

Computational Geometry

Third Edition

Mark de Berg · Otfried Cheong
Marc van Kreveld · Mark Overmars

Computational Geometry

Algorithms and Applications

Third Edition

 Springer

Prof. Dr. Mark de Berg
Department of Mathematics
and Computer Science
TU Eindhoven
P.O. Box 513
5600 MB Eindhoven
The Netherlands
mdberg@win.tue.nl

Dr. Marc van Kreveld
Department of Information
and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands
marc@cs.uu.nl

Dr. Otfried Cheong, né Schwarzkopf
Department of Computer Science
KAIST
Gwahangno 335, Yuseong-gu
Daejeon 305-701
Korea
otfried@kaist.edu

Prof. Dr. Mark Overmars
Department of Information
and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands
markov@cs.uu.nl

ISBN 978-3-540-77973-5

e-ISBN 978-3-540-77974-2

DOI 10.1007/978-3-540-77974-2

ACM Computing Classification (1998): F.2.2, I.3.5

Library of Congress Control Number: 2008921564

© 2008, 2000, 1997 Springer-Verlag Berlin Heidelberg

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable for prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, 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.

Cover design: KünkelLopka, Heidelberg

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

Computational geometry emerged from the field of algorithms design and analysis in the late 1970s. It has grown into a recognized discipline with its own journals, conferences, and a large community of active researchers. The success of the field as a research discipline can on the one hand be explained from the beauty of the problems studied and the solutions obtained, and, on the other hand, by the many application domains—computer graphics, geographic information systems (GIS), robotics, and others—in which geometric algorithms play a fundamental role.

For many geometric problems the early algorithmic solutions were either slow or difficult to understand and implement. In recent years a number of new algorithmic techniques have been developed that improved and simplified many of the previous approaches. In this textbook we have tried to make these modern algorithmic solutions accessible to a large audience. The book has been written as a textbook for a course in computational geometry, but it can also be used for self-study.

Structure of the book. Each of the sixteen chapters (except the introductory chapter) starts with a problem arising in one of the application domains. This problem is then transformed into a purely geometric one, which is solved using techniques from computational geometry. The geometric problem and the concepts and techniques needed to solve it are the real topic of each chapter. The choice of the applications was guided by the topics in computational geometry we wanted to cover; they are not meant to provide a good coverage of the application domains. The purpose of the applications is to motivate the reader; the goal of the chapters is not to provide ready-to-use solutions for them. Having said this, we believe that knowledge of computational geometry is important to solve geometric problems in application areas efficiently. We hope that our book will not only raise the interest of people from the algorithms community, but also from people in the application areas.

For most geometric problems treated we give just one solution, even when a number of different solutions exist. In general we have chosen the solution that is easiest to understand and implement. This is not necessarily the most efficient solution. We also took care that the book contains a good mixture of techniques like divide-and-conquer, plane sweep, and randomized algorithms. We decided not to treat all sorts of variations to the problems; we felt it is more important to introduce all main topics in computational geometry than to give more detailed information about a smaller number of topics.

Several chapters contain one or more sections marked with a star. They contain improvements of the solution, extensions, or explain the relation between various problems. They are not essential for understanding the remainder of the book.

Every chapter concludes with a section that is entitled *Notes and Comments*. These sections indicate where the results described in the chapter originated, mention other solutions, generalizations, and improvements, and provide references. They can be skipped, but do contain useful material for those who want to know more about the topic of the chapter.

At the end of each chapter a number of exercises is provided. These range from easy tests to check whether the reader understands the material to more elaborate questions that extend the material covered. Difficult exercises and exercises about starred sections are indicated with a star.

A course outline. Even though the chapters in this book are largely independent, they should preferably not be treated in an arbitrary order. For instance, Chapter 2 introduces plane sweep algorithms, and it is best to read this chapter before any of the other chapters that use this technique. Similarly, Chapter 4 should be read before any other chapter that uses randomized algorithms.

For a first course on computational geometry, we advise treating Chapters 1–10 in the given order. They cover the concepts and techniques that, according to us, should be present in any course on computational geometry. When more material can be covered, a selection can be made from the remaining chapters.

Prerequisites. The book can be used as a textbook for a high-level undergraduate course or a low-level graduate course, depending on the rest of the curriculum. Readers are assumed to have a basic knowledge of the design and analysis of algorithms and data structures: they should be familiar with big-Oh notations and simple algorithmic techniques like sorting, binary search, and balanced search trees. No knowledge of the application domains is required, and hardly any knowledge of geometry. The analysis of the randomized algorithms uses some very elementary probability theory.

Implementations. The algorithms in this book are presented in a pseudo-code that, although rather high-level, is detailed enough to make it relatively easy to implement them. In particular we have tried to indicate how to handle degenerate cases, which are often a source of frustration when it comes to implementing.

We believe that it is very useful to implement one or more of the algorithms; it will give a feeling for the complexity of the algorithms in practice. Each chapter can be seen as a programming project. Depending on the amount of time available one can either just implement the plain geometric algorithms, or implement the application as well.

To implement a geometric algorithm a number of basic data types—points, lines, polygons, and so on—and basic routines that operate on them are needed. Implementing these basic routines in a robust manner is not easy, and takes a lot

of time. Although it is good to do this at least once, it is useful to have a software library available that contains the basic data types and routines. Pointers to such libraries can be found on our Web site.

Web site. This book is accompanied by a Web site, which contains a list of errata collected for each edition of the book, all figures and the pseudo code for all algorithms, as well as some other resources. The address is

`http://www.cs.uu.nl/geobook/`

You can also use the address given on our Web site to send us errors you have found, or any other comments you have about the book.

About the third edition. This third edition contains two major additions: In Chapter 7, on Voronoi diagrams, we now also discuss Voronoi diagrams of line segments and farthest-point Voronoi diagrams. In Chapter 12, we have included an extra section on binary space partition trees for low-density scenes, as an introduction to realistic input models. In addition, a large number of small and some larger errors have been corrected (see the list of errata for the second edition on the Web site). We have also updated the notes and comments of every chapter to include references to recent results and recent literature. We have tried not to change the numbering of sections and exercises, so that it should be possible for students in a course to still use the second edition.

Acknowledgments. Writing a textbook is a long process, even with four authors. Many people contributed to the original first edition by providing useful advice on what to put in the book and what not, by reading chapters and suggesting changes, and by finding and correcting errors. Many more provided feedback and found errors in the first two editions. We would like to thank all of them, in particular Pankaj Agarwal, Helmut Alt, Marshall Bern, Jit Bose, Hazel Everett, Gerald Farin, Steve Fortune, Geert-Jan Giezeman, Mordecai Golin, Dan Halperin, Richard Karp, Matthew Katz, Klara Kedem, Nelson Max, Joseph S. B. Mitchell, René van Oostrum, Günter Rote, Henry Shapiro, Sven Skyum, Jack Snoeyink, Gert Vegter, Peter Widmayer, Chee Yap, and Günther Ziegler. We also would like to thank Springer-Verlag for their advice and support during the creation of this book, its new editions, and the translations into other languages (at the time of writing, Japanese, Chinese, and Polish).

Finally we would like to acknowledge the support of the Netherlands' Organization for Scientific Research (N.W.O.) and the Korea Research Foundation (KRF).

January 2008

*Mark de Berg
Otfried Cheong
Marc van Kreveld
Mark Overmars*

Contents

1	Computational Geometry	1
	Introduction	
1.1	An Example: Convex Hulls	2
1.2	Degeneracies and Robustness	8
1.3	Application Domains	10
1.4	Notes and Comments	13
1.5	Exercises	15
2	Line Segment Intersection	19
	Thematic Map Overlay	
2.1	Line Segment Intersection	20
2.2	The Doubly-Connected Edge List	29
2.3	Computing the Overlay of Two Subdivisions	33
2.4	Boolean Operations	39
2.5	Notes and Comments	40
2.6	Exercises	41
3	Polygon Triangulation	45
	Guarding an Art Gallery	
3.1	Guarding and Triangulations	46
3.2	Partitioning a Polygon into Monotone Pieces	49
3.3	Triangulating a Monotone Polygon	55
3.4	Notes and Comments	59
3.5	Exercises	60
4	Linear Programming	63
	Manufacturing with Molds	
4.1	The Geometry of Casting	64
4.2	Half-Plane Intersection	66
4.3	Incremental Linear Programming	71
4.4	Randomized Linear Programming	76

4.5	Unbounded Linear Programs	79
4.6*	Linear Programming in Higher Dimensions	82
4.7*	Smallest Enclosing Discs	86
4.8	Notes and Comments	89
4.9	Exercises	91
5	Orthogonal Range Searching Querying a Database	95
5.1	1-Dimensional Range Searching	96
5.2	Kd-Trees	99
5.3	Range Trees	105
5.4	Higher-Dimensional Range Trees	109
5.5	General Sets of Points	110
5.6*	Fractional Cascading	111
5.7	Notes and Comments	115
5.8	Exercises	117
6	Point Location Knowing Where You Are	121
6.1	Point Location and Trapezoidal Maps	122
6.2	A Randomized Incremental Algorithm	128
6.3	Dealing with Degenerate Cases	137
6.4*	A Tail Estimate	140
6.5	Notes and Comments	143
6.6	Exercises	144
7	Voronoi Diagrams The Post Office Problem	147
7.1	Definition and Basic Properties	148
7.2	Computing the Voronoi Diagram	151
7.3	Voronoi Diagrams of Line Segments	160
7.4	Farthest-Point Voronoi Diagrams	163
7.5	Notes and Comments	167
7.6	Exercises	170
8	Arrangements and Duality Supersampling in Ray Tracing	173
8.1	Computing the Discrepancy	175
8.2	Duality	177
8.3	Arrangements of Lines	179
8.4	Levels and Discrepancy	185

8.5	Notes and Comments	186	CONTENTS
8.6	Exercises	188	
9	Delaunay Triangulations	191	
	Height Interpolation		
9.1	Triangulations of Planar Point Sets	193	
9.2	The Delaunay Triangulation	196	
9.3	Computing the Delaunay Triangulation	199	
9.4	The Analysis	205	
9.5*	A Framework for Randomized Algorithms	208	
9.6	Notes and Comments	214	
9.7	Exercises	215	
10	More Geometric Data Structures	219	
	Windowing		
10.1	Interval Trees	220	
10.2	Priority Search Trees	226	
10.3	Segment Trees	231	
10.4	Notes and Comments	237	
10.5	Exercises	239	
11	Convex Hulls	243	
	Mixing Things		
11.1	The Complexity of Convex Hulls in 3-Space	244	
11.2	Computing Convex Hulls in 3-Space	246	
11.3*	The Analysis	250	
11.4*	Convex Hulls and Half-Space Intersection	253	
11.5*	Voronoi Diagrams Revisited	254	
11.6	Notes and Comments	256	
11.7	Exercises	257	
12	Binary Space Partitions	259	
	The Painter's Algorithm		
12.1	The Definition of BSP Trees	261	
12.2	BSP Trees and the Painter's Algorithm	263	
12.3	Constructing a BSP Tree	264	
12.4*	The Size of BSP Trees in 3-Space	268	
12.5	BSP Trees for Low-Density Scenes	271	
12.6	Notes and Comments	278	
12.7	Exercises	279	

13 Robot Motion Planning	283
Getting Where You Want to Be	
13.1 Work Space and Configuration Space	284
13.2 A Point Robot	286
13.3 Minkowski Sums	290
13.4 Translational Motion Planning	297
13.5* Motion Planning with Rotations	299
13.6 Notes and Comments	303
13.7 Exercises	305
14 Quadtrees	307
Non-Uniform Mesh Generation	
14.1 Uniform and Non-Uniform Meshes	308
14.2 Quadtrees for Point Sets	309
14.3 From Quadtrees to Meshes	315
14.4 Notes and Comments	318
14.5 Exercises	320
15 Visibility Graphs	323
Finding the Shortest Route	
15.1 Shortest Paths for a Point Robot	324
15.2 Computing the Visibility Graph	326
15.3 Shortest Paths for a Translating Polygonal Robot	330
15.4 Notes and Comments	331
15.5 Exercises	332
16 Simplex Range Searching	335
Windowing Revisited	
16.1 Partition Trees	336
16.2 Multi-Level Partition Trees	343
16.3 Cutting Trees	346
16.4 Notes and Comments	352
16.5 Exercises	353
Bibliography	357
Index	377