532
21.5 Multivector Classification	532
21.6 Blade Factorization	533
21.7 The Meet and Join of Blades	536
21.8 Structural Exercises	540
CHAPTER 22. SPECIALIZING THE STRUCTURE FOR EFFICIENCY	541
22.1 Issues in Efficient Implementation	541
22.2 Generative Programming	543
22.3 Resolving the Issues	544
22.3.1 The Approach	545
22.4 Implementation	546
22.4.1 Algebra Specification	546
22.4.2 Implementation of the General Multivector Class	547
22.4.3 Implementation of the Specialized Multivector Classes	549
22.4.4 Optimizing Functions Over the Algebra	550
22.4.5 Outermorphisms	552
22.4.6 Optimizing the Nonlinear Functions	553
22.5 Benchmarks	554
22.6 A Small Price to Pay	556
22.7 Exercises	556

CHAPTER 23. USING THE GEOMETRY IN A RAY-TRACING APPLICATION	557
23.1 Ray-Tracing Basics	558
23.2 The Ray-Tracing Algorithm	559
23.3 Representing Meshes	560
23.4 Modeling the Scene	566
23.4.1 Scene Transformations	566
23.5 Tracing the Rays	573
23.5.1 The Representation of Rays	573
23.5.2 Spawning Rays	575
23.5.3 Ray-Model Intersection	577
23.5.4 Reflection	579
23.5.5 Refraction	580
23.6 Shading	580
23.7 Evaluation	581

PART IV APPENDICES

A METRICS AND NULL VECTORS	585
A.1 The Bilinear Form	585
A.2 Diagonalization to Orthonormal Basis	586
A.3 General Metrics	586
A.4 Null Vectors and Null Blades	587
A.5 Rotors in General Metrics	587
B CONTRACTIONS AND OTHER INNER PRODUCTS	589
B.1 Other Inner Products	589
B.1.1 The Dot Product	589
B.1.2 Hestenes' Inner Product	590
B.1.3 Near Equivalence of Inner Products	590
B.1.4 Geometric Interpretation and Usage	591
B.2 Equivalence of the Implicit and Explicit Contraction Definitions	591
B.3 Proof of the Second Duality	594
B.4 Projection and the Norm of the Contraction	595

C	SUBSPACE PRODUCTS RETRIEVED	597
C.1	Outer Product from Peometric Product	597
C.2	Contractions from Geometric Product	598
C.3	Proof of the Grade Approach	599
D	COMMON EQUATIONS	603
	BIBLIOGRAPHY	609
	INDEX	613

List of Figures

1.1	Example of the use of geometric algebra	2
1.2	Code to generate Figure 1.1	5
1.3	Example of the use of geometric algebra	6
1.4	The outer product and its interpretations	11
2.1	Spanning homogeneous subspaces in a 3-D vector space	25
2.2	Imagining vector addition	27
2.3	Bivector representations	32
2.4	Imagining bivector addition in 2-D space	33
2.5	Bivector addition in 3-D space	34
2.6	The associativity of the outer product	35
2.7	Solving linear equations with bivectors	40
2.8	Intersecting lines in the plane	41
2.9	Code for drawing bivectors	58
2.10	Drawing bivectors screenshot (Example 1)	59
2.11	The orientation of front- and back-facing polygons	59
2.12	A wire-frame torus with and without backface culling	60
2.13	The code that renders a model from its 2-D vertices (Exercise 2)	61
2.14	Sampling a vector field and summing trivectors	62
2.15	Code to test for singularity (Example 3)	63
2.16	A helix-shaped singularity, as detected by Example 3	64
3.1	Computing the scalar product of 2-blades	70
3.2	From scalar product to contraction	72
3.3	The contraction of a vector onto a 2	76
3.4	Duality of vectors in 2-D	81

3.5	Duality of vectors and bivectors in 3-D	82
3.6	Projection onto a subspace	84
3.7	Three uses of the cross product	87
3.8	Duality and the cross product	89
3.9	Orthonormalization code (Example 1)	93
3.10	Orthonormalization	94
3.11	Reciprocal frame code	96
3.12	Color space conversion code (Example 4)	97
3.13	Color space conversion screenshot	98
4.1	The defining properties of a linear transformation	100
4.2	Projection onto a line \mathbf{a} in the \mathbf{b} -direction	104
4.3	A rotation around the origin of unit vectors in the plane	105
4.4	Projection of a vector onto a bivector	121
4.5	Matrix representation of projection code	122
4.6	Transforming normals vector	123
5.1	The ambiguity of the magnitude of the intersection of two planes	126
5.2	The <i>meet</i> of two oriented planes	130
5.3	A line meeting a plane in the origin	131
5.4	When the <i>join</i> of two (near-)parallel vectors becomes a 2-blade (Example 3)	140
6.1	Invertibility of the subspace products	142
6.2	Ratios of vectors	146
6.3	Projection and rejection of a vector	156
6.4	Reflecting a vector in a line	158
6.5	Gram-Schmidt orthogonalization	163
6.6	Gram-Schmidt orthogonalization code (Example 2)	164
7.1	Line and plane reflection	169
7.2	A rotation in a plane parallel to \mathbf{I} is two reflections in vectors in that plane	170
7.3	A rotor in action	171
7.4	Sense of rotation	175
7.5	The unique rotor-based rotations in the range $\phi = [0, 4\pi)$	176
7.6	(a) A spherical triangle. (b) Composition of rotations through concatenation of rotor arcs	180
7.7	A reflector in action	189
7.8	The rotor product in Euclidean spaces as a Taylor series	197

7.9	Interactive version of Figure 7.2	205
7.10	Rotation matrix to rotor conversion	207
7.11	2-D Julia fractal code	210
7.12	A 2-D Julia fractal, computed using the geometric product of real vectors	211
7.13	3-D Julia fractal	212
8.1	Directional differentiation of a vector inversion	227
8.2	Changes in reflection of a rotating mirror	229
8.3	The directional derivative of the spherical projection	241
10.1	A triangle $\mathbf{a} + \mathbf{b} + \mathbf{c} = 0$ in a directed plane \mathbf{I}	249
10.2	The angle between a vector and a bivector (see text)	252
10.3	A spherical triangle	253
10.4	Interpolation of rotations	259
10.5	Interpolation of rotations (Example 1)	266
10.6	Crystallography (Example 2)	267
10.7	External camera calibration (Example 3)	268
11.1	The extra dimension of the homogeneous representation space	274
11.2	Representing offset subspaces in \mathbb{R}^{n+1}	280
11.3	Defining offset subspaces fully in the base space	288
11.4	The dual hyperplane representation in \mathbb{R}^2 and \mathbb{R}^1	290
11.5	The intersection of two offset lines L and M to produce a point p	293
11.6	The <i>meet</i> of two skew lines	295
11.7	The relative orientation of oriented flats	296
11.8	The combinations of four points taken in the cross ratio	300
11.9	The combinations of four lines taken in the cross ratio	301
11.10	Conics in the homogeneous model	308
11.11	Finding a line through a point, perpendicular to a given line	310
11.12	The orthogonal projection in the homogeneous model (see text)	315
11.13	The beginning of a row of equidistant telegraph poles	319
11.14	Example 2 in action	323
11.15	Perspective projection (Example 4)	325
12.1	Plücker coordinates of a line in 3-D	329
12.2	A pinhole camera	337
12.3	The epipolar constraint	342
12.4	The plane of rays generated by a line observation L	343

12.5	The projection of the optical center onto all rays generates an eyeball	348
12.6	Reconstruction of motion capture data	351
12.7	Reconstruction of markers	352
12.8	Crossing lines code	354
13.1	Euclidean transformations as multiple reflections in planes	366
13.2	Flat elements in the conformal model	373
13.3	Planar reflection in the conformal model	377
13.4	Chasles' screw	382
13.5	Computation of the logarithm of a rigid body motion	384
13.6	Rigid body motion interpolation	385
13.7	Reflection in a rotating mirror	387
13.8	The output of the solution to Example 2	393
13.9	Example 4 in action: the interpolation of rigid body motions	394
14.1	Dual rounds in the conformal model	398
14.2	Intersection of two spheres of decreasing radii	405
14.3	Visualization of a 2-D Euclidean point on the representative paraboloid	411
14.4	The representation of a circle on the representative paraboloid	412
14.5	Cross section of the parabola of null vectors	413
14.6	Visualization of the intersection of circles on the representative paraboloid	414
14.7	A Voronoi diagram in the conformal model	416
14.8	Inner product as distance measure	418
14.9	Forward kinematics of a robot arm	421
14.10	Inverse kinematics of a robot arm	422
14.11	A Voronoi diagram of a set of points, as computed by Example 1	429
14.12	Euclid's elements (Example 2)	432
14.13	Example 3 in action	433
14.14	Fitting-a-sphere code	435
15.1	The meet and plunge of three spheres	439
15.2	The plunge of diverse elements	441
15.3	The meet and plunge of two spheres at decreasing distances	442
15.4	Visualization of flats as plunge	443
15.5	Orbits of a dual line versor	444
15.6	Tangents of elements	445
15.7	Factorization of rounds	447

15.8	Affine combination of conformal points	448
15.9	Affine combination of circles and point pairs	449
15.10	Orthogonal projections in the conformal model of Euclidean geometry	450
15.11	Various kinds of vectors in the conformal model	452
15.12	Definition of symbols for the Voronoi derivations	456
15.13	Construction of a contour circle	462
15.14	Screenshot of Example 2	463
15.15	Screenshot of Example 3 on projection and <code>plunge</code>	464
16.1	Inversion in a sphere	467
16.2	Reflection in a sphere	469
16.3	Generation of a snail shell	472
16.4	Swapping scaling and translation	473
16.5	Computation of the logarithm of a positively scaled rigid body motion	475
16.6	Loxodromes	478
16.7	Conformal orbits	479
16.8	Hyperbolic geometry	481
16.9	Spherical geometry	482
16.10	Imaging by the eye	484
16.11	Reflection in a point pair	485
16.12	Dupin cycloid as the inversion of a torus into a sphere	486
16.13	Metrical Mystery Tour	487
16.14	Function <code>matrix4x4 ToVersor()</code>	489
16.15	Function <code>log(const TRSVersor &V)</code>	493
16.16	Screenshot of Example 4	495
19.1	Function <code>canonical ReorderingSign(int a, int b)</code>	514
19.2	Function <code>gp_op (BasisBlade a, BasisBlade b)</code>	515
20.1	Matrices for geometric product, outer product, and left contraction	525
20.2	Implementation of the outer product of multivectors	527
21.1	Venn diagrams illustrating union, intersection, and the delta product of two sets	536
21.2	Venn diagrams illustrating <code>meet</code> , <code>join</code> , and the delta product of two blades	537
22.1	Basic tool-chain from source code to running application	544
22.2	Code generated by <code>Gaigen 2</code>	551
22.3	Generated matrix-point multiplication code	553

LIST OF FIGURES

xxv

23.1	Teapot polygonal mesh	560
23.2	Screenshot of the user interface of the modeler	567
23.3	Rotating an object	570
23.4	The spaceball interface	571

List of Tables

2.1	Geometrical properties of a subspace	43
2.2	Pascal's triangle of the number of basis k -blades in n -dimensional space	45
2.3	Notational conventions for blades and multivectors for Part I of this book	47
2.4	C++ Operator bindings	55
5.1	The order of the arguments for a <code>meet</code> may affect the sign of the result	134
7.1	Reflection of an oriented subspace \mathbf{X} in a subspace \mathbf{A}	190
8.1	Directional differentiation and vector derivatives	226
8.2	Elementary results of multivector differentiation	237
10.1	The point group $2H_4$	255
11.1	The geometric algebra of the homogeneous model of 3-D Euclidean space	273
11.2	The number of blades representing subspaces and directions	287
11.3	Nonzero blades in the homogeneous model of Euclidean geometry	291
11.4	Specialized multivector types in the <code>h3ga</code>	320
12.1	Common Plücker coordinate computations	330
12.2	Transformation of the flats in the homogeneous model	335
13.1	Multiplication table for the inner product of the conformal model of 3-D Euclidean geometry \mathbb{E}^3 , for two choices of basis	361
13.2	The interpretation of vectors in the conformal model	363
13.3	A list of the most important specialized multivector types in <code>c3ga</code>	391
13.4	Constants in <code>c3ga</code>	392
14.1	Nonzero blades in the conformal model of Euclidean geometry	407
16.1	Basic operations in the conformal model and their versors	476
16.2	Common proper transformations of some of the standard elements of the conformal model	477

18.1	Matrix representations of Clifford algebras of signatures (p, q)	507
19.1	The bitmap representation of basis blades	512
19.2	Bitwise boolean operators used in Java code examples	513
19.3	Reversion, grade involution, and Clifford Conjugate for basis blades	519
22.1	Performance benchmarks for the ray tracer	555

List of Programming Examples

Section	Title	Model	
1.1	An Example in Geometric Algebra	3-D conformal	5
2.13.1	Drawing Bivectors	2-D vector space	58
2.13.2	Exercise: Hidden Surface Removal	3-D vector space	61
2.13.3	Singularities in Vector Fields	3-D vector space	63
3.11.1	Orthonormalization	3-D vector space	93
3.11.2	Exercise: Implementing the Cross Product	3-D vector space	96
3.11.3	Reciprocal Frames	3-D vector space	97
3.11.4	Color Space Conversion	3-D vector space	98
4.9.1	Orthogonal Projection	3-D vector space	122
4.9.2	Orthogonal Projection, Matrix Representation	3-D vector space	122
4.9.3	Transforming Normal Vectors	3-D vector space	123
5.11.1	The Meet and Join	3-D vector space	138
5.11.2	Efficiency	3-D vector space	139
5.11.3	Floating Point Issues	3-D vector space	139
6.7.1	Exercise: Subspace Products Retrieved	3-D vector space	161
6.7.2	Gram-Schmidt Orthogonalization	3-D vector space	162
7.10.1	Reflecting in Vectors	3-D vector space	204
7.10.2	Two Reflections Equals One Rotation	3-D vector space	204
7.10.3	Matrix-Rotor Conversion 1	3-D vector space	204
7.10.4	Exercise: Matrix-Rotor Conversion 2	3-D vector space	206
7.10.5	Julia Fractals	2-D vector space	208
10.7.1	Interpolating Rotations	3-D vector space	265
10.7.2	Crystallography	3-D vector space	267

Section	Title	Model	
10.7.3	External Camera Calibration	3-D vector space	269
11.13.1	Working with Points	3-D homogeneous	321
11.13.2	Intersecting Primitives	3-D homogeneous	322
11.13.3	Don't Add Lines	3-D homogeneous	324
11.13.4	Perspective Projection	3-D homogeneous	326
12.5.1	Loading Transformations into OpenGL	3-D homogeneous	349
12.5.2	Transforming Primitives with OpenGL Matrices	3-D homogeneous	350
12.5.3	Marker Reconstruction in Optical Motion Capture	3-D homogeneous	352
13.10.1	Metric Matters	3-D conformal	390
13.10.2	Exercise: The Distance Between Points	3-D conformal	393
13.10.3	Loading Transformations into OpenGL, Again	3-D conformal	394
13.10.4	Interpolation of Rigid Body Motions	3-D conformal	395
14.9.1	Voronoi Diagrams and Delaunay Triangulations	2-D conformal	430
14.9.2	Exercise: Drawing Euclid's Elements	3-D conformal	431
14.9.3	Conformal Primitives and Intersections	3-D conformal	433
14.9.4	Fitting a Sphere to a Set of Points	3-D conformal	435
15.8.1	The Plunge	3-D conformal	462
15.8.2	Affine Combinations of Points	2-D conformal	463
15.8.3	Euclidean Projections	3-D conformal	464
16.10.1	Homogeneous 4×4 Matrices to Conformal Versors	3-D conformal	488
16.10.2	Logarithm of Scaled Rigid Body Motion	3-D conformal	493
16.10.3	Interpolation of Scaled Rigid Body Motions	3-D conformal	493
16.10.4	The Seashell	3-D conformal	494

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 sub-fields 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

