Preface

The representation of multidimensional data is an important issue in diverse fields including database management systems (e.g., spatial databases, multimedia databases), computer graphics, game programming, computer vision, geographic information systems (GIS), pattern recognition, similarity searching (also known as similarity retrieval), solid modeling, computer-aided design (CAD), robotics, image processing, computational geometry, finite-element analysis, and numerous others. The key idea in choosing an appropriate representation is to facilitate operations such as search. This means that the representation involves sorting the data in some manner to make it more accessible. In fact, the term *access structure* or *index* is often used as an alternative to the term *data structure* in order to emphasize the importance of the connection to sorting.

The most common definition of "multidimensional data" is a collection of points in a higher dimensional space (i.e., greater than 1). These points can represent locations and objects in space as well as more general records where each attribute (i.e., field) corresponds to a dimension and only some, or even none, of the attributes are locational. As an example of nonlocational point data, consider an employee record that has attributes corresponding to the employee's name, address, sex, age, height, weight, and social security number. Such records arise in database management systems and can be treated as points in, for this example, a seven-dimensional space (i.e., there is one dimension for each attribute), although the different dimensions have different type units (i.e., name and address are strings of characters; sex is binary; while age, height, weight, and social security number are numbers). Note that the address attribute could also be interpreted in a locational sense using positioning coordinates such as latitude and longitude readings although the stringlike symbolic representation is far more common.

When multidimensional data corresponds to locational data, we have the additional property that all of the attributes usually have the same unit (possibly with the aid of scaling transformations), which is distance in space. In this case, we can combine the distance-denominated attributes and pose queries that involve proximity. For example, we may wish to find the closest city to Chicago within the two-dimensional space from which the locations of the cities are drawn. Another query seeks to find all cities within 50 miles of Chicago. In contrast, such queries are not very meaningful when the attributes do not have the same type. Nevertheless, other queries such as range queries that seek, for example, all individuals born between 1940 and 1960 whose weight ranges between 150 and 200 pounds are quite common and can be posed regardless of the nature of the attributes.

When multidimensional data spans a continuous physical space (i.e., an infinite collection of locations), the issues become more interesting. In particular, we are no longer just interested in the locations of objects, but, in addition, we are also interested in the space that they occupy (i.e., their extent). Some example objects with extent include lines (e.g., roads, rivers), intervals (which can correspond to time as well as

space), regions of varying shape and dimensionality (e.g., lakes, counties, buildings, crop maps, polygons, polyhedra), and surfaces. The objects (when they are not points) may be disjoint or could even overlap.

Spatial multidimensional data is generally of relatively low dimension. However, there are many application domains where the data is of considerably higher dimensionality and is not necessarily spatial. This is especially true in databases for application domains such as multimedia and bioinformatics data, to name just two, where the data is a set of objects, and the high dimensionality is a direct result of attempts to describe the objects via a collection of features that characterize them (also known as a *feature vector*). For example, in the case of image data, some commonly used features include color, color moments, textures, shape descriptions, and so on, expressed using scalar values. Although such data falls into the class of point data, it often requires using very different representations than those that are traditionally used for point data. The problem is often attributed to the *curse of dimensionality* [159], which simply means that the indexing methods do not do as good a job of facilitating the retrieval of the objects that satisfy the query as is the case when the data has lower dimensionality.

The above discussion of nonspatial data of high dimensionality has been based on the premise that we know the features that describe the objects (and hence the dimensionality of the underlying feature space). In fact, it is usually quite difficult to identify the features, and thus we frequently turn to experts in the application domain from which the objects are drawn for assistance in this process. Nevertheless, often the features cannot be easily identified even by the domain experts. In particular, sometimes the only information available is a distance function that indicates the degree of similarity (or dissimilarity) between all pairs of objects in the database as well as how to compute the similarity between objects in the domain. Usually the distance function d is required to obey the triangle inequality, be nonnegative, and be symmetric, in which case it is known as a *metric* (also referred to as a *distance metric*). In this case, we resort to what are termed metric or distance-based representations, which differ from the more conventional representations in that the objects are now sorted with respect to their distance from a few key objects, known as *pivots*, rather than with respect to some commonly known reference point (e.g., the origin in the spatial domain), which may not necessarily be associated with any of the objects.

This book is about the representation of data such as that described above. It is organized into four chapters. Each chapter is like a part, sometimes as big as a book. Chapter 1 deals with the representation of point data, usually of low dimensionality. One of the keys to the presentation in this chapter is making a distinction between whether the representation organizes the data (e.g., a binary search tree) or the embedding space from which the data is drawn (e.g., a trie). Considerable attention is also devoted to issues that arise when the volume of the data is so high that it cannot fit in memory, and hence the data is disk resident. Such methods are characterized as bucket methods and have been the subject of extensive research in the database community.

Chapter 2 deals with the representation of objects. It focuses on object-based and image-based representations. These distinctions are analogous to those made in computer graphics where algorithms and decompositions of objects are characterized as being object-based or image-based corresponding to whether the representations result from the decomposition of the objects or the environment in which the objects are embedded, respectively. The central theme in this chapter is how the representations deal with the key queries of where and what—that is, given an object, we want to know its location, and, given a location, we want to know the identity of the object (or objects) of which it is a member. Efficiently responding to these queries, preferably with one representation, has motivated many researchers in computer graphics and related fields such as computer vision, solid modeling, and spatial databases, to name a few. This leads, in part, to a distinction between representations on the basis of being either object hierarchies, of which the R-tree [791] and its variants (commonly used in database applications) are examples, or space hierarchies, of which the pyramid and its variants (commonly

used in computer vision applications) are examples. The representations that have been developed can also be differentiated on the basis of whether the objects are represented by their interiors or by their boundaries.

Chapter 3 deals with the representation of intervals and small rectangles. These representations are applicable in a number of important areas. For example, they are used in very large-scale integration (VLSI) design applications (also known as CAD), where the objects are modeled as rectangles for the sake of simplicity. The number of objects in these applications is quite large, and the size of the objects is several orders of magnitude smaller than that of the space from which they are drawn. This is in contrast to cartographic applications, where the sizes of the objects are larger. These representations are also used in game programming applications (e.g., [475, 476, 1025, 1456, 1890]), where their deployment is motivated by the fact that the objects are in spatial registration, thereby enabling efficient calculation of set-theoretic operations while not being as sensitive to changes in position as is the case for some other hierarchical representations that are based on a regular decomposition of the underlying space (e.g., the region quadtree). The representation of intervals is also of interest for applications that involve temporal data where intervals play a role (e.g., [951]). They can also be used for the representation of moving objects, which finds use in spatiotemporal databases (e.g., [21, 950, 1057, 1092, 1305, 2013] and the text of Güting and Schneider [790]).

Chapter 4 deals with the representation of high-dimensional data where the data is not spatial and the actual dimensionality of the data is also an issue. The key is that the data usually consists of features where the number and identity of the features that should be represented is not always clear. In particular, this data arises in applications such as multimedia, where the use of a large number of features to describe the objects is based on the premise that by increasing the number of features we are increasing our ability to differentiate between the objects for retrieval. An alternative is to make use of a distance function that measures the similarity (or dissimilarity) between the objects. A discussion of a number of distance-based indexes is included here.

Chapter 4 is motivated by the important problem of *similarity searching*—the process of finding and retrieving objects that are similar to a given object or set of objects. This process invariably reduces to finding the nearest or k nearest neighbors, and hence a considerable amount of attention is paid to this query by showing how to perform it efficiently for both distance-based and multidimensional indexing methods. The observation that, for high-dimensional data, the use of multidimensional indexing methods often fails to yield any improvement in performance over a simple sequential scan of the data is also addressed. This is in part due to the cost of computing distance in high dimensions, the nature of the data distribution, and the relatively low volume of available data vis-à-vis its dimensionality (i.e., the curse of dimensionality). One response is to take note of the observation that the "inherent dimensionality" of a dataset is often much lower than that of the underlying space. In particular, this has led to the development of techniques to reduce the dimensionality of the data using methods such as Singular Value Decomposition (SVD). An alternative solution that is discussed is how to make use of contractive embedding methods that embed the data in a lower-dimensional space, and then make use of a filter-and-refine algorithm in conjunction with multidimensional indexing methods to prune irrelevant candidates from consideration.

Many of the data structures that are presented in the four chapters are hierarchical. They are based either on the principle of recursive decomposition (similar to *divide-and-conquer* methods [27]) or on the aggregation of objects into hierarchies. Hierarchical data structures are useful because of their ability to focus on the interesting subsets of the data. This focusing results in an efficient representation and in improved execution times.

Hierarchical representations of spatial data were the subject of one of my earlier books [1637]. It dealt with the four basic spatial data types: points, rectangles, lines, and volumes. Many of these representations were based on variants of the quadtree and

octree data structure that have been applied to many problems in a variety of domains. In this case, the term *quadtree* was used in a general sense to describe a class of representations whose common property was that they were based on the principle of recursive decomposition of space. They were differentiated on the following bases:

1. The type of data they are used to represent
2. The principle guiding the decomposition process
3. The resolution (variable or not)

This book is a considerable expansion of [1637] with approximately 10% overlap. Nevertheless there still remains a considerable amount of material from [1637] that has not been included in this book (e.g., from Chapters 4 and 5). One of the features of [1637] was the inclusion of detailed code for building and updating a number of data structures (at least one for each of the spatial data types point, lines, and rectangles). This code is also present in this book but is now part of the exercises and their solutions. In addition, the VASCO (Visualization and Animation of Spatial Constructs and Operations) [243, 244, 245, 246]) set of JAVA applets is available for use to illustrate many operations for a large number of the data structures presented in the current book (see *http://www.cs.umd.edu/~hjs/quadtree/index.html*). Readers interested in seeing how these methods are used in the context of an application should try the SAND Internet Browser [1644] at the same URL, which provides an entry point to explore their use in a geographic information system (GIS). The SAND Internet Browser is part of a considerably more complex system rooted in the QUILT quadtree GIS [1732], the SAND spatial database engine [71, 73], and the SANDTcl scripting tool for building spatial database applications [568, 570].

One substantial difference in the current book from [1637] is the inclusion of a part on high-dimensional data, representing approximately one third of the text (Chapter 4). In addition, almost all of the material on object-based and image-based representations in Chapter 2 is new. It is interesting to note that this material and the observation that most object representations are motivated by the desire to respond to the *what* and *where* queries served as the motivation for this book. Similarly, much of the material on point-based representations in Chapter 1 is also new with a particular emphasis on bucketing methods. In the case of linear hashing and spiral hashing, the explanations of the methods have been moved to Appendix B and C, respectively, while Chapter 1 focuses on how to adapt them to multidimensional data. Since the B-tree is really a fundamental foundation of the various bucket methods in Chapter 1 and of hierarchical object-based interior representations such as the R-tree in Chapter 2, an overview of B-trees has been included as Appendix A. This was motivated by our past experience with [1637], which revealed that often the extent to which readers were familiar with the B-tree did not match our expectations. Only Chapter 3, which deals with intervals and collections of rectangles, can be considered as both an update and an expansion of material in [1637] as it also contains a substantial number of new representations.

The main focus of this book is on representations and indexing methods rather than the execution of operations using them, which is the primary focus of another one of my books [1636]. Nevertheless, a few operations are discussed in greater detail. In particular, substantial coverage is given to the k-nearest neighbor finding operation in Sections 4.1 and 4.2 in Chapter 4. Its use in similarity searching is the main reason for the development of most of the representations discussed in this chapter. The related point location problem is also discussed in great detail in Sections 2.1.3.2 and 2.1.3.3 of Chapter 2. Another operation that is discussed in some detail (in Section 2.2.3.4 of Chapter 2) is simplification of surface data for applications in computer graphics and visualization. We assume that the representations and operations that we discuss are implemented in a serial environment although there has been a considerable interest in implementations in a parallel environment (e.g., [210, 213, 373, 470, 524, 532, 862, 898, 916, 997, 1182, 1719, 2006] and related references cited therein), as well as in making use of graphics processing units (GPU) (e.g., [739, 1146, 1815]).

Nevertheless there remain many topics for which justice requires a considerably more thorough treatment. However, due to space limitations, detailed discussion of them has been omitted, and, instead, the interested reader is referred to the appropriate literature. For example, surface representations are discussed in the context of boundary-based representations in Section 2.2 of Chapter 2. These are known as topological data structures. However, the area is much richer than what is covered here as can be seen by referring to the collection of papers edited by Rana [1536]. The presentation in Chapter 2 mentions simplification methods and data structures (Section 2.2.3.4) and hierarchical surface-based boundary representations (Sections 2.2.2–2.2.4). Simplification methods are also known as *Level of Detail (LOD) methods* and are reviewed comprehensively in the text by Luebke, Reddy, Cohen, Varshney, Watson, and Huebner [1212]. Additional work from the perspective of geometric modeling can be found in the texts by Bartels, Beatty, and Barsky [131], Mortenson [1316], Goldman [716], Farin [596], and Warren and Weimer [1962]. Many of the topics discussed here can also be classified under the category of *visual computing* (e.g., see Nielsen [1372]). Another related area in which our coverage is intentionally limited is real-time collision detection where speed is of the essence, especially in applications such as game programming. For a thorough treatment of this topic, see the recent texts by van den Bergen [198] and Ericson [564]. Similarly, representations for spatiotemporal data and moving objects are not discussed in great detail, and interested readers are referred to some of the references mentioned earlier.

Solid modeling is another rich area where the presentation has been limited to the boundary model (BRep), found in Section 2.2.1 of Chapter 2, which also includes a detailed discussion of variants of the winged-edge representation. For more detailed expositions on this and related topics, see the books by Mäntylä [1232], Hoffmann [870], and Paoluzzi [1457]. See also some early influential survey articles by Requicha [1555], Srihari [1798], and Chen and Huang [345]. Of course, there is also a much greater connection to computer graphics than that which is made in the text. This connection is explored further in another of my books [1636] and is also discussed in some early surveys by Overmars [1442] and Samet and Webber [1666, 1667].

Results from computational geometry, although related to many of the topics covered in this book, are only presented in the context of representations for intervals and collections of small rectangles (Sections 3.1 and 3.2 of Chapter 3) and the point location problem (Sections 2.1.3.2 and 2.1.3.3 of Chapter 2). For more details on early work involving some of these and related topics, the interested reader is encouraged to consult the texts by Preparata and Shamos [1511], Edelsbrunner [540], Boissonnat and Yvince [230], and de Berg, van Kreveld, Overmars, and Schwarzkopf [196], as well as the early collections of papers edited by Preparata [1509] and Toussaint [1884]. See also the surveys and problem collections by Edelsbrunner [539]. O'Rourke [1423], and Toussaint [1885]. This work is closely related to earlier research on searching as surveyed by Bentley and Friedman [173], the text of Overmars [1440], and Mehlhorn's [1282] text, which contains a unified treatment of multidimensional searching. For a detailed examination of similar and related topics in the context of external memory and massive amounts of data, see the survey of Vitter [1933].

Many of the data structures described in this book find use in spatial databases, geographic information systems (GIS), and multimedia databases. The close relationship between spatial databases and geographic information systems is evidenced by recent texts that combine results from the areas, such as those of Laurini and Thompson [1113], Worboys [2023], Günther [773], Rigaux, Scholl, and Voisard [1562], and Shekhar and Chawla [1746], as well as the surveys of Güting [784], Voisard and David [1938], and Shekhar, Chawla, Ravada, Fetterer, Liu, and Lu [1747].

For a slightly different perspective, see the text of Subrahmanian [1807], which deals with multimedia databases. Another related area where these data structures have been used is constraint databases, which are discussed in great detail in the recent text by Revesz [1559]. Note that, with the exception of the nearest neighbor finding operation, which is common to many of the fields spanned by this book, we do not discuss the

numerous spatial database operations whose execution is facilitated by the data structures presented in this text, such as, for example, spatial join (e.g., [89, 262, 261, 772, 1072, 894, 934, 1195, 1207, 1223, 1415, 1474, 2076] as well as in a recent survey [935]), which are covered in some of the previously cited references.

In a similar vein, a number of survey articles have been written that deal with the representations that are described in this book from a database perspective, such as the one by Gaede and Günther [670]. The survey article by Samet [1642], however, combines a computer graphics/computer vision perspective with a database perspective. See also the earlier surveys by Nagy and Wagle [1335], Samet and Rosenfeld [1647], Peuquet [1490], and Samet [1628, 1632]. Others, such as Chávez, Navarro, Baeza-Yates, and Marroquín [334], Böhm, Berchtold, and Keim [224], Hjaltason and Samet [854, 855], and Clarkson [401], are oriented towards similarity searching applications.

The material in this book also has a strong connection to work in image processing and computer vision. In particular, the notion of a pyramid (discussed in Section 2.1.5.1 of Chapter 2) has a rich history, as can be seen in the collection of articles edited by Tanimoto and Klinger [1844] and Rosenfeld [1573]. The connection to pattern recognition is also important and can be seen in the survey by Toussaint [1882]. The pioneering text by Rosenfeld and Kak [1574] is a good early treatment of image processing and should be consulted in conjunction with the more recent text by Klette and Rosenfeld [1034], which makes the connection to digital geometry.

Nevertheless, given the broad and rapidly expanding nature of the field, I am bound to have omitted significant concepts and references. In addition, at times I devote a disproportionate amount of attention to some concepts at the expense of others. This is principally for expository purposes; in particular, I feel that it is better for the reader to understand some structures well rather than to receive a quick runthrough of buzzwords. For these indiscretions, I beg your pardon and still hope that you bear with me.

Usually my approach is an algorithmic one. Whenever possible, I have tried to motivate critical steps in the algorithms by a liberal use of examples. This is based on my strong conviction that it is of paramount importance for the reader to see the ease with which the representations can be implemented and used. At times, some algorithms are presented using pseudo-code so that readers can see how the ideas can be applied. I have deliberately not made use of any specific programming language in order to avoid dating the material while also acknowledging the diverse range in opinion at any given instance of time as to which language is appropriate. The main purpose of the pseudo-code is to present the algorithm clearly. The pseudo-code is algorithmic and is a variant of the ALGOL [1346] programming language, which has a data structuring facility incorporating pointers and record structures. I do not make use of object orientation although my use of record structures is similar in spirit to rudimentary classes in SIMULA [437] and FLEX [1008], which are the precursors of modern object-oriented methods. I make heavy use of recursion. This language has similarities to C [1017], C++ [1805], Java (e.g., [93]), PASCAL [952], SAIL [1553], and ALGOL W [137]. Its basic features are described in Appendix D. However, the actual code is not crucial to understanding the techniques, and it may be skipped on a first reading. The index indicates the page numbers where the code for each algorithm is found.

In many cases I also give an analysis of the space and time requirements of different data structures and algorithms. The analysis is usually of an asymptotic nature and is in terms of *big O* and Ω notation [1043]. The *big O* notation denotes an upper bound. For example, if an algorithm takes $O(\log_2 N)$ time, then its worst-case behavior is never any worse than $\log_2 N$. The Ω notation denotes a lower bound. As an example of its use, consider the problem of sorting N numbers. When we say that sorting is $\Omega(N \cdot \log_2 N)$ we mean that given any algorithm for sorting, there is some set of N input values for which the algorithm will require at least this much time.

At times, I also describe implementations of some of the data structures for the purpose of comparison. In such cases, counts such as the number of fields in a record are often given. However, these numbers are only meant to clarify the discussion. They are

not to be taken literally, as improvements are always possible once a specific application is analyzed more carefully.

Each section contains a substantial number of exercises. Many of the exercises develop the material in the text further as a means of testing the reader's understanding, as well as suggesting future directions. When the exercise or its solution is not my own, I have credited its originator by preceding it with their name. The exercises have not been graded by difficulty. Usually, their solution does not require any mathematical skills beyond the undergraduate level. However, while some of the exercises are quite straightforward, the solutions to others require some ingenuity. Solutions, or references to papers that contain the solutions, are provided for a substantial number of the exercises that do not require programming. Of course, the reader is counseled to try to solve the exercises before turning to the solutions. The solutions are presented here as it is my belief that much can be learned by self-study (both for the student and, even more so, for the author!). The motivation for pursuing this approach was my wonderful experience on my first encounter with the rich work on data structures by Knuth [1044, 1045, 1046].

An extensive bibliography is provided. It contains entries for both this book and its companion predecessors [1636, 1637]. Each reference is annotated with one or more keywords and a list of the numbers of the book sections in which it is cited in the current text (denoted by *F*) or either of the other two texts (denoted by *D* for [1637] and *A* for [1636]) as well as in the exercises and solutions. Not all of the references that appear in the bibliography are cited in the text. They are included for the purpose of providing readers the ability to access as completely as possible the body of literature relevant to the topics discussed in the books. In order to increase the usefulness of these uncited references (as well as all of the references), a separate keyword index is provided that indicates some of the references in the bibliography that discuss them. Of course, this list is not exhaustive. For the more general categories (e.g., "points"), a second level index entry is often provided to narrow the search in locating relevant references. In addition, a name and credit index is provided that indicates the page numbers in this book on which each author's work is cited or a credit (i.e., acknowledgement) is made (e.g., in the main text, preface, appendices, exercises, or solutions).

My target audience is a varied one. The book is designed to be of use to professionals who are interested in the representation of spatial and multidimensional data and have a need for an understanding of the various issues that are involved so that they can make a proper choice in their application. The presentation is an algorithmic one rather than a mathematical one although the execution time and storage requirements are discussed. Nevertheless, the reader need not be a programmer nor a mathematician to appreciate the concepts described here. The material is useful not just to computer scientists. It is useful for applications in a diverse range of fields including geography, high-energy physics, engineering, mathematics, games, multimedia, bioinformatics, and other disciplines where location or the concept of a metric are important. The book can also be used in courses on a number of topics including databases, geographic information systems (GIS), computer graphics, computational geometry, pattern recognition, and so on, where the discussion of the data structures that it describes supplements existing material or serves as the central topic of the course. The book can also be used in a second course on data structures where the focus is on multidimensional and metric data structures. The organization of the material to be covered in these courses is discussed in greater detail below.

A Guide to the Instructor

This book can be used in a course on representations for multidimensional and metric data in multimedia databases, most likely at the graduate level. Such a course would include topics from the various parts as is appropriate to match the interests of the students. It would most likely be taught within a database framework, in which case the focus

would be on the representation of points as in Chapter 1 and high-dimensional data in Chapter 4. In particular, the emphasis would be on bucket methods (Sections 1.1, 1.7, and 1.8 of Chapter 1), multidimensional indexing methods for high-dimensional data (Section 4.4 of Chapter 4), and distance-based indexing methods (Section 4.5 of Chapter 4). In addition, quite a bit of attention should be focused on representations based on object hierarchies such as the R-tree (Section 2.1.5.2 of Chapter 2).

The book can also be used in a computer graphics/computer vision/solid modeling framework, as well as part of a game programming course, in which case the emphasis would be on Chapters 2 and 3. In addition, it should include the discussion of some of the more basic representations of point data from Sections 1.1 and 1.4–1.6 of Chapter 1. Such a course could serve as a prerequisite to courses in computer graphics and solid modeling, computational geometry, database management systems (DBMS), multidimensional searching, image processing, and VLSI design applications. More specifically, such a course would include the discussions of interior-based representations of image data in Sections 2.1.1–2.1.3 of Chapter 2 and representations of image data by their boundaries in Sections 2.2.2 and 2.2.3 of Chapter 2. The representation of two-dimensional regions such as chain codes in Section 2.3.2 of Chapter 2 and polygonal representations in Section 2.2.2 of Chapter 2 and the use of point methods for focusing the Hough Transform in Section 1.5.1.4 of Chapter 1 should also be included as they are relevant to image processing.

The discussions of plane-sweep methods and their associated data structures such as the segment tree, interval tree, and priority search tree in Sections 3.1 and 3.2 of Chapter 3, and of point location and associated data structures such as the K-structure and the layered dag in Sections 2.1.3.2 and 2.1.3.3, respectively, of Chapter 2, are all relevant to computational geometry. Other relevant topics in this context are the Voronoi Diagram and the Delaunay triangulation and tetrahedralization in Sections 2.2.1.4–2.2.1.7 of Chapter 2 as well as the approximate Voronoi diagram (AVD) in Section 4.4.5 of Chapter 4. The discussion of boundary-based representations such as the winged-edge data structure in Sections 2.2.1.1–2.2.1.3 of Chapter 2 is also relevant to computational geometry as well as to solid modeling. In addition, observe that the discussions of rectangle-representation methods and plane-sweep methods in Chapter 3 are of use for VLSI design applications.

Moreover, it is worth repeating the earlier comment that the discussion of bucket methods such as linear hashing, spiral hashing, grid file, and EXCELL, discussed in Sections 1.7 and 1.8 of Chapter 1, as well as the various variants of object hierarchies as typified by the R-tree in Section 2.1.5.2 of Chapter 2, are important in the study of database management systems. The discussion of topics such as k-d trees in Section 1.5 of Chapter 1, range trees in Section 1.2 of Chapter 1, priority search trees priority search tree in Section 1.3 of Chapter 1, and point-based rectangle representations in Section 3.3 of Chapter 3 are all relevant to multidimensional searching.

Another potential venue for using the book is a course on representations for spatial data either in a geography department where it would be part of a curriculum on geographic information systems (GIS) or within the context of course on spatial databases in a computer science department. Again, the material that would be covered would be a combination of that which is covered in a multimedia database course and the computer graphics/computer vision/solid modeling or game programming course. In this case, much of the material on the representation of high-dimensional data in Chapter 4 would be omitted although the material on nearest neighbor finding and approximate nearest neighbor finding (Sections 4.1–4.3 of Chapter 4) would be included. In addition, a greater emphasis would be placed on the representation of objects by their interiors (Section 2.1 of Chapter 2) and by their boundaries (Section 2.2 of Chapter 2).

This book could also be used in a second data structures course. The emphasis would be on the representation of spatial data. In particular, the focus would be on the use of the principle of "divide-and-conquer." Hierarchical data structures provide a good demonstration of this principle. In this case, the choice of topics is left to the

instructor although one possible organization of topics would choose some of the simpler representations from the various chapters as follows.

From Chapter 1, some interesting topics include the fixed-grid method and some of its variants (Section 1.1); numerous variants of quadtrees such as the point quadtree (Section 1.4.1), PR quadtree (Section 1.4.2.2), k-d tree (Section 1.5), and variants thereof (Sections 1.5.1.4, 1.5.2.1, and 1.5.2.3); bucket methods such as the grid file (Section 1.7.2.1), EXCELL (Section 1.7.2.2), linear hashing (Section 1.7.2.3.1 and Appendix B); range trees (Section 1.2); priority search trees for points (Section 1.3); and the comparison of some of the structures (Section 1.9).

From Chapter 2, some interesting topics include the opening matter (preambles of both the chapter and of Section 2.1); unit-size cells (Section 2.1.1 with the exception of the discussion of cell shapes and tilings in Section 2.1.1.1); blocks (Sections 2.1.2.1–2.1.2.9); BSP tree (Section 2.1.3.1); winged-edge data structure (Sections 2.2.1.1 and 2.2.1.2); image-based boundary representations such as the MX quadtree and MX octree (Section 2.2.2.2), PM quadtree (Section 2.2.2.6), and PM octree (Section 2.2.2.7); and hierarchical object-based interior representations such as the R-tree (Section 2.1.5.2.1).

From Chapter 3, some interesting topics include the discussion of plane-sweep methods and data structures such as the segment tree (Section 3.1.1), interval tree (Section 3.1.2), and the priority search tree for rectangles and intervals (Section 3.1.3) and the MX-CIF quadtree (Section 3.4.1).

From Chapter 4, the most interesting topic is the incremental nearest neighbor algorithm (Sections 4.1.1–4.1.3). A brief introduction to distance-based indexing methods is also appropriate. In this case the most appropriate representations for inclusion are the vp-tree (Sections 4.5.2.1.1 and 4.5.2.1.1), the gh-tree (Section 4.5.3.1), and the mb-tree (Section 4.5.3.3).

Throughout the book, both worst-case optimal methods and methods that work well in practice are emphasized. This is in accordance with my view that the well-rounded computer scientist should be conversant with both types of algorithms. It is clear that the material in this book is more than can be covered in one semester. Thus the instructor will invariably need to reduce it as necessary. For example, in many cases, the detailed examples can be skipped or used as a basis of a term project or programming assignments.

Note that regardless of the course in which this book is used, as pointed out earlier, the VASCO [243, 244, 245, 246]) set of JAVA applets can be used to illustrate many operations for a large number of the data structures presented in the current book (see *http://www.cs.umd.edu/~hjs/quadtree/index.html*). In particular, for point data, it includes the point quadtree, k-d tree (also known as a point k-d tree), MX quadtree, PR quadtree, bucket PR quadtree, PR k-d tree, bucket PR k-d tree, PMR quadtree, PMR k-d tree, 2-d range tree (also known as a two-dimensional range tree), priority search tree, R-tree, and the PK-tree. For line data, it includes the PM_1 quadtree, PM_2 quadtree, PM_3 quadtree, PMR quadtree, bucket PM quadtree, and the R-tree. For collections of rectangles, it includes the MX-CIF quadtree, rectangle quadtree, bucket rectangle quadtree (also known as a bucket rectangle PM quadtree), PMR rectangle quadtree (also known as a PMR rectangle quadtree), PMR rectangle k-d tree, and R-tree. For all of these data structures, students can see how they are built step by step, and also how queries such as deletion, finding nearest neighbors, and overlap queries (which include the more general range query and corridor or buffer queries) are supported. In each case, the operation proceeds incrementally. Finally, an applet is also provided for the region quadtree that shows how it can be constructed from other region representations such as rasters and chain codes as well as how to obtain them from the region quadtree.

Instructors who adopt this book in a course are also welcome to use a comprehensive set of lecture notes in the form of Powerpoint/PDF presentations that help clarify many of the concepts presented in the book.