SIAM J. COMPUT. © 2005 Society for Industrial and Applied Mathematics
Vol. 35, No. 1, pp. 22–58

# A PROBABILISTIC ANALYSIS OF TRIE-BASED SORTING OF LARGE COLLECTIONS OF LINE SEGMENTS IN SPATIAL DATABASES[*]

MICHAEL LINDENBAUM[†], HANAN SAMET[‡], AND GISLI R. HJALTASON[§]

**Abstract.** The size of five trie-based methods of sorting large collections of line segments in a spatial database is investigated analytically using a random lines image model and geometric probability techniques. The methods are based on sorting the line segments with respect to the space that they occupy. Since the space is two-dimensional, the trie is formed by interleaving the bits corresponding to the binary representation of the $x$ and $y$ coordinates of the underlying space and then testing two bits at each iteration. The result of this formulation yields a class of representations that are referred to as *quadtrie* variants, although they have been traditionally referred to as *quadtree* variants. The analysis differs from prior work in that it uses a detailed explicit model of the image instead of relying on modeling the branching process represented by the tree and leaving the underlying image unspecified. The analysis provides analytic expressions and bounds on the expected size of these quadtree variants. This enables the prediction of storage required by the representations and of the associated performance of algorithms that rely on them. The results are useful in the following two ways:

1. They reveal the properties of the various representations and permit their comparison using analytic, nonexperimental criteria. Some of the results confirm previous analyses (e.g., that the storage requirement of the MX quadtree is proportional to the total lengths of the line segments). An important new result is that for a PMR and Bucket PMR quadtree with sufficiently high values of the splitting threshold (i.e., $\geq 4$) the number of nodes is proportional to the number of line segments and is independent of the maximum depth of the tree. This provides a theoretical justification for the good behavior and use of the PMR quadtree, which so far has been only of an empirical nature.
2. The random lines model was found to be general enough to approximate real data in the sense that the properties of the trie representations, when used to store real data (e.g., maps), are similar to their properties when storing random lines data. Therefore, by specifying an equivalent random lines model for a real map, the proposed analytical expressions can be applied to predict the storage required for real data. Specifying the equivalent random lines model requires only an estimate of the effective number of random lines in it. Several such estimates are derived for real images, and the accuracy of the implied predictions is demonstrated on a real collection of maps. The agreement between the predictions and real data suggests that they could serve as the basis of a cost model that can be used by a query optimizer to generate an appropriate query evaluation plan.

**Key words.** large spatial databases, tries, sorting line segments, geometric probability, analysis of algorithms, spatial data structures, quadtrees, quadtries, cost model, query evaluation

**AMS subject classification.** 68W40

**DOI.** 10.1137/S0097539700368527

[*]Received by the editors February 25, 2000; accepted for publication (in revised form) March 24, 2005; published electronically September 8, 2005. This work was supported in part by the National Science Foundation under grants IRI-9712715, EIA-99-00268, EIA-99-01636, EAR-99-05844, IIS-00-86162, and EIA-00-91474 and by Microsoft Research.
http://www.siam.org/journals/sicomp/35-1/36852.html.

[†]Computer Science Department, Technion, 32000 Haifa, Israel (mic@cs.technion.ac.il).

[‡]Computer Science Department, University of Maryland, College Park, MD 10742 (hjs@umiacs.umd.edu).

[§]Deceased. Gisli Hjaltason was a graduate of the University of Maryland and a new faculty member at the University of Waterloo. He passed away tragically and unexpectedly on June 19, 2003. We mourn the loss of this promising scientist and dear friend.

## 1. Introduction.

### 1.1. Background.

The efficient management of data in large database systems depends on grouping the data in such a way that similar data are aggregated and also stored in proximity so that they can be operated upon at the same time or at approximately the same time (see, e.g., [17]). This grouping is usually achieved by sorting the data. The rationale for sorting the data is to facilitate the presentation of the data to the user (e.g., in reports) and also to speed up query processing using sort-based algorithms such as merge-join.

Although sorting has traditionally been applied to one-dimensional data, it is also applicable to data of higher dimensions. This data can consist of points in a higher dimensional space or of spatial objects that span the higher dimensional space (e.g., lines, regions, surfaces, volumes, etc.). In the case of spatial data in more than one dimension, which is the focus of this paper, the result of applying conventional sorting techniques does not always lead to simpler algorithms. For example, suppose that the data are sorted with respect to a particular reference point (e.g., all U.S. cities, represented as points, are sorted with respect to their distance from Chicago). In this case, if we wish to obtain the points in the order of their distance from another point (e.g., with respect to their distance from Omaha), then the sorting process will, in general, have to be reapplied to the entire set of data. The problem is that the data were sorted in an explicit manner. Instead, we need methods that provide an implicit ordering. Examples of such techniques are called *bucketing methods* (see, e.g., [43, 45]). In this case, the data are sorted on the basis of the space that they occupy and are grouped into cells (i.e., buckets) of a finite capacity.

There are two principal methods for sorting spatial data. The first makes use of an object hierarchy. It is based on propagating the space occupied by groups of the data objects up through the hierarchy (e.g., members of the R-tree family [3, 18]). We do not deal with this method in this paper. The second is based on a decomposition of the space occupied by the data into disjoint cells which are aggregated into larger cells (e.g., members of the quadtree family [44, 43, 45]). The decomposition can be either tree-based or trie-based. The distinction is that the former is applied to the values of the data (e.g., a binary search tree), while the latter makes use of the digits (termed a *trie* [5, 16, 28]) that comprise the domain of the values of the data. Data structures that make use of the latter in one dimension are also known as *digital trees* [28].

Our data consists mainly of line segments in two-dimensional space. Our focus is on using tries to sort the line segments with respect to the space that they occupy. We use tries because they result in partitioning different data sets at the same positions, thereby making it very easy and efficient to use merge-join query processing algorithms. Since the space is two-dimensional, the trie is formed by interleaving the bits corresponding to the binary representation of the $x$ and $y$ coordinates of the underlying space. Two similar, yet still different, trie-based data structures may be created depending on whether we test one bit at each iteration (a *k-d trie* [15]) or two bits at each iteration (a *quadtrie* [21, 43, 45]). In this paper, we focus on quadtries for collections of line segments. Unfortunately, the representations that make use of quadtries have been traditionally referred to as *quadtree* variants (see, e.g., [26, 44, 43, 45]). In our discussion, all quadtrees are based on tries and we precede the term *quadtree* with an appropriate qualifier whenever there is a potential for confusion. Thus the quadtrees that we discuss are distinct from those based on multidimensional binary search trees that are used for points (see, e.g., [13, 14]).

Variants of quadtree structures have been used for many different spatial objects including points, regions, lines, rectangles, surfaces, volumes, etc. Algorithms using them generally have good average execution times while maintaining a relatively compact representation [44, 43, 45]. To use these data structures in a database application, we must be able to predict their size. The most obvious advantage of such a capability is that it enables us to determine how much space will be required to store different data sets and to choose between different data structures from an efficiency standpoint. It may also be of use at query evaluation time to aid the estimation of the cost of a particular query execution plan (i.e., processing strategy) to be used by a query optimizer. For example, suppose that we are using a filter-and-refine strategy [6] for processing a window query. In particular, suppose further that we have a method of estimating the number of data structure blocks that intersect the window based on the window's size (see, e.g., [40]). Our results could be used in a reverse sense to estimate the number of objects (i.e., line segments in our case) that intersect these blocks. This could serve as a measure of the cost of the refinement step, which must subsequently determine which of the lines actually intersect the query window. Continuing the filter-and-refine query processing strategy, suppose that we are using the histogram method (see, e.g., [29, 31, 34]) for estimating the number of objects that intersect the query window. We can now plug this information into our results to estimate the number of data structure blocks that intersect the window. This is a good measure of the cost of the filter step, i.e., the I/O (Input/Output) cost for the spatial data structure. An alternative factor in measuring performance when data is disk-based is to examine how the various data structure blocks are declustered on various disks (see, e.g., [33]), but this is beyond the scope of this paper and is not discussed further here.

**1.2. Related work.** Traditional worst-case analysis is often inappropriate because the worst case tends to be both very bad and highly improbable. Thus most approaches to the analysis of hierarchical data structures have been statistical in nature.

A number of statistical approaches have been tried. The most common uses a uniform distribution in the underlying space (see, e.g., [2, 11]). An alternative is to use a nonuniform distribution. Some techniques that have been used include the Gaussian distribution [36] as well as the clustering of uniformly distributed points [38] or even predetermined shapes [3]. Another approach uses a fractal distribution [9] that has the advantage of exhibiting self-similarity, which means that portions of a part of the data set are statistically similar to the entire data set. The key to this approach is to compute a fractal dimension for a particular point data set and then use it in a query optimizer.

The methods described above are for point data sets.[1] In this paper, we are interested in data where every object has nonzero size (e.g., collections of lines, regions, and so on, instead of being restricted to collections of points). Tamminen [54] considers the performance of quadtrees and bintrees under the assumption that the image consists of a single random line treated as a region, and he analyzes the number of nodes in them using geometric probability. Shaffer, Juvvadi, and Heath [52] follow Tamminen's approach and use a local straight line model to perform an analysis that yields the relative (rather than absolute) storage requirements of the region quadtree and bintree data structures. Other works on region data include Dyer [8], Shaffer [51], and

[1]The fractal model has been applied to points derived from a collection of line segments in the sense that the points corresponded to intersections of line segments [9]. This was used to predict the effective occupancy of nodes in an R-tree that stores point data.

Faloutsos, Jagadish, and Manolopoulos [10], Mathieu, Puech, and Yahia [30], Puech and Yahia [42], and Vassilakopoulos and Manolopoulos [56], and these investigate the size of quadtree representations of region data and study some other related questions using some assumptions on the branching probabilities of nodes in the tree. Nelson and Samet [35, 36, 37] consider the distributions of node occupancies in hierarchical geometric data structures that store a variable number of geometric data items per node, which include points and lines. However, the methods of Nelson and Samet have much wider applicability. This approach is similar to hashing [28], where each node acts like a bucket.

Although these approaches sometimes lead to remarkable agreement between theory and simulation (see, e.g., [1, 36]), they have a common drawback. The explicit model of the image on which the statistical analysis is performed is either exceedingly simple or is not given at all. When the model is not given, the model must be implied from other assumptions. In particular, an assumption is made on the splitting probability in the data structure (see, e.g., [1, 30, 36, 42]), which implies the existence of some *implicit* model on the data. However, when we ask what kind of data (real or contrived) fit this model, the only possible answer is a circular one that says that the data give rise to these probabilities. Unfortunately, there is no explicit indication of whether there exists some image model associated with these splitting probabilities. Thus the connection between the analysis and the performance with real image data is not clear. In contrast, our approach, as described below, is to use an *explicit* nontrivial random image model and to then show that data can be generated corresponding to this model, which also fits the analysis. Note that we are not claiming that the data we generate correspond exactly to typical images, although we deal with this issue as well. In this sense our approach is complementary to the work of Flajolet and Puech [15], who analyzed the partial match query time for hierarchical data structures, while we analyze their storage requirements. Unlike their data, which consist of random points in a high-dimensional space whose coordinate values are drawn from a uniform distribution, our data consist of randomly drawn lines. It is important to note that a line is a qualitatively different data type than a point, as the action of every line on the structure is not local.

An alternative nonstatistical approach was applied by Hunter [22] and Steiglitz [23] to show that, for a polygon of perimeter $l$, the size (i.e., the number of nodes) of the corresponding MX quadtree (a variant of the region quadtree described in more detail in section 2) is $O(l)$. This classic result, although derived for simple polygons, has been observed to be sufficiently general to be useful for predicting the performance of a number of algorithms for different images represented by a region quadtree [48].

As we pointed out above, in this paper we investigate the use of a random image model consisting of $M$ randomly drawn lines. Unlike Tamminen's approach [54], which considers a single random line, here we treat the much more general and complicated situation of an arbitrary number of lines. We use geometric probability to analyze four variants of the quadtree that can be built for data that obey this model by determining the expected number of nodes in each variant. These variants are the MX quadtree [23, 43, 45], the PM quadtree [49], the PMR quadtree [35, 36, 37], and a new variant of the PMR quadtree representation, which we call a Bucket PMR quadtree (see also [19, 20]).

**1.3. Contributions.** The analysis that we provide is important for two reasons. First, it allows for a meaningful, quantitative, and analytic comparison of a number of different options for representing linear spatial data, and it provides tools for choosing between these options in a way which is neither experimental nor domain dependent. The second reason is even more practical: we found that we could actually predict the storage requirements of these representation options by specifying an equivalent random lines data set, with some equivalent number of lines, and use the proposed analytic expressions for predicting its size.

In particular, our analysis shows that for images of the same complexity, the PMR quadtree and the Bucket PMR quadtree for sufficiently high values of the splitting threshold (i.e., $\geq 4$) are the most efficient in the sense that they require the least storage. The PM quadtree follows, and the MX quadtree requires the largest amount of storage. This qualitative ordering verifies experimental results obtained in the past [22, 23, 35, 49] and agrees with the extensive experimentation that we have carried out. This verification, along with its accompanying theoretical justification, is one of the contributions of our research.

The agreement between the results of our analysis and the data was not surprising in the case of the MX quadtree because it confirmed previous results (i.e., [22, 23]). However, in the case of the PMR and Bucket PMR quadtrees for sufficiently high values of the splitting threshold (i.e., $\geq 4$) our analysis breaks new ground because we are able to derive theoretically and verify experimentally for both random data and real map data that the number of nodes is asymptotically proportional to the number of line segments. This is quite significant, as it enables us to predict the number of nodes required by this representation, and, most importantly, to show that it is independent of the maximum depth of the tree.

It is important to note that we do not claim that our proposed random image model yields data instances which are visually similar to what appears in realistic geometric applications, such as road networks. Nevertheless, we do show that the analysis can be interpreted in terms of the geometric properties of the image, such as line length and the number of intersections between lines. With this interpretation, the predictions, derived for random images, may be applied to real data by measuring the relevant geometric property and using it to specify equivalent random images. Testing the predictions on a real set of maps yielded relatively accurate predictions of the storage required for the maps.

Although our analysis is for a particular data type (i.e., collections of line segments) and data structures, we believe that it has wider applicability. In particular, the geometric probability approach could be extended for data types other than line segments (e.g., points, polygons, surfaces, solids, etc.). Furthermore, the random image model can be used in a statistical analysis of other trie-based spatial data structures.

The rest of this paper is organized as follows. Section 2 gives a brief overview of the quadtree representations of collections of line segments, including the definitions of the four variants that we analyze. Section 3 presents the random image model and reviews some necessary results from geometric probability. Section 4 contains a statistical analysis using the model and the results of its application to each of the aforementioned quadtree variants. It also contains the results of some experiments with instances of the random image model. Section 5 describes the application of the analysis to predict the storage requirements for real data and presents results of extensive experiments that support its validity. Section 6 contains concluding remarks and gives some directions for future research.

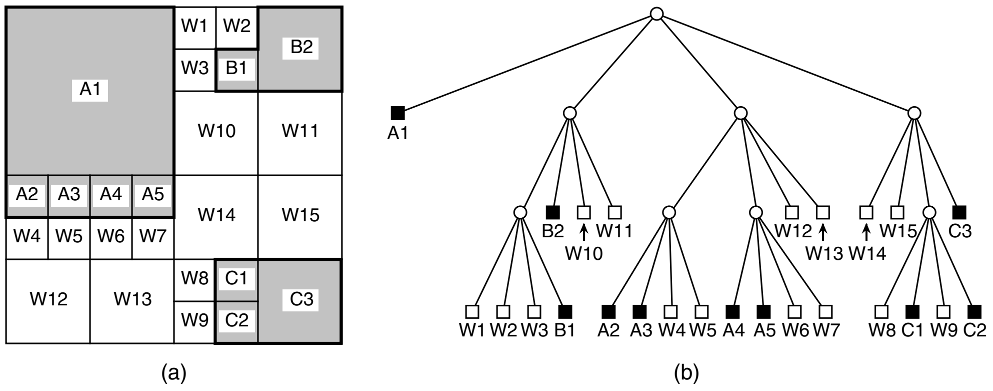
A1
W1
W2
W3
B1
B2
W10
W11
A2
A3
A4
A5
W14
W15
W4
W5
W6
W7
W12
W13
W8
C1
C3
W9
C2
(a)
W1 W2 W3 B1 A2 A3 W4 W5 A4 A5 W6 W7 W8 C1 W9 C2
B2 W10 W11 W12 W13 W14 W15 C3
(b)

FIG. 1. (a) *Block decomposition and* (b) *its tree representation for the region quadtree corresponding to a collection of three regions* A*,* B*, and* C*.*

**2. Overview of quadtree representations of collections of line segments.** A quadtree is a hierarchical variable resolution data structure based on the recursive partitioning of the two-dimensional plane into quadrants. It can be viewed as a 4-ary tree, where each node represents a region in the plane called a block, and the children of each node represent a partition of that region into four parts. This scheme is useful for representing geometric data at a variable resolution. The most commonly known version of the quadtree is the *region quadtree* [26]. It is used for the representation of planar regions. In this case, for two-dimensional data, the environment containing the regions is recursively decomposed into four rectangular congruent blocks until each block either is completely occupied by a region or is empty (such a decomposition process is termed *regular*). For example, Figure 1(a) is the block decomposition for the region quadtree corresponding to three regions A, B, and C. Notice that, in this case, all the blocks are square, have sides whose size is a power of 2, and are located at specific positions.

The traditional, and most natural, access structure for a region quadtree corresponding to a two-dimensional image is a tree with a fanout of 4 (see, e.g., Figure 1(b)). Each leaf node in the tree corresponds to a different block $b$ and contains the identity of the region associated with $b$. Each nonleaf node $f$ corresponds to a block whose volume is the union of the blocks corresponding to the four children of $f$. In this case, the tree is a containment hierarchy and closely parallels the decomposition in that they are both recursive processes and the blocks corresponding to nodes at different depths of the tree are similar in shape.

Quadtree variants exist for representing other spatial data types than just planar regions. For example, they are used to represent collections of points [12, 46], collections of line segments [35, 36, 37, 49], as well as more complicated objects (e.g., rectangles [25]). Generalizations of the quadtree to three and higher dimensions (e.g., octrees [22, 24, 32] and bintrees [27, 48, 55]) have also been investigated. These generalizations have many of the same basic properties.

The different variants of the quadtree data structure can be subdivided into two categories: those based on a regular decomposition of space using predefined positions for the partition lines (i.e., trie-based), and those in which the positions of the partition lines are determined explicitly by the data as they are inserted into the data structure (i.e., tree-based or data-based). In most cases, use of regular decomposition indicates

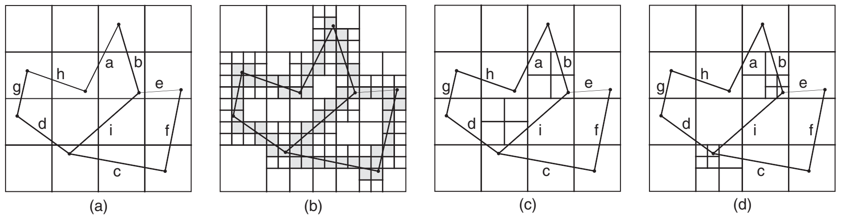

FIG. 2. (a) *Collection of line segments in a* $4 \times 4$ *grid,* (b) *its MX quadtree,* (c) *its PM quadtree, and* (d) *its Bucket PMR quadtree with a bucket capacity of* 2. *Only the resulting decomposition of the underlying space into blocks is shown. The corresponding tree structure that acts as an access structure to ensure logarithmic access times is not shown.*

that the shape of the resulting data structure is independent of the order in which the data are inserted into the structure when building it. This is not the case for data-based decompositions. Interestingly, for most applications, regular decomposition works at least as well as data-based decomposition. Moreover, regular decomposition is easier to implement and analyze. At times, a distinction between different variants of the quadtree data structure is also made on the basis of whether or not there is a predefined maximum depth (denoted by $N$).

In this paper we only consider quadtree representations of collections of line segments. We restrict ourselves to quadtrees based on a regular decomposition. In the rest of this section we review the different quadtree variants that we study. They differ in the condition that is used to determine when a quadtree block should be decomposed—this condition is termed a *splitting rule*.

The simplest quadtree representation is the *MX quadtree*, which assumes that the underlying domain of the data is a $2^N \times 2^N$ grid. The MX quadtree is built by digitizing the line segments and labeling each unit-sized cell (i.e., pixel) through which it passes as being of type `boundary`. The remaining pixels are marked `WHITE` and are merged, if possible, into larger and larger quadtree blocks, as done in the region quadtree. Figure 2(b) is the MX quadtree for the collection of line segment objects in Figure 2(a). A drawback of the MX quadtree is that it associates a thickness with a line. Also, it is difficult to detect the presence of a vertex whenever five or more line segments meet. The above definition of the MX quadtree is given in a bottom-up manner. We can also define it in a top-down manner. In particular, we start with one block corresponding to the entire $2^N \times 2^N$ space and recursively decompose it into four blocks, halting the decomposition when a block is empty or is of size $1 \times 1$ (i.e., the block corresponds to a pixel).

The PM quadtree is a refinement of the MX quadtree that is motivated by the observation that the number of blocks in the decomposition can be reduced by terminating the subdivision whenever a line segment passes through a block completely (i.e., it enters and exits the block rather than starting or terminating in the block). Nevertheless, even when this observation is used, the resulting structure still has full decomposition at each vertex or endpoint of a line segment. To avoid this situation, we further modify the above top-down MX quadtree definition so that decomposition takes place as long as more than one line segment passes through the block, unless all of the line segments that pass through the block are incident at the same vertex, which is also required to be in the same block. In addition, the decomposition is also halted whenever a $1 \times 1$ block is encountered. The PM quadtree is the structure that results when these additional halting conditions are imposed on the top-down definition of the MX quadtree. The fact that the PM quadtree is defined in a top-down manner means that each block is maximal. It should be clear that each block at a depth less than the maximum depth contains at most one vertex. For example, Figure 2(c) is the PM quadtree corresponding to the collection of line segment objects in Figure 2(a).

The PM quadtree is vertex-based in the sense that the vertices play a role in determining when the decomposition process stops. In particular, the decomposition process halts when there is more than one line segment in a block, provided that all of the line segments are incident at the same vertex in the block. An alternative is to use an edge-based representation, where a block is split whenever there are more than $q$ line segments passing through it. Thus the blocks serve as buckets with a capacity $q$. This is known as a *Bucket PMR quadtree*. The drawback of this method is that, whenever more than $q$ line segments are incident at a vertex $v$, the block containing $v$ will be decomposed until reaching the maximum depth $N$, which corresponds to a $1 \times 1$ block. For example, Figure 2(d) is the Bucket PMR quadtree corresponding to the collection of line segment objects in Figure 2(a) when using a bucket capacity of $q = 2$ and a maximum depth $N = 4$.

As pointed out above, the drawback of the Bucket PMR quadtree is that if the number of line segments incident at a vertex exceeds the bucket capacity, then the decomposition in the neighborhood of the vertex will not halt unless we reach the maximum allowable depth $N$. This problem is resolved by the *PMR quadtree*, which is similar to a Bucket PMR quadtree with the difference that, in the PMR quadtree, a block is decomposed once, and only once, if the insertion causes it to have more than $q$ line segments. Therefore, in the PMR quadtree, $q$ serves as a *splitting threshold*, which is quite different than a bucket capacity, which is its role in the Bucket PMR quadtree. There is no maximum depth in the PMR quadtree.

The PMR quadtree is constructed by inserting the line segments one by one into an initially empty structure consisting of one block. Each line segment is inserted into all the blocks that it intersects or occupies in its entirety. During this process, the occupancy of each block that is intersected by the line segment is checked to see if the insertion causes it to exceed the splitting threshold. If the splitting threshold is exceeded, the block is split *once*, and only once, into four blocks of equal size.

Figure 3(e) is the PMR quadtree for the collection of line segment objects in Figure 2(a) with a splitting threshold of $q = 2$. The nine line segments, labeled a–i, are inserted in alphabetic order. It should be clear that the shape of the PMR quadtree for a given collection of line segments is not unique; instead, it depends on the order in which the line segments are inserted into it. In contrast, the shapes of the MX, PM, and Bucket PMR quadtrees are unique. Figure 3(a)–(e) shows some of the steps in the process of building the PMR quadtree of Figure 3(e) with a splitting threshold of 2. In each part of Figure 3(a)–(e), the line segment that caused the subdivision is denoted by a thick line, while the gray regions indicate the blocks where a subdivision has taken place.

The insertion of line segments c, e, g, h, and i causes the subdivisions in parts (a), (b), (c), (d), and (e), respectively, of Figure 3. The insertion of line segment i causes three blocks to be subdivided (i.e., the SE block in the SW quadrant, the SE quadrant, and the SW block in the NE quadrant). The final result is shown in Figure 3(e). Note the difference from the PM quadtree in Figure 2(c)—that is, the NE block of the SW quadrant is decomposed in the PM quadtree, while the SE block of the SW quadrant is not decomposed in the PM quadtree. We also observe that, unlike the Bucket PMR quadtree in Figure 2(d), we did not have to split to the maximum

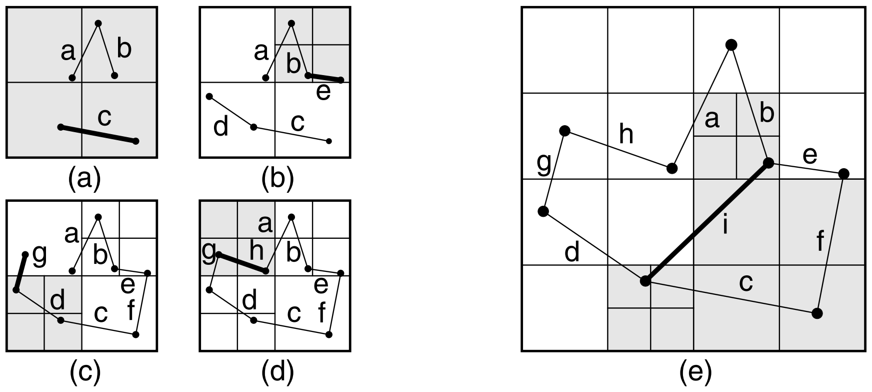

FIG. 3. *PMR quadtree with a splitting threshold of* 2 *for the collection of line segments of Figure* 2(a). (a)–(e) *illustrate snapshots of the construction process with the final PMR quadtree given in* (e).

depth in the neighborhood of the vertices, where line segments c, d, i, as well as b, e, i, meet.

It should be clear that the number of line segments in a PMR quadtree block can exceed the value of the splitting threshold $q$. A value of four for $q$ is usually sufficient to store collections of line segments efficiently, as it implies that junctions of two, three, and four line segments (which are common in maps of roads and rivers, etc.) do not cause a split. Of course, there are situations in which more than four line segments will meet at a vertex. However, we assume that such situations are rare. Note that, at times, we want to express the dependence of the Bucket PMR quadtree and the PMR quadtree on $q$ explicitly, in which case we use the term *bucket* $PMR_q$ *quadtree* and $PMR_q$ *quadtree*, respectively, to describe the structure.

Note that, in the general case, we may have a collection of line segments that intersect, while the intersection point is not a vertex. Such a situation can result when we are representing a nonplanar graph. As we shall see later, this is not an issue for the line segment arrangements that are generated by our random image model (described in section 3).

**3. Random image models.** In the first part of this paper (sections 3 and 4) we assumed that the quadtree variants that are discussed represent geometric structures, which are instances of a random process described as follows. Observe that the line $L(\rho, \theta)$ consists of the points $(x, y)$ satisfying the relation

$$L(\rho, \theta) = \{(x, y)|x \cos \theta + y \sin \theta = \rho\}.$$

The line $L(\rho, \theta)$ is perpendicular to the vector $(\cos\theta, \sin\theta)$ (see Figure 4(a)). Although the position of every particular line, $L(\rho, \theta)$, naturally depends on the origin and orientation of the coordinate system, we shall soon see that the probability of every random line, drawn according to our model, does not. Therefore, the origin and orientation of the coordinate system relative to the image does not make a difference. The arbitrarily chosen location of the coordinate system in Figure 4(a) illustrates this invariance property. For the rectangular region $R$,

$$R = \{(x, y)|\ |x|, |y| < 2^{N-1}\},$$

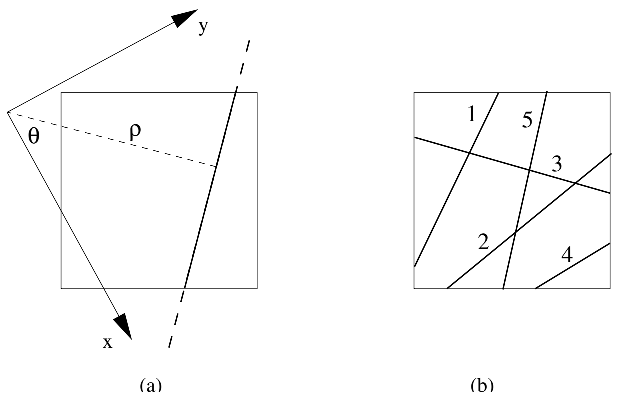

FIG. 4. *The random image model:* (a) *The process of generating a random line (note that the coordinate system is arbitrary with respect to the image).* (b) *A typical instance of the random image constructed for $M = 5$ independently drawn lines labeled* 1–5.

let $T$ be the parameter set

$$T = \{(\rho, \theta) | L(\rho, \theta) \cap R \neq \emptyset\},$$

which includes all the parameter pairs $(\rho, \theta)$ that represent lines intersecting with $R$. Let

$$p(\rho, \theta) = \begin{cases} \frac{1}{|T|} & (\rho, \theta) \in T, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

be a probability density function, where

$$|T| = \int_T d\rho d\theta.$$

This distribution, called the *uniform density distribution*, is the only one which ensures that the probability of choosing a particular random line is independent of the coordinate system in which $\rho$ and $\theta$ are defined (i.e., it is independent of the translation or rotation of the coordinate system [50]). Therefore, it is the natural density function to specify when modeling collections of random lines. As an illustration, see Figure 4(b), where an instance of the random image containing five lines labeled 1–5 is described.

Every instance of our random image model is a $2^N \times 2^N$ image with $M$ random lines that intersect it and which are chosen independently according to the density function (1). The continuous uniform density distribution implies that three lines intersect at the same point with probability zero. This follows from the observation that two intersecting lines define a point, say $(x_0, y_0)$, which can be regarded as prespecified for the third line. The parameters of every line which intersects this point must be in the set $\{(\rho, \theta) | x_0 \cos\theta + y_0 \sin\theta = \rho\}$, the measure of which is zero (i.e., a one-dimensional quantity) with respect to the measure of $T$ (i.e., a two-dimensional quantity). Therefore, after drawing a finite number of random lines, the probability that any three of them will intersect $(x_0, y_0)$ is zero. This property is usually satisfied for real spatial data such as road maps, as intersections of four or more line segments are rare (i.e., junctions of four or more roads).

Each of the line segments clipped from a random infinite line by the boundary of the image is subdivided further into smaller line segments by its intersection with other infinite lines so that, eventually, no line segment crosses another line segment except at the endpoints of the line segments.

Thus the data represented by the various quadtrees (in the first, analytical, part of the paper) are a collection of random line segments specified as a set of $M$ random infinite lines. Note that the infinite lines are not of interest by themselves but are useful for creating the distribution of line segments which we analyze.

We do not claim that our proposed random image model will enable us to generate data that correlate with what appears in realistic geometric applications, such as road networks. Finding such a correlation is unlikely, as realistic geometric data do not consist of lines whose endpoints lie on the image boundary. Nevertheless, the analysis can be interpreted, as we shall see later, in terms of the geometric properties of the image, such as line length and the number of intersections between lines. This allows us to apply its results to real data with similar geometric properties. Most important, the analysis provides us a means to justify claims about the relative qualitative behavior of the different data structures.

Before starting the analysis, we first present three results from geometric probability which form the basis of our results [50].

**Geometric probability theorem 1 (Theorem GP1).** Let $C_1$ be a convex planar set included in the convex planar set $C \subset R$. Let $L_1$ and $L$ be the perimeters of $C_1$ and $C$, respectively. Let $l$ be a random line chosen using the uniform density distribution given by (1). Therefore, the probability that a line $l$ passing through $C$ also passes through $C_1$ is

$$p\{l \cap C_1 \neq \emptyset | l \cap C \neq \emptyset\} = \frac{L_1}{L}.$$

**Geometric probability theorem 2 (Theorem GP2).** Let $C \subset R$ be a convex planar set with area $A$ and perimeter $L$. Let $l$ be a random line chosen using the uniform density distribution given by (1). Suppose that $l$ intersects with $C$ and creates a chord $H$ with length $|H|$. Then the expected length of $H$ is

$$E[\,|H|\,] = \frac{\pi A}{L}.$$

**Geometric probability theorem 3 (Theorem GP3).** Let $C \subset R$ be a convex planar set with area $A$ and perimeter $L$. Let $l_1$ and $l_2$ be two random lines, independently chosen using the uniform density distribution (1). If both lines $l_1$ and $l_2$ intersect with $C$, then the probability that $l_1$ intersects with $l_2$ inside $C$ is

$$p\{(l_1 \cap l_2) \cap C \neq \emptyset | l_1 \cap C \neq \emptyset, l_2 \cap C \neq \emptyset\} = \frac{2\pi A}{L^2}.$$

**4. Statistical analysis of quadtree representations of collections of line segments.** An image as defined in section 3 is an instance of a random event. It follows that its hierarchical representation, using one of the quadtree variants, is also a random event. Moreover, the existence of a node in the tree, or its being a leaf, is a random event. Let $v$ be a node of the tree, at depth $d$, corresponding to some particular cell. Both the existence of $v$ in the tree and its potential split are random events corresponding to the particular arrangement of random lines. Let $P_d$ be the probability that both of these events happen. Recall that the distribution of the random lines is independent of the coordinate system translation, thereby implying that $P_d$ depends only on the depth.

Let $v_e$ denote the event that the node $v$ exists. Let $v_s$ denote the event that the splitting condition holds for the block corresponding to node $v$. Letting $\mathrm{Prob}(v_e, v_s)$ be the probability of the joint event that both node $v$ at depth $d$ exists ($v_e$) and that the splitting condition is satisfied for the square region corresponding to node $v$ ($v_s$), we have that

$$P_d = \mathrm{Prob}(v_e, v_s) = \mathrm{Prob}(v_s)\mathrm{Prob}(v_e|v_s). \tag{2}$$

$\mathrm{Prob}(v_s)$ is the probability that the splitting condition is satisfied for a particular square region corresponding to a node $v$ (at depth $d$), while $\mathrm{Prob}(v_e|v_s)$ is the conditional probability that node $v$ at depth $d$ exists, given that the splitting condition holds for the square region corresponding to $v$. Note that both $v_s$ and $v_e$ depend on the depth $d$ as well.[2] For example, in the MX quadtree, $\mathrm{Prob}(v_s)$ is the probability that at least one line passes through this region. The node $v$ exists if every member in the sequence of its recursive parents splits as well. For the MX quadtree, this always happens (if $v_s$ is true) because there is at least one line passing through all of them: the line which passes through $v$ and satisfies its splitting condition. Therefore, for the MX quadtree, $\mathrm{Prob}(v_e|v_s) = 1$ and $P_d = \mathrm{Prob}(v_s)$. We shall see later that this relation is not necessarily satisfied for every tree structure (e.g., the PM quadtree as discussed in subsection 4.2.1).

Let $S$ be the total number of nodes in the tree. Every nonleaf node at depth $d-1$ contributes 4 nodes at depth $d$. The maximal number of nodes at depth $d-1$ is $4^{d-1}$, implying that the expected size of the tree is

$$E[S] = 1 + \sum_{d=1}^{N} 4^d \cdot P_{d-1}. \tag{3}$$

Note that while the splitting events associated with, say, neighboring nodes, are definitely dependent events, this does not effect the calculation of the expected value [41].

Equation (3), which gives the expected number of nodes in the tree, serves as the basis for our analysis. In the following subsections we focus on splitting rules for each of the quadtree variants discussed in section 2. For each rule, we derive the corresponding splitting probabilities $P_d$ and then use (3) to calculate the expected size of the data structure.

**4.1. MX quadtree.** An MX quadtree represents a collection of line segments on the plane by partitioning the plane into square blocks using the splitting rule that says a block is split if both the depth of its corresponding node is less than $N$ and if the block contains at least one line segment. If the block does not contain a line segment, then it is not subdivided further and its corresponding node is a leaf. Otherwise, it is subdivided and its corresponding node has four children (see Figure 2(b)).

[2]This "backward" decomposition was preferred over the "more natural" $\mathrm{Prob}(v_e)\mathrm{Prob}(v_s|v_e)$ because it isolates the event $v_s$, which is "local" and depends only on the configuration of the lines intersecting the block corresponding to $v$. In contrast, $v_e$ is a more complex event, depending on the existence of all of the ancestors of $v$.

**4.1.1. Analysis.** In order to compute the probabilities $P_d$, we use the following argument. A node (denoted $v$) at depth $d$ corresponds to a $2^{N-d} \times 2^{N-d}$ square (denoted $v$ as well). Theorem GP1 implies that a particular random line passes through this region with probability

$$p_d = \frac{4 \cdot 2^{N-d}}{4 \cdot 2^N} = \left(\frac{1}{2}\right)^d .$$

The probability that exactly $k$ out of $M$ lines pass through this region is

$$p_{d,k} = \binom{M}{k} \left(\frac{1}{2}\right)^{dk} \left[1 - \left(\frac{1}{2}\right)^d\right]^{M-k} . \tag{4}$$

The probability that this region corresponds to a nonleaf node is

$$P_d = \text{Prob}(v_s)\text{Prob}(v_e|v_s) = \text{Prob}(v_s) = 1 - p_{d,0} = 1 - \left[1 - \left(\frac{1}{2}\right)^d\right]^M , \tag{5}$$

where $\text{Prob}(v_s)$ is the probability that one or more lines pass through this region, thereby satisfying the splitting condition. As mentioned above, for the MX quadtree, $\text{Prob}(v_e|v_s)$ is always 1 because there is at least one line passing through all the regions corresponding to the recursive parents of this region: the line which passes through $v$ and satisfies its splitting condition.

Inserting (5) into (3), we get

$$E[S] = 1 + \sum_{d=1}^{N} 4^d \cdot P_{d-1} = \sum_{d=1}^{N} 4^d \left[1 - \left[1 - \left(\frac{1}{2}\right)^{d-1}\right]^M\right]. \tag{6}$$

It is difficult to derive closed forms of sums of this nature. To our knowledge, no relevant solutions exist in the literature. Furthermore, we tried, without success, to evaluate it using various symbolic equation solvers, and consulted their developers as well. Therefore, as we are primarily interested in comparing the asymptotic behavior of the storage requirements of the various data structures, we resort to closed form upper and lower bounds for $E[S]$—that is, the expected number of nodes in the tree. However, for practical use of this estimate, we suggest inserting the known parameters (i.e., $M$ and $N$) into the sum (6) and evaluating it numerically. These comments are also applicable in the analyses of the rest of the data structures (see subsections 4.2.1 and 4.3).[3]

Our technique is based on decomposing (6) into two sums $\sum_1$ and $\sum_2$ corresponding to the number of nodes at depth less than or equal to $d_0$ and all the nodes at a depth greater than $d_0$, respectively. In essence, our analysis focuses on evaluating the second sum, while the first sum is bounded by the number of nodes in the complete tree (when calculating the upper bound) or by zero (when calculating the lower bound). We find it convenient to formulate our analysis of $E[S]$ in terms of an

[3]Note that, although the form of the sum (6) intuitively calls for the use of the commonly known $1 + x \leq e^x$ inequality, it does not help here. The problem is that, in the case at hand, we want to bound (6) from above, which means that the term $\left[1 - \left(\frac{1}{2}\right)^{d-1}\right]^M$ should be bound from below, but this is not possible with this inequality.

additional parameter $\beta$ (as well as $M$ and $N$), which is defined as

$$\beta = M\left(\frac{1}{2}\right)^{d_0}. \tag{7}$$

The depth $d_0$ is chosen so that $\beta$ is less than 1. This enables us to make some assumptions leading to crucial simplifications (i.e., that certain sums converge as the index of summation gets infinitely large). The result (i.e., $E[S]$) is in terms of $M$, $N$, and $\beta$. Once the result has been obtained, the value of $d_0$ is adjusted so that the bounds on $E[S]$ are as tight as possible under the constraints that $d_0$ is an integer bounded by $N$ and that $\beta < 1$.

Decomposing $E[S]$ into two sums $\sum_1$ and $\sum_2$ yields

$$\begin{aligned} E[S] &= 1 + \sum_{d=1}^{d_0} 4^d \left[1 - \left[1 - \left(\frac{1}{2}\right)^{d-1}\right]^M\right] + \sum_{d=d_0+1}^{N} 4^d \left[1 - \left[1 - \left(\frac{1}{2}\right)^{d-1}\right]^M\right] \\ &= \textstyle\sum_1 + \sum_2. \end{aligned} \tag{8}$$

$$\textstyle\sum_1 = 1 + \displaystyle\sum_{d=1}^{d_0} 4^d \left[1 - \left[1 - \left(\frac{1}{2}\right)^{d-1}\right]^M\right] \le \sum_{d=0}^{d_0} 4^d \cdot 1 \approx \frac{4^{d_0}}{1 - \frac{1}{4}} \approx \frac{4}{3}\frac{M^2}{\beta^2}. \tag{9}$$

Note that what we have done is decompose $E[S]$ into two parts, $\sum_1$ and $\sum_2$, where $\sum_1$ corresponds to a complete tree at a depth less than or equal to $d_0$. Taking the binomial expansion of $\sum_2$, we get

$$\begin{aligned} \textstyle\sum_2 &= \sum_{d=d_0+1}^{N} 4^d \left[1 - \sum_{k=0}^{M} \left(\frac{1}{2}\right)^{kd-k} \binom{M}{k} (-1)^k\right] \\ &= \sum_{d=d_0+1}^{N} \sum_{k=1}^{M} 2^{2d} \left(\frac{1}{2}\right)^{kd-k} \binom{M}{k} (-1)^{k-1}. \end{aligned} \tag{10}$$

Changing the order of summation and separating the $k = 1$ ($\sum_3$), $k = 2$ ($\sum_4$), and $k > 2$ ($\sum_5$) cases, we get

$$\textstyle\sum_2 = \sum_3 + \sum_4 + \sum_5, \tag{11}$$

$$\begin{aligned} \textstyle\sum_3 &= \sum_{d=d_0+1}^{N} 2^{2d} \left(\frac{1}{2}\right)^{d-1} \binom{M}{1} \\ &= \sum_{d=d_0+1}^{N} 2^d \cdot 2 \cdot M = 4M(2^N - 2^{d_0}) = 4M \cdot 2^N - \frac{4M^2}{\beta}, \end{aligned} \tag{12}$$

$$\textstyle\sum_4 = -\displaystyle\sum_{d=d_0+1}^{N} 2^{2d} \left(\frac{1}{2}\right)^{2d-2} \binom{M}{2} = -\sum_{d=d_0+1}^{N} 4\binom{M}{2} = -2M(M-1)(N-d_0), \tag{13}$$

(14)

$$\begin{aligned}
\textstyle\sum_5 &= \sum_{d=d_0+1}^{N} \sum_{k=3}^{M} 2^k \binom{M}{k} \left(\frac{1}{2}\right)^{(k-2)d} (-1)^{k-1} \\
&\le \sum_{k=3}^{M} \sum_{d=d_0+1}^{N} 2^k \binom{M}{k} \left(\frac{1}{2}\right)^{(k-2)d} \\
&\approx \sum_{k=3}^{M} 2^k \binom{M}{k} \frac{(\frac{1}{2})^{d_0(k-2)}(\frac{1}{2})^{k-2}}{1-(\frac{1}{2})^{k-2}} \\
&= \sum_{k=3}^{M} 4 \binom{M}{k} \left[\frac{\beta}{M}\right]^{k-2} \frac{1}{1-(\frac{1}{2})^{k-2}} \\
&= \sum_{k=3}^{M} 4 \cdot \frac{M\cdot(M-1)\cdot(M-2)\cdots(M-k+1)}{k!} \frac{\beta^{k-2}}{M^{k-2}} \frac{1}{1-(\frac{1}{2})^{k-2}} \le \frac{4}{3} M^2 \frac{\beta}{1-\beta}.
\end{aligned}$$

Note that all approximations performed while calculating $\sum_1$ and $\sum_5$ are also upper bounds. We continue by summing all contributions, which are partly expected values and partly upper bounds for expected values, to get

(15)

$$\begin{aligned}
E[S] &= \textstyle\sum_1 + \sum_3 + \sum_4 + \sum_5 \\
&\le 4\cdot M \cdot 2^N - 2\cdot M(M-1)\cdot N + M^2 \left[-\frac{4}{\beta} + \frac{4}{3}\frac{1}{\beta^2} + \frac{4}{3}\frac{\beta}{1-\beta} + 2\log_2 \frac{M}{\beta}\right].
\end{aligned}$$

Figure 5 shows the value of the sum estimate given by (6) (second curve from the top) as well as the upper bound (upper curve) given by (15) as a function of $M$ at a maximal depth of $N = 10$. Recall that the upper bounds in the figure are minimal in the sense that, for each value of $M$, upper bounds were calculated for every possible value of $d_0$ (subject to the constraint $\beta < 1$) and the minimal (tightest) upper bound was taken.

**4.1.2. Interpretation.** Asymptotically, the dominant contribution to the number of nodes comes from the first term in (15), which may be transformed into a more familiar form using the GP2. Letting $L_i$ be the length of the $i$th line in our geometric structure, the expected total length $L$ of all lines is

$$E[L] = \sum_{i=1}^{M} E[L_i] = M \cdot \pi \frac{(2^N)^2}{4\cdot 2^N} = \frac{\pi}{4}\cdot M \cdot 2^N. \tag{16}$$

Substituting (16) into the first term of (15), we get

$$E[S] \approx \frac{16}{\pi} E[L]. \tag{17}$$

In other words, the expected number of nodes in an MX quadtree is proportional to the total expected length of the lines, which agrees with results derived previously under different (nonprobabilistic) models [22, 23].

**4.1.3. A lower bound.** The derivation of $E[S]$ given by (9)–(15) may be used to set a lower bound on the expected number of nodes. $E[S]$ consists of the contributions

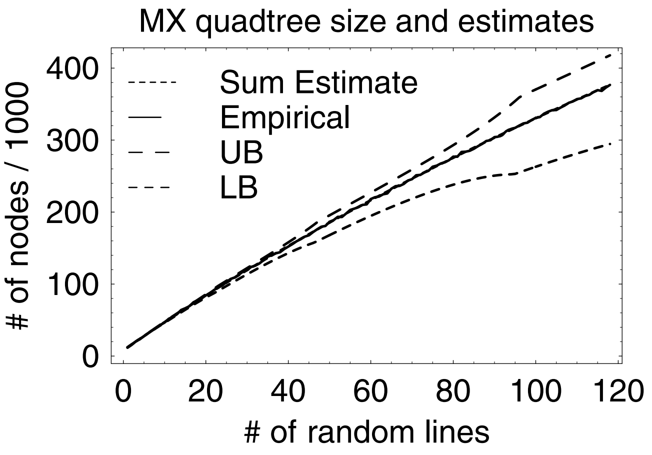

FIG. 5. *The upper bound (UB), the model prediction (Sum Estimate), the empirical estimate (Empirical), and the lower bound (LB) on the number of nodes necessary to store an MX quadtree. The empirical estimate almost coincides with the model prediction. These are the two middle curves, which actually look like one curve. The x and y axes correspond to the number of lines and nodes, respectively, for a tree of depth $N = 10$.*

of $\sum_1$, $\sum_3$, $\sum_4$, and $\sum_5$. $\sum_3$ and $\sum_4$ are exact values, $\sum_1$ is positive, thereby having a lower bound of 0, while $\sum_5$ can be easily bounded from below by $-\frac{4}{3}M^2\frac{\beta}{1-\beta}$. Thus,

$$E[S] \geq \sum_3 + \sum_4 - \frac{4}{3}M^2\frac{\beta}{1-\beta}. \tag{18}$$

Furthermore, for large $N$ satisfying $2^N \geq M \cdot N$, we have that $\sum_1$, $\sum_4$, and $\sum_5$ are small with respect to $\sum_3$. Thus the difference between the upper bound and the lower bound is small, and each of the bounds is a good approximation of $E[S]$ (see the lowest curve in Figure 5).

**4.2. PM quadtree.** Our variant of a PM quadtree represents a collection of line segments in the plane. It partitions the plane into square blocks using the splitting rule that says a block is split unless the depth of the corresponding node is $N$, or only one line passes through the block, or there is just one vertex in the block and all the lines that pass through the block meet at that vertex. Thus if the block contains a single line, or all the lines pass through a common point in the block and there is no other endpoint in the block, then the block is not subdivided further and its corresponding node is a leaf. Otherwise, it is subdivided and its corresponding node has four children (see Figure 2(c)).

**4.2.1. Analysis.** Recall from the opening remarks in section 4 that the probability that a region corresponds to a nonleaf node is $P_d = \text{Prob}(v_s)\text{Prob}(v_e|v_s)$. Consider first $\text{Prob}(v_s)$, the probability that the splitting condition is satisfied. Let $\alpha$ be the probability that two lines intersect inside a square region $Q$ given that each of these lines passes through $Q$. For a square $2^{N-d} \times 2^{N-d}$ region $(0 \leq d \leq N)$, the probability that the splitting conditions of a PM quadtree are satisfied for a node $v$ in depth $d$ may be written as

$$Prob(v_s) = 1 - p_{d,0} - p_{d,1} - \alpha \cdot p_{d,2}, \tag{19}$$

where $p_{d,k}$ is the probability that exactly $k$ of $M$ lines pass through a square region of side $2^{N-d}$. The random image model implies that three lines intersect with zero probability at a common point. Therefore, this event may be ignored, leading to the conclusion that a region is always split if three or more lines pass through it. If only two lines pass through the region, then the region is split only if the two lines do not intersect within the region. The intersection probability $\alpha$ may be inferred from Theorem GP3, which implies that

$$\alpha = \frac{2\pi A}{L^2} = \frac{2\pi(2^{N-d})^2}{(4 \cdot 2^{N-d})^2} = \frac{\pi}{8}.$$

Hence,

$$\text{(20)} \qquad \begin{aligned} \text{Prob}(v_s) = 1 &- \binom{M}{0}\left[1 - \left(\frac{1}{2}\right)^d\right]^M \\ &- \binom{M}{1}\left(\frac{1}{2}\right)^d\left[1 - \left(\frac{1}{2}\right)^d\right]^{M-1} \\ &- \frac{\pi}{8}\binom{M}{2}\left(\frac{1}{2}\right)^{2d}\left[1 - \left(\frac{1}{2}\right)^d\right]^{M-2}. \end{aligned}$$

In contrast to the other quadtree variants considered in this paper, the probability $Prob(v_e|v_s)$ is not 1 here. In particular, for a PM quadtree, it is possible that the splitting condition is satisfied for some node (region) at depth $d$ but not for its parent node. For the random image model, this arises only in one case, which is when exactly two lines pass through both the node $v$ and its parent node, with their intersection point lying inside the region corresponding to the parent node and outside the region corresponding to the node $v$. In this case, we have the anomalous situation that, although the node $v$ does not exist, its corresponding region would be split (if it did exist).

Note that $P_d \leq Prob(v_s)$. We start our analysis by treating this bound, denoted by $P'_d$, as the splitting probability itself. The difference between the following analysis and the one carried out for the MX quadtree is that here the sum itself is an upper bound as well. Later, we also derive a tighter bound, based on an asymptotic approximation.

To obtain a bounded expressed in a simpler way, let us once again, as in the MX quadtree, formulate our analysis of $E[S]$ in terms of an additional parameter $\beta$ (as well as $M$ and $N$), which is defined in (7). Here we shall not try to find a lower bound on the sum, as the sum itself is an upper bound.

The result of inserting (20) into (3) (i.e., a bound on $E[S]$) is decomposed into two sums $\sum_1$ and $\sum_2$ corresponding to the number of nodes at depth less than or equal to $d_0$, and all the nodes at a depth greater than $d_0$, respectively. In essence, we assume that the part of the tree at depth less than or equal to $d_0$ is complete. The value of $d_0$ is adjusted later so that the bounds on $E[S]$ are as tight as possible under the constraints that $d_0$ is an integer bounded by $N$ and that $\beta < 1$.

Thus, we have

$$\text{(21)} \qquad E[S] \leq 1 + \sum_1^N 4^d P'_{d-1} = 1 + \sum_{d=1}^{d_0} 4^d P'_{d-1} + \sum_{d=d_0+1}^{N} 4^d P'_{d-1} = \sum_1 + \sum_2.$$

A bound on $\sum_1$ is obtained as in the case of the MX quadtree:

$$\sum_1 = 1 + \sum_{d=1}^{d_0} P'_{d-1} 4^d \leq \sum_{d=0}^{d_0} 4^d \approx \frac{4}{3} 4^{d_0} = \frac{4}{3}\frac{M^2}{\beta^2}. \tag{22}$$

$\sum_2$ is evaluated by taking its binomial expansion to get a sum of the powers of $\left(\frac{1}{2}\right)^d$ (i.e., $\left(\frac{1}{2}\right)^0$, $\left(\frac{1}{2}\right)^d$, $\left(\frac{1}{2}\right)^{2d}$, ...). After some algebraic manipulation, the $\left(\frac{1}{2}\right)^0$ and $\left(\frac{1}{2}\right)^d$ terms cancel out, and we get

$$\sum_2 = \sum_{d=d_0+1}^{N} \sum_{k=2}^{M} C_k \left(\frac{1}{2}\right)^{kd-k} \cdot 4^d, \tag{23}$$

where

$$C_k = (-1)^{k-1}\left[\binom{M}{k} - \binom{M}{1}\cdot\binom{M-1}{k-1} + \binom{M}{2}\cdot\binom{M-2}{k-2}\frac{\pi}{8}\right].$$

Changing the order of summation and separating the $k = 2$ ($\sum_3$) and $k > 2$ ($\sum_4$) cases yield

$$\sum_2 = \sum_3 + \sum_4, \tag{24}$$

$$\begin{aligned} \sum_3 &= \sum_{d=d_0+1}^{N} C_2 \left(\frac{1}{2}\right)^{2d-2} 4^d \\ &= 2\left(1 - \frac{\pi}{8}\right) M(M-1)(N-d_0) \approx 1.215 M(M-1)(N-d_0), \end{aligned} \tag{25}$$

$$\begin{aligned} \sum_4 &= \sum_{k=3}^{M} C_k 2^k \sum_{d=d_0+1}^{N} \left(\frac{1}{2}\right)^{d(k-2)} \approx \sum_{k=3}^{M} C_k 2^k \frac{(\frac{1}{2})^{(k-2)(d_0+1)}}{1-(\frac{1}{2})^{k-2}} \\ &= 4\sum_{k=3}^{M} \frac{C_k}{1-(\frac{1}{2})^{k-2}}\left[\frac{\beta}{M}\right]^{k-2}. \end{aligned} \tag{26}$$

Now, let us examine the coefficients $C_k$:

$$\begin{aligned} C_k &= (-1)^{k-1}\left[\binom{M}{k} - \binom{M}{1}\cdot\binom{M-1}{k-1} + \binom{M}{2}\cdot\binom{M-2}{k-2}\frac{\pi}{8}\right] \\ &= (-1)^{k-1}\left[\binom{M}{k} - M\cdot\frac{k}{M}\cdot\binom{M}{k} + \binom{M}{2}\frac{k(k-1)}{M(M-1)}\cdot\binom{M}{k}\frac{\pi}{8}\right] \\ &= (-1)^{k-1}\binom{M}{k}\left[1 - k + \frac{k(k-1)}{2}\frac{\pi}{8}\right]. \end{aligned} \tag{27}$$

By checking a few values of $k$, it can be shown that, for $k \geq 3$,

$$-0.137 M^k \leq C_k \leq 0.027 M^k. \tag{28}$$

Inserting (28) into (27) and accounting for the worst cases lead to

$$-1.1 M^2 \frac{\beta}{1-\beta} \leq \sum_4 \leq 0.22 M^2 \frac{\beta}{1-\beta}. \tag{29}$$

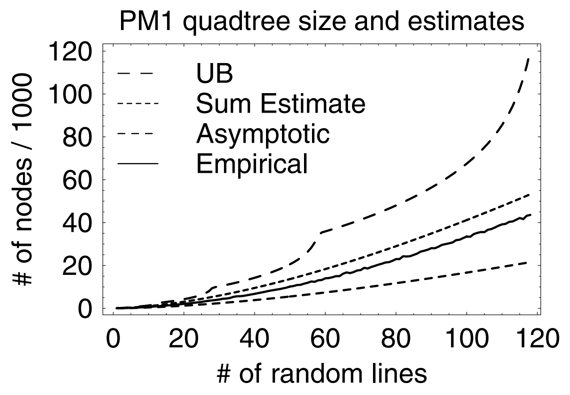

FIG. 6. *The upper bound (UB), the sum estimate (Sum Estimate), which is also an upper bound, the empirical estimate (Empirical), and the asymptotic approximation (Asymptotic) on the number of nodes in a PM quadtree. The x and y axes correspond to the number of lines and nodes, respectively, for a tree of depth $N = 10$. Note the "rounded staircase-like" behavior of the upper bound resulting from our method of analysis, which calculates several bounds (each for different integer values of the $d_0$ parameter) retaining the tightest one.*

Therefore, the contribution of $\sum_4$ to $E[S]$ is $O(M^2)$. Collecting the contributions of $\sum_1$, $\sum_3$, and $\sum_4$, we have

$$\begin{aligned} E[S] &= \textstyle\sum_1 + \sum_3 + \sum_4 \\ &\leq 1.215M(M-1)N + M^2 \left[\frac{4}{3}\frac{1}{\beta^2} + 0.22\frac{\beta}{1-\beta} - 1.215\log_2\frac{M}{\beta}\right]. \end{aligned} \tag{30}$$

Once again, $d_0$ is chosen to minimize (30), subject to the conditions that $d_0$ is an integer less than $N$ and that $\beta$ (as defined in (7)) is less than 1. Figure 6 shows the value of the upper bounds given by (30) (the uppermost curve) as a function of $M$ at a maximal depth of $N = 10$, as well as the value of the sum (21) from which it is derived (second curve from the top). Recall that, here, the sum is also an upper bound itself. In addition, we also recall that the upper bounds in the figure are minimal in the sense that, for each value of $M$, upper bounds were calculated for every possible value of $d_0$ (subject to the constraint $\beta < 1$) and the minimal (tightest) upper bound was taken.

**4.2.2. An asymptotic approximation.** The bound developed above is valid but may not be tight. We now propose an asymptotic approximation which is expected to be more accurate when $N$ is large. Denoting the parent node of $v$ (as well as the corresponding region) by $fv$, we have

(31)
$$\begin{aligned} \text{Prob}(v_e|v_s) = \text{Prob}((fv_e \textit{ and } fv_s)|v_s) &= \text{Prob}(fv_s|v_s)\text{Prob}(fv_e|(v_s \textit{ and } fv_s)) \\ &= \gamma_d \text{Prob}(fv_e|(v_s \textit{ and } fv_s)) \leq \gamma_d, \end{aligned}$$

where $\gamma_d$ is the conditional probability $\text{Prob}(fv_s|v_s)$. As discussed above, the only possibility of this conditional event not happening is when exactly two lines cross both the region $v$ and its parent region $fv$, and when these lines are arranged so that they do not intersect within $v$ but do intersect within $fv$.

This probability does not have a simple expression, though, as it depends on the number of lines intersecting the region, which is distributed differently for different depths. Note that, as the depth is increased, the probability that more than two lines intersect the region (i.e., $\sum_{k=3}^{M} p_{d,k}$ (see (4)) gets smaller and eventually becomes negligible. Quantitatively, the ratio of the probability that three or more lines intersect the region and the probability that exactly two lines intersect the region (i.e., $(\sum_{k=3}^{M} p_{d,k})/p_{d,2}$) tends to zero as $d$ increases. As a result, the probability $\gamma_d$ correspondingly decreases and reaches a constant asymptotic value $\gamma_\infty \approx 0.405$, which corresponds to the assumption that exactly two lines intersect the parent region $fv$. (The value of $\gamma_\infty$ was estimated by probabilistic simulation.)

Thus, with the asymptotic approximation, and relying on (31), we get a tighter bound on $P_d$:

$$P_d = \text{Prob}(v_s)\text{Prob}(v_e|v_s) \leq \text{Prob}(v_s)\gamma_\infty = P_d^\infty. \tag{32}$$

Using the bound $P_d^\infty$ as $P'_d$ in the sum (21) we get an asymptotic approximation, which is expected to better model the splitting process when the depth is high. Note that this approximation is not an upper bound, as the asymptotic approximation of the splitting probability holds only for large depths. Naturally, the asymptotic approximation is good when the majority of the nodes satisfies the above assumption but is expected to fail otherwise (i.e., when the maximal depth of the tree is small and the number of lines is large). In our experiments, we show that this is indeed the case (see Table 1). Note that technically the appoximation is equal to the upper bound we obtained above (when we used $P'_d = Prob(v_s)$ in the sum) multiplied by a factor of $\gamma_\infty$. Figure 6 shows the value of the resulting asymptotic approximation as a function of $M$ at a maximal depth of $N = 10$ (lowest curve).

**4.2.3. Interpretation.** The number of possible line pairs in the image is $\binom{M}{2}$. Multiplying this number by $\alpha$ yields the expected number of intersections. Approximating $\binom{M}{2}$ by $M^2/2$, we have that the expected number of vertices (line intersections) in the whole image is approximately $\frac{\pi}{16}M^2$. Considering only the dominant first term in (30), which is roughly proportional to $N \cdot M^2$, we see that the results of the analysis may be interpreted as confirming that both the number of vertices and the maximum depth of the tree impact the number of nodes necessary. The dependence of $E[S]$ on the number of vertices (i.e., the factor $M^2$) is intuitively clear, as each vertex will be stored in a separate node in the tree.

The dependence of $E[S]$ on the maximum depth of the tree (i.e., $N$) is less obvious. On the one hand, it could be said that, since our result is only an upper bound, it may be that the actual PM quadtree node count does not increase with the depth. However, it actually confirms a known result that, when the vertices of the line segements are constrained to lie on the grid points of a $2^n \times 2^n$ grid, the PM quadtree can be as deep as $4n$ [47]. In fact, for an image generated by the random line model, there is no constraint on the positions of the vertices (i.e., the intersection points), and thus the maximum decomposition depth can be even higher than $4 \cdot n$, as this depth depends on the locations of the vertices. Thus we see that this dependence is in agreement with the fact that, in the worst case, some of the vertices created by a random configuration of lines could appear in nodes at the maximum level. The linear dependence predicts that the probability of the occurrence of such "bad" line sets is not zero. This behavior was confirmed by our experiments, which found that the expected number of nodes in the PM quadtree increases linearly, but very slowly, with depth (see subsection 4.6). A related result is that, in most cases, the randomly generated PM quadtree is small, but in some rare cases it can be very large.

**4.3. Bucket PMR$_q$ quadtree.** A bucket PMR$_q$ quadtree represents a collection of line segments in the plane. It partitions the plane into square blocks using the splitting rule that stipulates that a block is split if both the depth of its corresponding node is less than $N$ and more than $q$ line segments pass through the block. Thus if the block contains $q$ or less line segments, then it is not subdivided further and its corresponding node is a leaf. Otherwise, it is subdivided and its corresponding node has four children (see Figure 2(d)).

Note that if the expected degree of a vertex $v$ is $k$, then choosing $q < k$ results in splitting the node containing $v$ to the maximal depth. Observe that for the random image model, the degree of every vertex is 4. Thus, using a Bucket PMR$_q$ quadtree, with $q < 4$ is not recommended. Nevertheless, for the sake of completeness, we briefly comment on these special cases below.

The Bucket PMR$_0$ quadtree is an MX quadtree. For the Bucket PMR$_1$ quadtree,

$$P_d < \mathrm{Prob}(v_s) = 1 - p_{d,0} - p_{d,1}. \tag{33}$$

Observe that $\mathrm{Prob}(v_e|v_s) = 1$ for all Bucket PMR quadtrees. For the Bucket PMR$_2$ quadtree,

$$P_d < \mathrm{Prob}(v_s) = 1 - p_{d,0} - p_{d,1} - (1-\alpha) \cdot p_{d,2}. \tag{34}$$

For the Bucket PMR$_3$ quadtree, the splitting probability is the same as in (34) with the subtraction of a term corresponding to the small probability that three line segments intersect the region, but not each other. Thus, the splitting probabilities for $q = 3$ satisfy

$$P_d < \mathrm{Prob}(v_s) < 1 - p_{d,0} - p_{d,1} - (1-\alpha) \cdot p_{d,2} \tag{35}$$

and are assumed to be very close to its bound.

For a Bucket PMR$_2$ quadtree, it is clear that the splitting probability is identical to that of the PM quadtree, apart from changing a multiplicative constant from $\alpha$ to $1-\alpha$. Therefore, the expected number of nodes is given by an expression similar to (30). The same considerations hold for the Bucket PMR$_3$ quadtree.

While it is less obvious, the Bucket PMR$_1$ quadtree behaves very similarly to the PM quadtree as well. This is apparent by observing that the term $\sum 3$ (25), which dominates the number of nodes in the PM quadtree, grows by a factor of $1/(1-\pi/8)$ if the split probability (33) is used.

The situation, however, changes dramatically when considering bucket PMR$_q$ quadtrees with values of $q$ equal to 4 and higher. For $q = 4$, the splitting probability satisfies

$$P_d < \mathrm{Prob}(v_s) = 1 - p_{d,0} - p_{d,1} - p_{d,2}. \tag{36}$$

The probability $P_d$ is smaller than the right side of the above inequality because the probabilities of some additional events that imply the absence of splitting are not included. For example, if three or four lines intersect the region, but not each other, then the region is not split. The inclusion of these contributions is complicated and is not needed for computing an upper bound. Using the upper bound $P'_d = \mathrm{Prob}(v_s)$ in (36) as the probability $P_d$, and the same techniques as in subsection 4.2, we define $\beta$ and $d_0$ as in (7) and decompose the sum corresponding to $E[S]$ into two sums, $\sum_1$

from (22) and $\sum_2$ from (23). Here, however,

$$(37)\quad \begin{aligned} C_k &= (-1)^{k-1}\left[\binom{M}{k} - \binom{M}{1}\cdot\binom{M-1}{k-1} + \binom{M}{2}\cdot\binom{M-2}{k-2}\right] \\ &= (-1)^{k-1}\left[\binom{M}{k} - \binom{M}{1}\cdot\frac{k}{M}\cdot\binom{M}{k} + \binom{M}{2}\cdot\frac{k(k-1)}{M(M-1)}\cdot\binom{M}{k}\right] \\ &= (-1)^{k-1}\binom{M}{k}\left[\frac{(k-1)(k-2)}{2}\right], \end{aligned}$$

implying that $C_2 = 0$ and that

$$(38)\qquad -\frac{1}{8}M^k \le C_k \le \frac{1}{6}M^k.$$

The bounds (38) are derived by evaluating $C_k$ for several values of $k$ and by observing that $|C_k|$ decreases with $k$. The fact that $C_2 = 0$ is important, as it means that $\sum_3$ is 0, and thus once we bound the finite geometric series in $\frac{1}{2}$ by the infinite geometric series, the expected number of nodes will no longer depend on $N$. Now,

$$(39)\qquad \sum\nolimits_4 = \sum_{k=3}^{M} C_k 2^k \sum_{d=d_0+1}^{N}\left(\frac{1}{2}\right)^{d(k-2)} \approx \sum_{k=3}^{M} C_k 2^k \frac{(\frac{1}{2})^{(k-2)(d_0+1)}}{1-(\frac{1}{2})^{k-2}}$$

$$(40)\qquad = 4\sum_{k=3}^{M}\frac{C_k}{1-(\frac{1}{2})^{k-2}}\left[\frac{\beta}{M}\right]^{k-2} \le \frac{4}{3}M^2\frac{\beta}{1-\beta}.$$

This derivation is based on bounding the finite geometric progression with the corresponding infinite progression, expressing $d_0$ in terms of $\beta$ and $M$, and performing some additional arithmetic manipulations. Therefore,

$$(41)\qquad E[S] \le \frac{4}{3}M^2\left[\frac{1}{\beta^2} + \frac{\beta}{1-\beta}\right].$$

Figure 7 shows the value of the upper bound given by (41) (uppermost curve) as a function of $M$ at a maximal depth of $N = 10$ as well as the value given by the corresponding sum (middle curve). Again, we recall that the upper bounds in the figure are minimal in the sense that, for each value of $M$, upper bounds were calculated for every possible value of $d_0$ (subject to the constraint $\beta < 1$) and the minimal (tightest) upper bound was chosen. To get a clearer interpretation of the bound, select a specific value for $\beta$, say, 0.75 (which corresponds to sets of $6, 12, 24, \ldots$ lines and to $d_0$ values of $3, 4, 5, \ldots$, respectively). Then, we get

$$(42)\qquad E[S] \le 6.37M^2 \qquad (q = 4).$$

The bound (42) means that for a bucket PMR$_q$ quadtree (with $q = 4$) the expected number of nodes is proportional to the expected number of intersection points (approximately $\frac{\pi M^2}{16}$ as derived in subsection 4.2.2), henceforth referred to as *vertices*, and does not depend on the maximal depth $N$. Therefore, if the maximal depth is large enough, the subdivision stops before reaching the maximal depth almost everywhere. Alternatively, the $O(M^2)$ intersection points result in $O(M^2)$ line segments. Thus, an equally powerful characterization of this result is that the number of nodes is proportional to the number of line segments and does not depend on the maximal

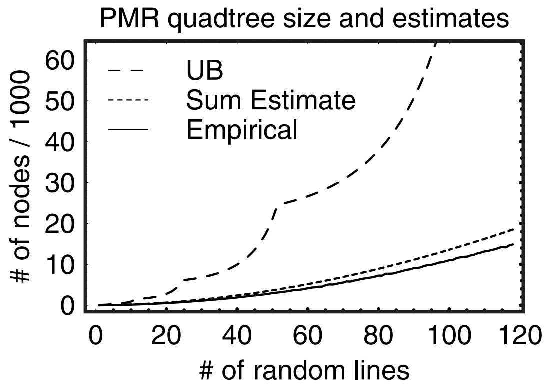

FIG. 7. *The upper bound (UB), the sum estimate (Sum Estimate), and the empirical estimate (Empirical) on the number of nodes necessary to store a PMR$_4$ quadtree. The x and y axes correspond to the number of lines and nodes, respectively, for a tree of depth $N = 10$. Note the "rounded staircase-like" behavior of the upper bound resulting from our method of analysis, which calculates several bounds (each for different integer values of the $d_0$ parameter) retaining the tightest one.*

depth of the tree. Higher values of $q$ require a more complicated analysis, as the number of line segments created by the intersection of more than two lines is a random variable of more complicated statistics. In particular, we have more possibilities to consider than just the two events corresponding to the intersection or nonintersection of two lines. It is clear, however, that the splitting probability decreases as $q$ increases, and therefore, for $q \geq 4$ the bound (41) still holds. Note that it is not possible to reduce this bound by much (even for higher values of $q$) since the bound on the first term, $\sum_1$, remains $\frac{4}{3}\frac{M^2}{\beta^2} < \frac{4}{3}M^2$.

Regions that contain a vertex (resulting from the intersection of lines) are split if the number of line segments that are incident at this vertex is higher than $q$. This explains the significant change in the quadtree size when $q \geq 4$. For our random image model, all vertices have four line segments incident at them, and therefore the regions in a Bucket PMR$_2$ quadtree must split until the maximal depth is achieved. On the other hand, in the Bucket PMR$_4$ quadtree (and for values of $q \geq 4$), regions that contain a single vertex are not split, which leads to a tree whose size is independent of the maximal depth $N$.

**4.4. PMR$_q$ quadtree.** A PMR$_q$ quadtree represents a collection of line segments in the plane. It depends on a parameter $q$ and is created as a dynamic result of a sequence of insertion of line segments using the splitting rule that says a block $b$ is split once, and only once, if $b$ is intersected by the new line segment and if $b$ already contains $q$ or more line segments. Thus the block is subdivided at most once when a new line segment, which intersects it, enters the structure. Clearly, this may not be enough to ensure that the number of line segments stored in each leaf node is $q$ or less.

Note that since the PMR$_q$ quadtree depends on the order in which the line segments are inserted into it, we must specify some order. We assume that the $M$ infinite lines are generated in one step, followed by the insertion of the $O(M^2)$ line segments in an arbitrary order. For the case that $q \geq 4$, we may use the upper bound (41) that we obtained on the number of nodes in a bucket PMR$_q$ quadtree to also bound the number of nodes in a PMR$_q$ quadtree. First, note that the node set associated with the Bucket $\mathrm{PMR}_q$ quadtree is a superset of the node set associated with the $\mathrm{PMR}_q$ quadtree, provided that the maximal depth of the tree is high enough, since the number of times a node in the Bucket $\mathrm{PMR}_q$ quadtree is split upon insertion is at least as large as the number of times it is split in the $\mathrm{PMR}_q$ quadtree. Second, observe that the maximal depth of the $\mathrm{PMR}_q$ quadtree for $i$ line segments is $i - q$, as the first split happens when the $q+1$st line segment is inserted. Therefore, the $\mathrm{PMR}_q$ quadtree representation of the $O(M^2)$ line segments created by the $M$ infinite random lines is of a maximal finite depth $O(M^2) - q$. Thus, we can construct a bucket $\mathrm{PMR}_q$ quadtree with a maximal depth $N = M^2$ so that, regardless of the order in which the $O(M^2)$ line segments are inserted into the $\mathrm{PMR}_q$ quadtree $t$, the nodes in $t$ will always be a subset of the nodes in the bucket $\mathrm{PMR}_q$ quadtree $v$ for the $O(M^2)$ line segments. Recall that the bound on the number of nodes in a bucket $\mathrm{PMR}_q$ quadtree that we obtained was independent of the depth $N$ of the bucket $\mathrm{PMR}_q$ quadtree as we let $N$ go to infinity when we computed $\sum_4$ in (40). Thus the bound that we obtained in (41) is also good for bucket $\mathrm{PMR}_q$ quadtrees of any depth. Therefore, it is also applicable to $\mathrm{PMR}_q$ quadtrees subject to $q \geq 4$.

Unfortunately, we cannot get a reasonable bound using this method for $q < 4$. For $q = 2$, for example, recall that the number of nodes in the Bucket $\mathrm{PMR}_2$ quadtree is similar to this number in the PM quadtree, which is given by (30). Therefore, the upper bound on the number of nodes in the Bucket $\mathrm{PMR}_2$ quadtree is linear in the maximum depth. The guarantee that the Bucket $\mathrm{PMR}_2$ quadtree is a superset of the $\mathrm{PMR}_2$ quadtree requires setting the maximal depth of the Bucket $\mathrm{PMR}_2$ quadtree higher than the maximal of the $\mathrm{PMR}_2$ quadtree. The latter, however, may be as high as $O(M^2)$, where $M$ is the number of infinite random lines, thus giving an unrealistically high upper bound. Therefore, obtaining an upper bound on the number of nodes in a $\mathrm{PMR}_2$ quadtree is an issue left for future research, although, as mentioned before, the case of $q \geq 4$ is more relevant from a practical standpoint, as we do not want the common situation of a road junction (i.e., when four line segments meet at a point) to cause an arbitrarily large amount of splitting. The same considerations apply for $q = 1$ and $q = 3$.

**4.5. A general discussion of the bounds.** The bounds developed here lead to the following asymptotic results on the expected number of nodes as a function of the number of random lines $M$ and the level of permitted subdivision $N$:

$$\begin{aligned} &\textit{MX quadtree } E[S] = O(M \cdot 2^N),\\ &\textit{PM quadtree } E[S] = O(M^2 \cdot N),\\ &\textit{PMR}_q \textit{ quadtree } E[S] = O(M^2) \quad (q \geq 4),\\ &\textit{bucket PMR}_q \textit{ quadtree } E[S] = O(M^2 \cdot N) \quad (q = 2),\\ &\textit{bucket PMR}_q \textit{ quadtree } E[S] = O(M^2) \quad (q \geq 4). \end{aligned} \tag{43}$$

These results were obtained by summing the expected number of nodes at each level of the hierarchical structures. The differences are due to the different rates at which the splitting probabilities decrease as the depth increases. For example, for the MX quadtree, the probability that a node splits decreases with the depth, but does not decrease fast enough, resulting in a tree of exponential size. On the other hand, for the PM quadtree, the splitting probability decreases at a fast enough rate to offset the exponential growth of the tree, thereby resulting in a tree whose size is proportional to its depth. The same holds for the Bucket $\mathrm{PMR}_2$ quadtree. For $q \geq 4$, for both the $\mathrm{PMR}_q$ quadtree and the bucket $\mathrm{PMR}_q$ quadtree, the splitting probability decreases at an even faster rate, thereby implying that the expected number of nodes at each level decreases exponentially with the depth and that the sum converges (i.e., is independent of the depth of the tree).

The main conclusions that can be drawn from these results are as follows. For the MX quadtree, the number of nodes is proportional to the total length of the line segments. This conclusion confirms a similar result obtained by Hunter [22] and Hunter and Steiglitz [23]. For both the PM and Bucket $\text{PMR}_2$ quadtrees, the number of nodes can be interpreted as being proportional to the product of the number of intersections among the lines (i.e., the original lines in the random image model or, alternatively, the vertices of the resulting line segments) and the maximal depth of the tree. For both the $\text{PMR}_q$ quadtree and the Bucket $\text{PMR}_q$ quadtree with node capacities $q \geq 4$, the number of nodes is proportional to the number of line segments (recall that there are $O(M^2)$ intersection points for the $M$ lines, resulting in $O(M^2)$ line segments). It also appears that, for the $\text{PMR}_q$ $(q \geq 4)$ quadtree, almost everywhere, the subdivision stops before the maximal depth, provided, of course, that the density of the lines (i.e., the $M$ random lines) does not make the tree almost full.

To get the actual values of the bounds (in contrast to the orders of magnitude summarized above) we use the exact upper bounds as given in (15), (30), and (41), which depend on a parameter $\beta$. It is worth re-emphasizing that the values $d_0$ and $\beta$ are not a part of the random image model. They are just parameters used to simplify the expression of the bounds. In order to apply these bounds, it is required to choose a value of $\beta$ which minimizes them while satisfying relation (7) (with $d_0$ being an integer bounded by $N$). Fixing the value of $\beta$ at some constant (e.g., 0.75) gives a bound, which may not be the tightest but is still useful for understanding the behavior of the size of the data structure. From a strict theoretical standpoint, such an arbitrary choice of a value for $\beta$ is not justified because it usually implies a noninteger value for $d_0$. Note also that the upper bounds contain negative terms which reduce the bounds and make them tighter. These negative terms compensate for nodes which are counted twice in other (positive) terms. For example, nodes in the PM quadtree at a depth less than $d_0$ are accounted for both by the $\sum_1 = \frac{4}{3}\frac{M^2}{\beta^2}$ term and also by the $M^2 \cdot N$ term. The negative term $-M^2 \log_2 \frac{M}{\beta} \approx -M^2 d_0$ compensates for this situation. Note also that these bounds hold for all values of $M$ and $N$. However, they become trivial when $M \geq 2^N$.

The bounds in this paper were computed under the assumption of a particular image model. We conjecture that the results apply also to more general images. In section 5 we examine several methods for inferring the size of quadtrees that represent real maps and test them experimentally. In essence, we characterize the map by some property, which may be the total length of its constituent line segments, the number of vertices, etc., and use this property to specify a class of random images which share the same property (in an expected value sense). Next, we conjecture that the number of quadtree nodes required to represent the real map is equal to the expected number of nodes required to represent a random map from that class.

**4.6. Some experimental results for instances of the random model.** We conducted several experiments with synthetic and real data. In this section we describe the tests that were made with synthetic data. They were aimed at determining how close the upper and lower bounds on the expected storage costs come to the actual storage costs when using random data. Section 5 describes the results of tests with real data.

In these experiments, we built the MX, PM, and Bucket $\text{PMR}_4$ quadtrees of

Table 1

*Comparison of the result of expression* (3) *in plain form (MODEL) and asymptotic form (MODEL$_\infty$), upper bounds (UB), lower bounds (LB), and the actual (experimental) number of nodes (ER) for M random lines and a maximal depth N in an MX, PM, and Bucket PMR$_4$ quadtrees.*

| M | N | MX | | | | PM | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | MODEL | UB | LB | ER | MODEL | UB | MODEL$_\infty$ | ER |
| 25 | 10 | 94.8K | 97.5K | 90.7K | 94.9K | 3.88K | 5.35K | 1.57K | 2.58K |
| 50 | 10 | 179K | 189K | 162K | 181K | 12.8K | 19K | 5.18K | 9.43K |
| 75 | 10 | 255K | 269K | 225K | 256K | 25.1K | 43.2K | 10.2K | 19.3K |
| 100 | 10 | 325K | 365K | 259K | 325K | 39.9K | 64.9K | 16.2K | 32.2K |
| 25 | 14 | 1.63M | 1.63M | 1.62M | 1.62M | 6.78K | 8.27K | 2.74K | 2.94K |
| 50 | 14 | 3.23M | 3.24M | 3.21M | 3.24M | 24.6K | 30.9K | 9.95K | 11.7K |
| 75 | 14 | 4.82M | 4.83M | 4.79M | 4.80M | 51.6K | 70.2K | 20.9K | 25.5K |
| 100 | 14 | 6.39M | 6.43M | 6.32M | 6.37M | 87K | 113K | 35.2K | 45.2K |

| M | N | Bucket PMR$_4$ | | |
|---|---|---|---|---|
| | | MODEL | UB | ER |
| 25 | 10 | 0.846K | 4.34K | 0.669K |
| 50 | 10 | 7.5K | 32.5K | 6.01K |
| 75 | 10 | 3.38K | 17.4K | 2.73K |
| 100 | 10 | 13.1K | 69.5K | 10.6K |
| 25 | 14 | 0.863K | 4.34K | 0.681K |
| 50 | 14 | 3.52K | 17.4K | 2.83K |
| 75 | 14 | 7.97K | 32.5K | 6.35K |
| 100 | 14 | 14.2K | 69.5K | 11.4K |

several depths $N$, using random synthetic data created by the random image model described in section 3. For each case, several instances of each random image were created and the average quadtree size was calculated. The results are summarized in Table 1 (see also Figure 5). They usually agree with the analytical predictions, and, in particular, the nonasymptotic bounds for all of the quadtree variants always hold.

We also evaluated the expression (3) for the values of the number of lines $M$ and the maximum depth $N$ of the quadtrees. We found that while this sum closely approximates the actual expected number of nodes for the MX quadtree, it only bounds the number of nodes for the PM and the Bucket PMR$_4$ quadtrees. This is expected because, for these trees, the $P'_d$ expressions, which we used, are bounds of the probabilities and not the probabilities themselves. The PM quadtree asymptotic approximation, which corrects every bound by the $\gamma_\infty$ factor (see subsection 4.2.2), yields a more accurate estimate for trees with a high depth and a low number of lines (as expected). For the Bucket PMR$_4$ quadtree, the sum is a more accurate estimate (as it errs only by about 25%).

The upper bounds obtained for the MX quadtree were consistently very close to the observed node counts. In contrast, the upper bounds for the PM and Bucket PMR$_4$ quadtrees consistently exceed the observed node counts by factors as high as 3 and 7, respectively. This difference can be attributed to the simplifications made in the bound derivation process. As mentioned above, we did not attach much significance to obtaining tighter bounds, as they are not used for predictions but, instead, only for performing a qualitative comparison between the different quadtree representations.

We also conducted some experiments to test the performance of the PM quadtree under "asymptotic" conditions—that is, when $N$ is relatively large and $M$ is not too high. These experiments were undertaken to determine whether or not the PM quadtree grows linearly with the maximal depth $N$ (like the upper bound that we found). The results were still inconclusive; when examining the number of nodes, we found that the average number (over 1000 random trials) increases slowly but steadily with maximal depth, but the high variance still does not allow us to conduct a decisive statistical test. We also examined histograms of the actual maximal depth

Table 2

*Description of the TIGER files maps (see Figure 8) used in the experiments as well as the corresponding actual storage requirements for the MX, PM, and Bucket PMR$_4$ quadtrees.*

| Map Name | Depth | Vertices | NSV | NormL | Segments | Number of nodes | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | MX | PM | PMR$_4$ |
| Falls Church | 10 | 448 | 317 | 16.77 | 638 | 76869 | 4105 | 1317 |
| Falls Church | 12 | 448 | 317 | 16.77 | 638 | 336873 | 4477 | 1349 |
| Falls Church | 14 | 448 | 317 | 16.77 | 638 | 1387557 | 4633 | 1381 |
| Falls Church | 16 | 448 | 317 | 16.77 | 638 | 5600821 | 4681 | 1413 |
| Alexandria | 10 | 4074 | 2123 | 49.89 | 5380 | 191469 | 25249 | 9709 |
| Alexandria | 12 | 4074 | 2123 | 49.89 | 5380 | 915025 | 28929 | 10105 |
| Alexandria | 14 | 4074 | 2123 | 49.89 | 5380 | 3873949 | 30413 | 10429 |
| Alexandria | 16 | 4074 | 2123 | 49.89 | 5380 | 15774517 | 31061 | 10753 |
| Arlington | 10 | 6657 | 3978 | 67.91 | 9205 | 242373 | 43913 | 17637 |
| Arlington | 12 | 6657 | 3978 | 67.91 | 9205 | 1234449 | 54753 | 18677 |
| Arlington | 14 | 6657 | 3978 | 67.91 | 9205 | 5339437 | 58949 | 19481 |
| Arlington | 16 | 6657 | 3978 | 67.91 | 9205 | 21891973 | 61277 | 20281 |
| Howard | 10 | 15009 | 5283 | 57.58 | 17419 | 191861 | 63361 | 29321 |
| Howard | 12 | 15009 | 5283 | 57.58 | 17419 | 1031317 | 95945 | 32921 |
| Howard | 14 | 15009 | 5283 | 57.58 | 17419 | 4563069 | 111169 | 33937 |
| Howard | 16 | 15009 | 5283 | 57.58 | 17419 | 18866341 | 118305 | 34713 |
| DC | 10 | 12818 | 8805 | 107.99 | 19183 | 332589 | 73477 | 35377 |
| DC | 12 | 12818 | 8805 | 107.99 | 19183 | 1805965 | 92153 | 37813 |
| DC | 14 | 12818 | 8805 | 107.99 | 19183 | 7971497 | 99093 | 39685 |
| DC | 16 | 12818 | 8805 | 107.99 | 19183 | 32905753 | 103297 | 41533 |
| Calvert | 10 | 29174 | 4690 | 60.39 | 31143 | 213965 | 88769 | 44057 |
| Calvert | 12 | 29174 | 4690 | 60.39 | 31143 | 1094453 | 125053 | 48129 |
| Calvert | 14 | 29174 | 4690 | 60.39 | 31143 | 4734589 | 133141 | 48493 |
| Calvert | 16 | 29174 | 4690 | 60.39 | 31143 | 19402105 | 135241 | 48665 |
| Prince George's | 10 | 50161 | 18055 | 117.55 | 59551 | 315289 | 157977 | 88485 |
| Prince George's | 12 | 50161 | 18055 | 117.55 | 59551 | 1937553 | 277209 | 104993 |
| Prince George's | 14 | 50161 | 18055 | 117.55 | 59551 | 8993017 | 318889 | 107889 |
| Prince George's | 16 | 50161 | 18055 | 117.55 | 59551 | 37751493 | 334265 | 109825 |
| Montgomery | 10 | 79822 | 19793 | 118.74 | 90022 | 299601 | 186265 | 115693 |
| Montgomery | 12 | 79822 | 19793 | 118.74 | 90022 | 1927197 | 365701 | 145389 |
| Montgomery | 14 | 79822 | 19793 | 118.74 | 90022 | 9068057 | 424869 | 149613 |
| Montgomery | 16 | 79822 | 19793 | 118.74 | 90022 | 38212101 | 449905 | 151913 |

as a function of the allowed maximal depth. Here we found that even for depths as large as 30 we still had trials (and corresponding random collections of 25 lines), where the PM quadtree had nodes at this maximal depth. In contrast, this never happened for the PMR quadtree.

**5. Predicting storage requirements for real data.** In this section we show that the expected storage predictions, derived for the random image model, are also useful when real data is considered.

We conducted our tests using real data corresponding to U.S. city and county road maps that are part of the TIGER files used by the U.S. Census Bureau (see Figure 8). We used maps ranging from a small map having only 585 road segments (Falls Church, Virginia) to the largest map having 39,719 segments (Montgomery County, Maryland). The actual data for the maps, such as the depth ($N$), number of vertices, segments, nonshape vertices (abbreviated $NSV$ and described below), the normalized length (abbreviated $NormL$ and equal to the total length of the line segments divided by $2^N$), and the number of nodes in the MX, PM, and Bucket PMR$_4$ quadtrees, are given in Table 2.

Our approach to applying the expected node count predictions to a real map $r$ depends on finding, for each map $r$, a class $c$ of a random line image which shares some property with $r$. The expected number of nodes required to represent a random image from class $c$ is taken to be the estimate for the number of nodes required to represent the map $r$.

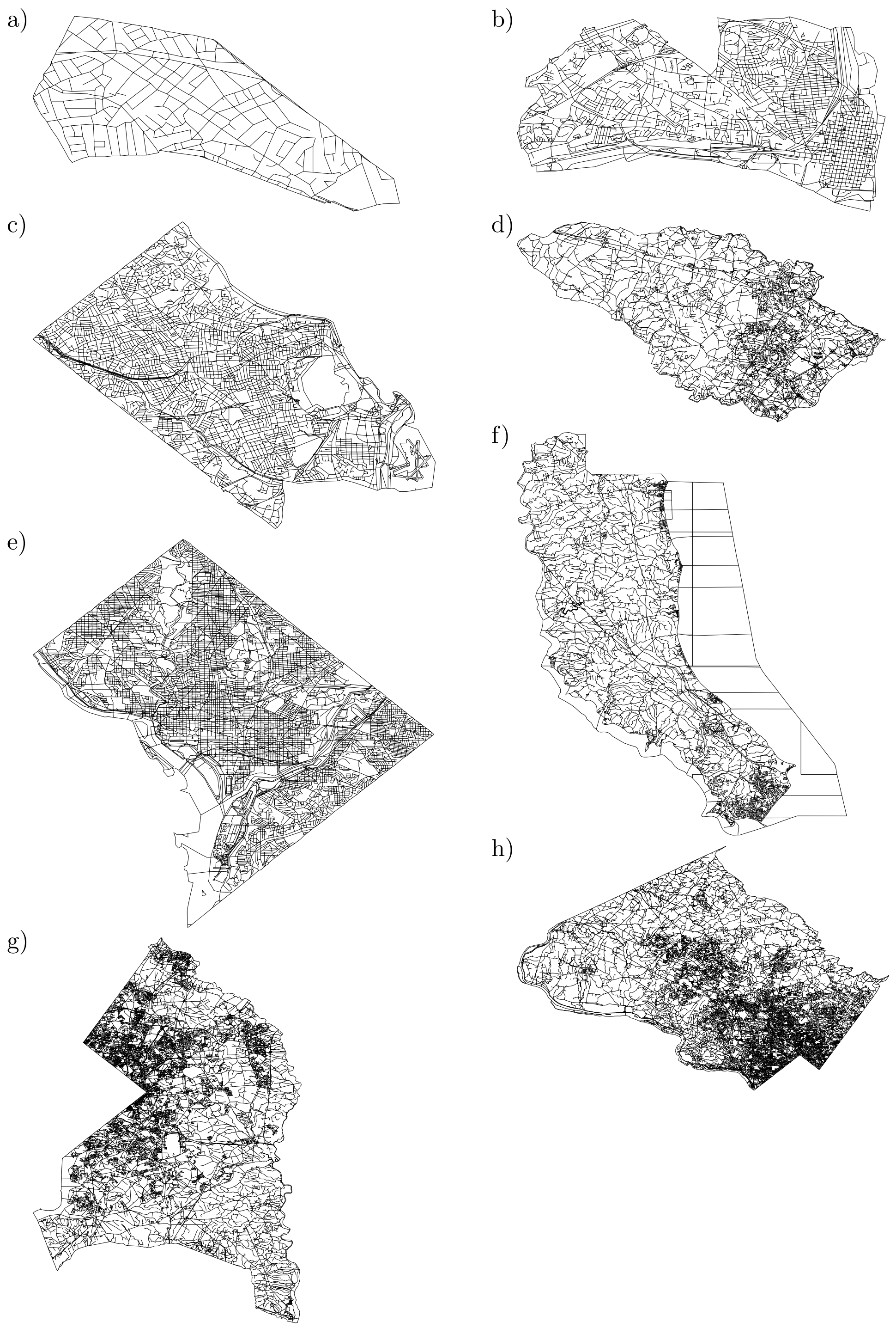

FIG. 8. *Eight maps used to test the estimates on the number of nodes in the representing quadtrees.*

In most cases, the equivalent random image to a given map $r$ is specified, as in section 3, by the effective number of lines $\hat{M}$, which is inferred, in a number of alternative ways (termed *estimators* and described below), from $r$. Another degree of freedom in the specification of the equivalent random image is gained if we account for sparse or empty image parts. Therefore, the equivalent random image is specified in two stages. First, a random image is specified by $\hat{M}$. Second, only a fraction of this image is retained, while the rest is considered to be empty. A normalization factor is inferred from the image $r$ and specifies the fraction that is retained. Thus, we assume that the number of nodes is directly proportional to the portion of the area that is nonempty, and we use this normalization factor to obtain the number of nodes in the equivalent class. For most of the estimators, no normalization is done and the images of the equivalent class are just random images resulting from the random image model defined in section 3. The parameters specifying the class of random images are estimated from other parameter values of the real map $r$ such as the total length $L_{map}$, the total number of vertices $V_{map}$, and the total number of segments $S_{map}$.

The first estimator that we consider is termed an *L-based estimator*. This estimator is based on measuring the total length $L_{map}$ of the line segments. Recall that, for the random images created by the random image model, the expected total line length $E[L]$ is $\pi/4 \cdot M \cdot 2^N$. This is the result of (16), which follows directly from Theorem GP2. Therefore, the number of lines in every one of the random images which share the "expected line length" property with the given map is estimated by $\hat{M} = (L_{map}/2^N) \cdot (4/\pi)$. For this estimator, there is no area normalization (that is, the equivalent random image is assumed not to contain empty regions). For example, the total length of the line segments in the smallest map (i.e., Falls Church, Virginia shown in Figure 8(a)) that we used is $16.77 \cdot 2^N$, which yields $\hat{M} \approx 21$ (the length is normalized relative to the side of the map). Note that this has the effect of converting the line segments of the test image to a different number of infinite lines, the intersection of which, with the space in which they are embedded, has the same total length (in an expected value sense).

The second estimator that we consider is termed a *V-based estimator*. It uses the number of vertices $V_{map}$ in the map to estimate the effective number of lines $\hat{M}$ by treating all vertices as intersection points between random lines. Recalling that the expected number of intersection points between the lines of the random image model is $E[V] = \pi M^2/16$, we have $\hat{M} = \sqrt{16V_{map}/\pi}$. Here, again as in the case of the L-based estimator, no area normalization is performed. Note, however, that most vertices (termed *shape points* [7]) in a typical map image are the results of a piecewise polygonal approximation of a curve, thereby implying that only two line segments meet at them. This is in contrast to the intersection points in a random image, which are true intersections (i.e., they correspond to the intersections of pairs of random lines). A simple heuristic, which we use here, deletes these degree-2 vertices from the total vertex count and yields the *NSV estimator*. Clearly, there are many cases for which this heuristic is nonapplicable. For example, we could not apply it to a non–self-intersecting curve (e.g., a spiral), as vertex deletion would predict that the representation requires just a single node (which is clearly erroneous).

The third estimator that we consider is termed an *S-based estimator*. Like the L-based and V-based estimators, it is based on replacing the given map with a random line image of the same size, and no area normalization is performed. Again, we calculate an effective number of random lines $\hat{M}$, but this time it is based on the number of line segments. We assume that each vertex corresponds to the intersection of two random lines, and hence results in four line segments (also termed *edges* or *segments*). Therefore, each vertex has degree 4. Each line segment is incident at two vertices. This is approximately true, as only a small fraction of the line segments (i.e., $2M$) are incident at just one vertex, as the other endpoint of the line is on the boundary of the image. Therefore, set the number of incidences (i.e., the sum of the degrees of the vertices), which is four times the expected number of vertices (i.e., $4 \cdot \pi M^2/16$), to two times the number of edges (i.e., $2 \cdot S_{map}$) and solve for $\hat{M}$, which is equal to $\sqrt{8S_{map}/\pi}$.

The fourth estimator that we consider is termed a *d-based estimator*. This estimator is based on replacing the given map with a (usually smaller) random line image having the same number of vertices and average line segment length (termed *density* here). The expected segment length in the random line image is crudely approximated as the ratio between the expected length of the part of a random line included in the image and the expected number of vertices on the line. Using Theorem GP2 from geometric probability, we know that the expected length of the part of a random line included in the image is $\pi \cdot 2^N/4$, while the expected number of vertices in the map is $\pi M^2/16$. Since each vertex corresponds to the intersection of two lines and hence lies on two lines, the expected number of vertices per line is $(\pi M^2/16)/(2M) = \pi M/8$. Therefore, the expected segment length is $(\pi \cdot 2^N/4)/(\pi M/8) = 2 \cdot 2^N/M$. Equating this expected segment length to the average segment length of the given map, calculated simply as the ratio between the total length $L_{map}$ and the number of segments $S_{map}$ (i.e., $L_{map}/S_{map}$), and solving for $M$ yield an estimate $\hat{M}$ on the effective number of lines, which is equal to $2 \cdot S_{map} \cdot 2^N/L_{map}$.

Unlike instances of the random image model, real maps tend to be highly nonuniform, and, in particular, to have a large proportion of short segments and large empty "white" regions. This implies that the value of the effective number of lines $\hat{M}$ calculated for the d-based estimator above leads to node number estimates which are much higher than the actual ones. Therefore, we have chosen to compensate for this deviation by imposing the additional natural constraint that the total number of NSVs in the actual map is equal to the expected number of vertices in the random line images. This constraint, which was also used to obtain the effective number of lines for the V-based estimator, is now used as an area normalization factor to specify the nonuniform class of random images. In particular, every one of these images is equal to the random lines image in one region and is empty in the rest of it. The area of the "busy" part is specified to be $\frac{NSV_{map}}{(\pi \cdot \hat{M}^2/16)}$ and is usually smaller than 1. Note that the expected density remains the same. Since the random image model defined in section 3 is uniform, we can assume that the expected number of nodes representing every region is proportional to its area, and thus the node count is reduced by the aforementioned factor.

In order to estimate the number of nodes in the quadtrees representing the actual maps, we round the different values of the estimate $\hat{M}$ and insert this estimate into the basic sum (3) for the expected number of nodes together with the appropriate node splitting probability (which depends only on the quadtree type). These results are tabulated in Table 3. Notice that we do not tabulate the S-based estimator, as it is very similar to the V-based estimator, and the data bears this out. In particular, the S-based estimator relies on the number of line segments. This number is related to the number of vertices, which is the basis of the V-based estimator. Upper bounds, which have a more compact form and do not require the evaluation of a sum, may be

Table 3

*Predicted storage requirements for the MX, PM, and bucket PMR$_4$ quadtrees using the three estimators. The numbers given in the table are the ratios between the predicted requirements using the estimator and the actual ones given in Table 2.*

| Map Name | Depth | L-estimator | | | V-estimator | | | d-estimator | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | MX | PM | PMR$_4$ | MX | PM | PMR$_4$ | MX | PM | PMR$_4$ |
| Falls Church | 10 | 1.05 | 0.50 | 0.45 | 1.90 | 1.57 | 1.65 | 0.94 | 1.31 | 1.63 |
| Falls Church | 12 | 1.00 | 0.61 | 0.45 | 1.88 | 1.98 | 1.65 | 0.97 | 1.75 | 1.67 |
| Falls Church | 14 | 0.99 | 0.73 | 0.44 | 1.87 | 2.44 | 1.63 | 0.98 | 2.23 | 1.65 |
| Falls Church | 16 | 0.98 | 0.86 | 0.43 | 1.87 | 2.94 | 1.59 | 0.99 | 2.74 | 1.62 |
| Alexandria | 10 | 1.16 | 0.57 | 0.57 | 1.75 | 1.28 | 1.45 | 0.71 | 0.95 | 1.32 |
| Alexandria | 12 | 1.09 | 0.72 | 0.57 | 1.72 | 1.70 | 1.49 | 0.78 | 1.40 | 1.47 |
| Alexandria | 14 | 1.06 | 0.89 | 0.56 | 1.71 | 2.17 | 1.47 | 0.81 | 1.89 | 1.48 |
| Alexandria | 16 | 1.06 | 1.07 | 0.54 | 1.71 | 2.66 | 1.44 | 0.82 | 2.39 | 1.44 |
| Arlington | 10 | 1.18 | 0.54 | 0.55 | 1.77 | 1.22 | 1.44 | 0.78 | 0.91 | 1.30 |
| Arlington | 12 | 1.07 | 0.64 | 0.55 | 1.70 | 1.55 | 1.50 | 0.83 | 1.29 | 1.47 |
| Arlington | 14 | 1.03 | 0.79 | 0.54 | 1.69 | 1.97 | 1.47 | 0.87 | 1.73 | 1.47 |
| Arlington | 16 | 1.02 | 0.95 | 0.52 | 1.68 | 2.40 | 1.42 | 0.88 | 2.18 | 1.43 |
| Howard | 10 | 1.30 | 0.28 | 0.24 | 2.50 | 1.06 | 1.14 | 0.40 | 0.50 | 0.78 |
| Howard | 12 | 1.09 | 0.27 | 0.23 | 2.32 | 1.13 | 1.13 | 0.51 | 0.73 | 1.03 |
| Howard | 14 | 1.03 | 0.31 | 0.22 | 2.27 | 1.35 | 1.13 | 0.57 | 1.00 | 1.10 |
| Howard | 16 | 1.01 | 0.36 | 0.22 | 2.25 | 1.63 | 1.11 | 0.60 | 1.30 | 1.11 |
| DC | 10 | 1.26 | 0.69 | 0.67 | 1.74 | 1.37 | 1.51 | 0.86 | 1.03 | 1.33 |
| DC | 12 | 1.12 | 0.86 | 0.69 | 1.67 | 1.83 | 1.63 | 0.93 | 1.55 | 1.57 |
| DC | 14 | 1.09 | 1.10 | 0.67 | 1.66 | 2.41 | 1.61 | 0.97 | 2.14 | 1.59 |
| DC | 16 | 1.08 | 1.33 | 0.65 | 1.66 | 2.99 | 1.55 | 0.98 | 2.73 | 1.54 |
| Calvert | 10 | 1.22 | 0.22 | 0.18 | 2.15 | 0.70 | 0.68 | 0.13 | 0.19 | 0.32 |
| Calvert | 12 | 1.08 | 0.23 | 0.17 | 2.08 | 0.79 | 0.69 | 0.22 | 0.39 | 0.57 |
| Calvert | 14 | 1.04 | 0.29 | 0.17 | 2.07 | 1.02 | 0.70 | 0.27 | 0.64 | 0.67 |
| Calvert | 16 | 1.03 | 0.35 | 0.17 | 2.07 | 1.28 | 0.71 | 0.30 | 0.90 | 0.70 |
| Prince George's | 10 | 1.42 | 0.37 | 0.32 | 2.32 | 1.08 | 1.14 | 0.35 | 0.42 | 0.63 |
| Prince George's | 12 | 1.14 | 0.34 | 0.30 | 2.12 | 1.11 | 1.18 | 0.48 | 0.68 | 1.01 |
| Prince George's | 14 | 1.06 | 0.40 | 0.30 | 2.07 | 1.41 | 1.20 | 0.56 | 1.03 | 1.16 |
| Prince George's | 16 | 1.03 | 0.49 | 0.29 | 2.06 | 1.78 | 1.20 | 0.60 | 1.41 | 1.19 |
| Montgomery | 10 | 1.50 | 0.32 | 0.25 | 2.51 | 0.98 | 0.94 | 0.20 | 0.24 | 0.35 |
| Montgomery | 12 | 1.15 | 0.26 | 0.22 | 2.22 | 0.91 | 0.93 | 0.30 | 0.45 | 0.71 |
| Montgomery | 14 | 1.05 | 0.30 | 0.22 | 2.15 | 1.15 | 0.95 | 0.38 | 0.74 | 0.89 |
| Montgomery | 16 | 1.02 | 0.37 | 0.21 | 2.13 | 1.43 | 0.95 | 0.42 | 1.05 | 0.94 |

obtained by inserting the rounded estimate $\hat{M}$ directly into the upper bounds (15), (30), and (41), although we do not tabulate them here.

The particular estimators described have a number of inherent limitations. For example, suppose that the scale of the line segments (roads) of the map is lowered by a factor of 2 so that all the roads are totally embedded in the NW quadrant of the original map image, while the rest of the scaled map image is empty. In this case, if the depth is high enough, it is likely that the number of leaf nodes in PM and Bucket PMR$_4$ quadtrees will stay the same, while that in the MX quadtree will decrease significantly. However, the total length of the lines in the quadtree of the scaled-down map will be off by a factor of 2, thereby implying that the L-based estimator is likely to be inaccurate for the PM and bucket PMR$_4$ quadtrees. This is most relevant for maps containing many line segments in a small area and appears to dampen the suitability of the L-based estimator for arbitrary images, although it does seem to work for images that span most of the space in which they are embedded.

From our experiments, the L-based estimator seems to perform best for the MX quadtrees, while the V-based estimator seems to work best for the PM and PMR$_4$ quadtrees. We constructed the d-based estimator to try to improve further on the estimates for the PM and PMR$_4$ quadtrees but found that it does not improve on the V-based estimator, and sometimes even does worse. The good performance of the L-based estimator for the MX quadtree was not surprising, as it confirms the original result of the analysis of Hunter [22] and Hunter and Steiglitz [23], who found a proportionality to the perimeter of the image. The correlation between the estimated and actual values that we observed is noteworthy, considering that the number of nodes in the maps ranged between 77,000 and 38 million (i.e., a factor of 500). Nevertheless, we believe that a more detailed examination of the differences between the actual and predicted storage requirements, as well as other properties, of hierarchical spatial data structures is an extremely interesting open problem.

We were also interested in testing the validity of the asymptotic results given in (43) on the expected number of nodes as a function of the number of line segments in real images and the level of permitted subdivision. To do that, we interpret these results in terms of an actual map parameter—for example, by replacing the number of infinite lines $M$ with its estimate, as done above. For example, using the S-based estimator indicates that the size of the PMR quadtree should be proportional to the number of map segments and independent of the maximal depth (as long as both are large). Note that this expectation is verified (or refuted) only on the basis of the real map data and not on the basis of any modeling assumptions, although we were indeed led to it by the random line model analysis.

We first examine the MX quadtree. Figure 9 shows the ratios of the node count to the length of a side of the image (i.e., $2^N$) as a function of the depth (i.e., $N$) for the different maps, which are close to being constant (i.e., horizontal lines) as expected. Figure 10 shows the ratios of the node count to the square root of the number of line segments as a function of the number of line segments (i.e., $M^2$) for the different depths. These curves are close to being constant (i.e., horizontal lines) when the number of line segments is large enough. This is expected when the S-estimator (which stipulates that the number of nodes is proportional to the square root of the number of segments) is used for $M$ and implies that the asymptotic estimate is useful for predicting the quadtree size for a wide range of maps. We used a logarithmic scale in Figure 10 to illustrate a similar relative deviation in the ratios for the different depths as the size of the data increases.

Next, we examine the PM quadtree. Figures 11 and 12 show the ratio of the node count to the NSVs and to the number of line segments (i.e., $M^2$), respectively, as a function of the depth for the different maps, which are close to being constant (i.e., horizontal lines). This means that the node count is independent of the depth. This is contrary to the prediction of the asymptotic analysis and to the existing worst cases that arise when the vertices of the line segments are constrained to lie on grid points [47]. This difference may be explained by observing that, for images generated using the random image model, factors that lead to the maximum depth (e.g., two vertices or nonintersecting lines being very close to each other, or a vertex and a line being very close [49]) are more likely to arise. For road networks, on the other hand, it is unlikely that a pair of intersections 10 cm from each other will be specified. In such a case, these intersections will be merged to a higher degree vertex, which is not split for the PM quadtree. Thus, it seems that there is room for a better model for road networks (and maybe for other types of real data), which takes such merging processes into account. This subject is left for future research.

Finally, we examine the $\text{PMR}_4$ quadtree. Figures 13 and 14 show the ratio of the node count to the NSVs and to the number of line segments (i.e., $M^2$), respectively, as a function of the depth for the different maps, which we expect to be constant (i.e., horizontal lines), especially for the larger maps. At lower depths, the ratios increase with depth for a particular map since the segment counts are constant and the number of nodes does increase with depth until converging once the decomposition rule can

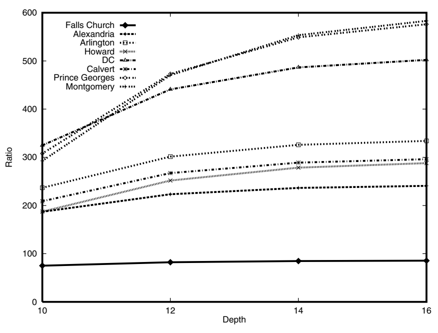

FIG. 9. *Ratio of MX quadtree nodes to side length.*

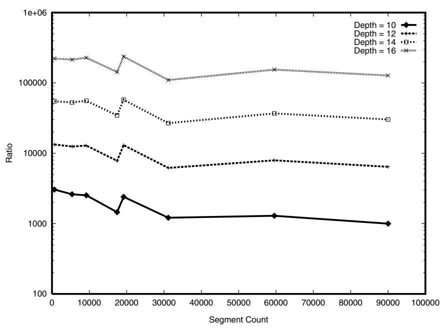

FIG. 10. *Ratio of MX quadtree nodes to square root of segment count.*

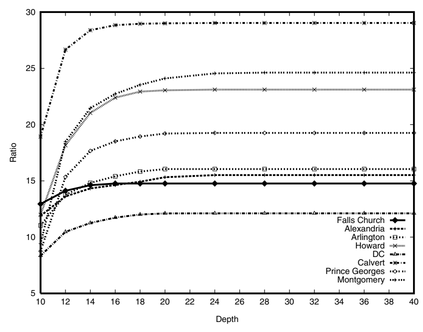

FIG. 11. *Ratio of PM quadtree nodes to NSV.*

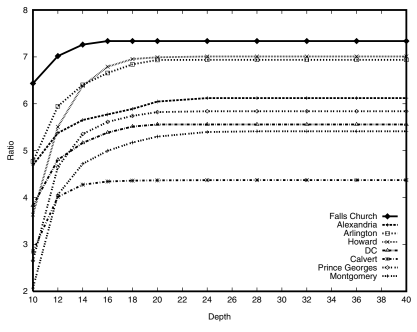

FIG. 12. *Ratio of PM quadtree nodes to the segment count.*

no longer be applied. It is interesting to observe that the lines in Figures 13 and 14 are not quite horizontal (i.e., representing a constant function) in the sense that they have a small positive slope. This is because the line segments in the maps are not really formed by random infinite lines. Thus, it is not true that the probability that more than two infinite lines intersect at a point is zero. In particular, we find that in our maps there are instances when more than four line segments meet at a point, and hence the number of nodes really grows linearly with depth (since the decomposition rule is still applicable, and, in fact, will always be applicable in this case), although this growth is not substantial in our graphs at higher depths. Experiments with larger values of $q$ verified that the number of nodes does in fact converge as the depth increases. This can be seen in Figure 15 for $q = 12$. Figure 5 shows the ratio of the node count to the segment count (i.e., $M^2$) versus the segment count at depth 16 for $q = 4$ and $q = 12$. The ratios are all within 6% of their average value. Figure 5 also reveals a general trend in which the ratios decrease as the maps get larger. We used a logarithmic scale to illustrate a similar relative deviation in the ratios for the different values of $q$ as the size of the maps increases.

To more clearly see the effect of the existence of points, where more than four line segments intersect, consider the map of Washington, D.C. (Figure 8). From Figure 11, we can see that the number of nodes increases by a ratio of about 7:4 when the depth is changed from 10 to 40. From Table 2, we can see that this amounts to an increase of about $(7/4 - 1) \cdot 35,000 = 26,250$ vertices. Now, let $W$ be the number of

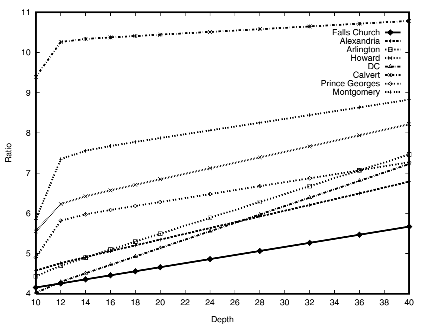

FIG. 13. *Ratio of PMR quadtree nodes to NSV, $q = 4$.*

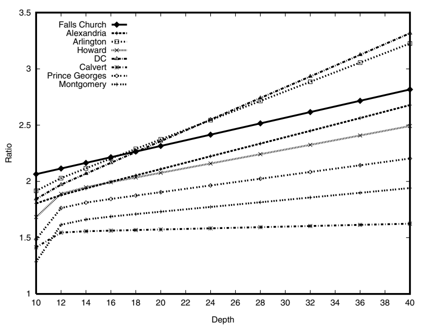

FIG. 14. *Ratio of PMR quadtree nodes to segment count, $q = 4$.*

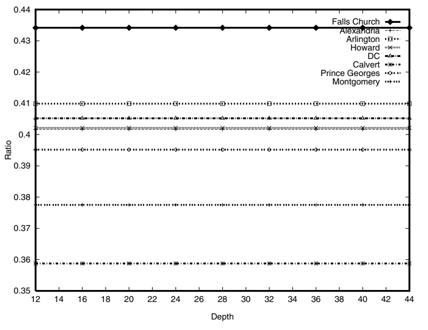

FIG. 15. *Ratio of PMR quadtree nodes to segment count, $q = 12$.*

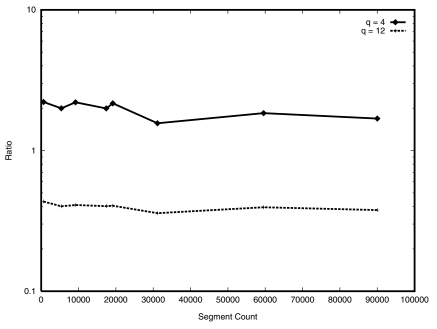

FIG. 16. *Ratio of PMR quadtree nodes to segment count versus segment count at a depth of* 16.

vertices associated with the intersection of more than four roads. Each such vertex leads to a split of its corresponding region until the maximal depth is reached. At every depth, four nodes are added. Therefore, when the depth is changed from 10 to 40, the number of nodes that are added is $W \cdot (40 - 10) \cdot 4 = 26,250$, implying that the number of such high degree vertices is about 220. Given that the Washington, D.C. map has 8805 NSVs (i.e., vertices of degree greater than 20), the fraction of such high degree vertices is about $220/8805 = 0.025$. While this number is larger than 0 (the value predicted by our model), it does not appear to change the predictions by much, as is apparent from the relatively modest increase in the node count as the depth increases. Note that for the other maps considered the relative increase in the node count, as well as the fraction of high degree vertices, is much lower. Observe also that the range of maximum tree depths considered in these graphs is much larger than in any reasonable application. In particular, if the original map is embedded in a $100 \times 100$ km square area, then a maximum depth of 40 amounts to a resolution of 0.1 micron. Therefore, for a more reasonable maximum depth, the variation in the predicted node count will be much lower even if, say, 0.025 of the vertices have high degree.

**6. Concluding remarks.** The analysis of the space requirements of a number of trie-based hierarchical geometric data structures for storing large collections of line segments was investigated using a random image model. An appropriate model was developed for each of these structures and estimates of $E[S]$, the expected number of nodes, were found for them. Future work includes the investigation of the use of these estimates in a cost model by a query optimizer to generate an appropriate query evaluation plan in a spatial database application. The analysis presented here is also of interest because it uses a detailed explicit model of the image, instead of relying on modeling the branching process represented by the tree and leaving the underlying image unspecified. The behavior of these expected values is intuitively and concisely expressed by analytic upper and lower bounds. Other directions for future research include the application of the geometric probability approach to additional data types besides line segments (e.g., points, polygons, surfaces, solids, etc.), as well as alternative trie-based spatial data structures.

We have demonstrated that these estimates, derived for a particular random image model, are applicable to real data. Specifically, in the case of line map images, we provided estimators which are based on simple characterizations of the map data, and which enabled us to successfully apply the results of the analytic model to real data and to obtain reasonably accurate and useful results. This was verified, however, only for map data, and characterizing collections of other types of line segments, or even more general types of spatial information, is still an open problem, and thus a subject for further research.

Our results can be used to justify claims on the qualitative differences between the different alternative spatial data structures. For example, we showed that the bucket (and conventional) $\text{PMR}_q$ quadtree for $q \geq 4$ is superior to the PM quadtree in terms of the number of nodes that are required. The problem with the PM quadtree is that, although its behavior is usually acceptable, there are cases in which it requires much space due to certain point and line configurations. This follows from our analysis and simulations, as well as from confirming earlier observations on the possible worst-case behavior of the PM quadtree [47].

Perhaps our most important result is showing that the space requirements of the Bucket $\text{PMR}_q$ and $\text{PMR}_q$ ($q \geq 4$) quadtrees are asymptotically proportional to the number of line segments. This was shown theoretically for a random image model and was also found to hold for random data and real map data. This is quite significant as it enables us to predict the number of nodes required by this representation, and, most important, to show that it is independent of the maximum depth of the tree. It is thus not surprising that the $\text{PMR}_q$ quadtree is useful in experimental systems (e.g., QUILT [53]) as well as commercial systems (e.g., United Parcel Service (UPS) [4]).

**Acknowledgments.** We thank S. K. Bhaskar and Gary D. Knott for suggestions about the evaluation of the sums.

## REFERENCES

[1] C. H. Ang, *Applications and Analysis of Hierarchical Data Structures*, Tech. report TR-2255, Department of Computer Science, University of Maryland, College Park, MD, 1989.

[2] W. G. Aref and H. Samet, *Optimization strategies for spatial query processing*, in Proceedings of the 17th Annual International Conference on Very Large Data Bases (VLDB), G. Lohman, ed., Barcelona, Spain, September 1991, pp. 81–90.

[3] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger, *The $R^*$-tree: An efficient and robust access method for points and rectangles*, in Proceedings of the ACM SIGMOD Conference, Atlantic City, NJ, June 1990, pp. 322–331.

[4] R. Bonefas, *Personal communication*, 1991.

[5] R. de la Briandais, *File searching using variable-length keys*, in Proceedings of the IRE/IEEE/ACM Western Joint Computer Conference, San Francisco, CA, 1959, pp. 295–298.

[6] T. Brinkhoff, H. P. Kriegel, R. Schneider, and B. Seeger, *Multi-step processing of spatial joins*, in Proceedings of the ACM SIGMOD Conference, Minneapolis, MN, June 1994, pp. 197–208.

[7] Bureau of the Census, *TIGER/Line Census Files*, 1990 Technical Documentation, Washington, DC, 1991.

[8] C. R. Dyer, *The space efficiency of quadtrees*, Comput. Graphics Image Process., 19 (1982), pp. 335–348.

[9] C. Faloutsos and I. Kamel, *Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension*, in Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Minneapolis, MN, May 1994, pp. 4–13.

[10] C. Faloutsos, H. V. Jagadish, and Y. Manolopoulos, *Analysis of the n-dimensional quadtree decomposition for arbitrary hyperectangles*, IEEE Trans. Knowledge Data Engrg., 9 (1997), pp. 373–383.

[11] C. Faloutsos, T. Sellis, and N. Roussopoulos, *Analysis of object oriented spatial access methods*, in Proceedings of the ACM SIGMOD Conference, San Francisco, CA, May 1987, pp. 426–439.

[12] R. A. Finkel and J. L. Bentley, *Quad trees: A data structure for retrieval on composite keys*, Acta Inform., 4 (1974), pp. 1–9.

[13] P. Flajolet, G. Gonnet, C. Puech, and J. M. Robson, *Analytic variations on quadtrees*, Algorithmica, 10 (1993), pp. 473–500.

[14] P. Flajolet and T. Lafforge, *Search costs in quadtrees and singularity perturbation asymptotics*, Discrete Comput. Geom., 12 (1994), pp. 151–175.

[15] P. Flajolet and C. Puech, *Partial match retrieval of multidimensional data*, J. ACM, 33 (1986), pp. 371–407.

[16] E. Fredkin, *Trie memory*, Comm. ACM, 3 (1960), pp. 490–499.

[17] G. Graefe, *Query evaluation techniques for large databases*, ACM Comput. Surveys, 25 (1993), pp. 73–170.

[18] A. Guttman, *R-trees: A dynamic index structure for spatial searching*, in Proceedings of the ACM SIGMOD Conference, Boston, MA, June 1984, pp. 47–57.

[19] E. G. Hoel and H. Samet, *Data-parallel spatial join algorithms*, in Proceedings of the 23rd International Conference on Parallel Processing, vol. 3, St. Charles, IL, 1994, pp. 227–234.

[20] E. G. Hoel and H. Samet, *Performance of data-parallel spatial operations*, in Proceedings of the 20th Annual International Conference on Very Large Data Bases (VLDB), Santiago, Chile, September 1994, pp. 156–167.

[21] M. Hoshi and P. Flajolet, *Page usage in a quadtree index*, BIT, 32 (1992), pp. 384–402.

[22] G. M. Hunter, *Efficient Computation and Data Structures for Graphics*, Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ, 1978.

[23] G. M. Hunter and K. Steiglitz, *Operations on images using quad trees*, IEEE Trans. Pattern Anal. Mach. Intell., 1 (1979), pp. 145–153.

[24] C. L. Jackins and S. L. Tanimoto, *Oct-trees and their use in representing three dimensional objects*, Comput. Graphics Image Process., 14 (1980), pp. 249–270.

[25] G. Kedem, *The quad-CIF tree: A data structure for hierarchical in-line algorithms*, in Proceedings of the ACM/IEEE 19th Annual Design Automation Conference, Las Vegas, NV, June 1982, pp. 352–257.

[26] A. Klinger, *Patterns and search statistics*, in Optimizing Methods in Statistics, J. S. Rustagi, ed., Academic Press, New York, 1971, pp. 303–337.

[27] K. Knowlton, *Progressive transmission of grey-scale and binary pictures by simple, efficient, and lossless encoding schemes*, Proc. IEEE 68 (1980), pp. 885–896.

[28] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching*, vol. 3, 2nd ed., Addison-Wesley, Reading, MA, 1998.

[29] J. H. Lee, D. H. Kim, and C. W. Chung, *Multi-dimensional selectivity estimation using compressed histogram information*, in Proceedings of the ACM SIGMOD Conference, Philadelphia, PA, June 1999, pp. 205–214.

[30] C. Mathieu, C. Puech, and H. Yahia, *Average efficiency of data structures for binary image processing*, Inform. Process. Lett., 26 (1987), pp. 89–93.

[31] Y. Matias, J. S. Vitter, and M. Wang, *Wavelet-based histograms for selectivity estimation*, in Proceedings of the ACM SIGMOD Conference, Seattle, WA, June 1998, pp. 448–459.

[32] D. Meagher, *Geometric modeling using octree encoding*, Comput. Graphics Image Process., 19 (1982), pp. 129–147.

[33] B. Moon and J. H. Saltz, *Scalability analysis of declustering methods for multidimensional range queries*, IEEE Trans. Knowledge Data Engrg., 10 (1998), pp. 310–327.

[34] M. Muralikrishna and D. J. DeWitt, *Equi-depth histograms for estimating selectivity factors for multi-dimensional queries*, in Proceedings of the ACM SIGMOD Conference, Chicago, IL, June 1988, pp. 28–36.

[35] R. C. Nelson and H. Samet, *A consistent hierarchical representation for vector data*, Comput. Graphics, 20 (1986), pp. 197–206.

[36] R. C. Nelson and H. Samet, *A Population Analysis of Quadtrees with Variable Node Size*, Tech. report TR-1740, Department of Computer Science, University of Maryland, College Park, MD, December 1986.

[37] R. C. Nelson and H. Samet, *A population analysis for hierarchical data structures*, in Proceedings of the ACM SIGMOD Conference, San Francisco, CA, May 1987, pp. 270–277.

[38] J. A. Orenstein, *Spatial query processing in an object–oriented database system*, in Proceedings of the ACM SIGMOD Conference, Washington, DC, May 1986, pp. 326–336.

[39] M. Ouksel and P. Scheuermann, *Storage mappings for multidimensional linear dynamic hashing*, in Proceedings of the ACM SIGACT–SIGMOD Symposium on Principles of Database Systems (PODS), Atlanta, GA, March 1983, pp. 90–105.

[40] B. U. Pagel, H. W. Six, H. Toben, and P. Widmayer, *Towards an analysis of range query performance in spatial data structures*, in Proceedings of the 12th Annual ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Washington, DC, May 1993, pp. 214–221.

[41] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd ed., McGraw Hill, New York, 1991.

[42] C. Puech and H. Yahia, *Quadtrees, octrees, hyperoctrees: A unified analytical approach to three data structures used in graphics, geometric modeling and image processing*, in Proceedings of the ACM Symposium on Computational Geometry, Baltimore, MD, June 1985, pp. 272–280.

[43] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, MA, 1990.

[44] H. Samet, *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*, Addison-Wesley, Reading, MA, 1990.

[45] H. Samet, *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann, San Francisco, CA, 2006.

[46] H. Samet, A. Rosenfeld, C. A. Shaffer, R. C. Nelson, and Y. G. Huang, *Application of hierarchical data structures to geographical information systems: Phase* III, Tech. report TR-1457, Department of Computer Science, University of Maryland, College Park, MD, 1984.

[47] H. Samet, C. A. Shaffer, and R. E. Webber, *Digitizing the plane with cells of non-uniform size*, Inform. Process. Lett., 24 (1987), pp. 369–375.

[48] H. Samet and M. Tamminen, *Efficient component labeling of images of arbitrary dimension represented by linear bintrees*, IEEE Trans. Pattern Anal. Mach. Intell., 10 (1988), pp. 579–586.

[49] H. Samet and R. E. Webber, *Storing a collection of polygons using quadtrees*, ACM Trans. Graphics, 4 (1985), pp. 182–222.

[50] L. A. Santalo, *Integral Geometry and Geometric Probability*, Encyclopedia of Math. Appl. 1, Addison-Wesley, Reading, MA, 1976.

[51] C. A. Shaffer, *A formula for computing the number of quadtree node fragments created by a shift*, Pattern Recognition Lett., 7 (1988), pp. 45–49.

[52] C. A. Shaffer, R. Juvvadi, and L. S. Heath, *Generalized comparison of quadtree and bintree storage requirements*, Image Vision Comput., 11 (1993), pp. 402–412.

[53] C. A. Shaffer, H. Samet, and R. C. Nelson, *QUILT: A geographic information system based on quadtrees*, Int. J. Geographical Inform. Systems, 4 (1990), pp. 103–131.

[54] M. Tamminen, *Encoding pixel trees*, Comput. Vision Graphics Image Process., 28, (1984), pp. 44–57.

[55] M. Tamminen, *Comment on quad- and octtrees*, Comm. ACM, 27 (1984), pp. 248–249.

[56] M. Vassilakopoulos and Y. Manolopoulos, *Analytical results on the quadtree storage-requirements*, in Proceedings of the 5th International Conference on Computer Analysis of Images and Patterns (CAIP '93), D. Chetverikov and W. G. Kropatsch, eds., Lecture Notes Comput. Sci. 719, Springer-Verlag, Berlin, 1993, pp. 41–48.