A Probabilistic Analysis of Trie-Based Sorting of Large Collections of Line Segments in Spatial Databases*

Michael Lindenbaum
Computer Science Department
Technion
32000 Haifa, ISRAEL

Hanan Samet
Gisli R. Hjaltason
Computer Science Department
University of Maryland
College Park, Maryland 10742 USA

February 17, 2000

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 two ways:

1. They reveal the properties of the various representations and permit their comparison using analytic, non-experimental, 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., ≥ 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 only been 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.

Keywords and phrases: large spatial databases, tries, sorting line segments, geometric probability, analysis of algorithms, spatial data structures, quadtrees, quadtries, cost model, query evaluation

*This work was supported in part by the National Science Foundation under Grant IRI-9712715.

1 Introduction

1.1 Background

The efficient management of data in large database systems depends on grouping the data in such a way so that similar data is aggregated or stored in proximity and hence can be operated upon together (e.g., [14]). This grouping is achieved by either sorting or hashing 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. The rationale for hashing is the fact that set processing algorithms based on it have an expected complexity of $O(N)$ instead of $O(N \log N)$ as in sorting.

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 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 the data is sorted with respect to a particular reference point (e.g., all 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 data. The problem is that the data was sorted in an explicit manner. Instead, we need methods which provide an implicit ordering. Examples of such techniques are called *bucketing methods* (e.g., [38]). 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. This method is analogous to hashing where the hashing function is said to be order-preserving (e.g., [35]). Thus we see that for spatial data there is really no distinction between using sorting or hashing to achieve grouping.

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 the hierarchy (e.g., members of the R-tree family [3, 15]). 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 [39, 38]). The decomposition can be either tree-based or trie-based. The distinction is that the former is applied to the values of the data, while the latter makes use of the digits (termed a *trie* [13, 25]) 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* [25].

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 but different trie-based data structures may be created depending on whether we test one bit at each iteration (a *k-d trie* [12]) or two bits at each iteration (a *quadtrie* [18, 38]). 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 (e.g., [23, 39, 38]). 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 (e.g., [10, 11]).

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 [39, 38]. In order to be able 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 datasets 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 [5] 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 (e.g., [36]). 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 (e.g., [26, 28, 30]) 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 cost for the spatial data structure.

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 makes use of a uniform distribution in the underlying space (e.g., [2, 8]). An alternative is to make use of a non-uniform distribution. Some techniques that have been used include the Gaussian distribution [32] as well as clustering methods uniformly distributed points [34] or even pre-determined shapes [3]. Another approach makes use of a fractal distribution [7] 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 which has extent such as collections of line segments. Tamminen [46] considers the performance of quadtrees and bintrees under the assumption that the image consists of a single random line treated as a region, and analyzes the number of nodes in them using geometric probability. Shaffer et al. [44] 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. Mathieu et al. [27] and Puech and Yahia [37] investigate the size of quadtree representations of region data and some other related questions using some assumptions on the branching probabilities of nodes in the tree. Nelson and Samet [31, 32, 33] consider the distributions of node occupancies in hierarchical geometric data structures that store a variable number of geometric data items per node which included points and lines but have a wider applicability. This approach is similar to hashing [25] where each node acts like a bucket.

Although these approaches sometimes lead to remarkable agreement between theory and simulation (e.g., [1, 32]), they have a common drawback. The explicit model of the image on which the statistical analysis is done is either exceedingly simple or is not given at all in which case it must be implied from other assumptions. In particular, an assumption is made on the splitting probability in the data structure (e.g., [1, 27, 32, 37]) which implies the existence

[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 [7]. This was used to predict the effective occupancy of nodes in an R-tree that stores point data.

of some *implicit* model on the data. However, when we want to know what kind of data (real or contrived) fits this model, the only possible answer is a circular one that says that the data is such that gives 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* non-trivial random image model, and then to 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 corresponds 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 [12] who analyzed the partial match query time for hierarchical data structures, while we analyze their storage requirements. Unlike their data, which consists of random points in a high dimensional space whose coordinate values are drawn from a uniform distribution, our data consists of randomly drawn lines. It is important to note that lines are a qualitatively different data type than points, as the action of every line on the structure is not local.

An alternative non-statistical approach was applied by Hunter and Steiglitz [19, 20] to show that for a polygon of perimeter l, the size (i.e., the number of nodes) of the corresponding region quadtree 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 [41].

As we pointed above, in this paper we investigate the use of a random image model consisting of M randomly drawn lines. Unlike Tamminen's approach [46], 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 five variants of the quadtree that can be built for data that obey this model by determining the expected number of nodes in each of them. These variants are the region quadtree [23, 20], the MX quadtree [20, 38], the PM quadtree [42], the PMR quadtree [31, 32, 33], and a new variant of the PMR quadtree representation suggested here which we call a Bucket PMR quadtree (see also [16, 17]).

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 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., ≥ 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 [19, 20, 31, 42] and agrees with the extensive experimentation that we have carried out. This verification and the accompanying theoretical justification is one of the contributions of our research.

The agreement between the results of our analysis and the data was not a surprise in the case of the MX quadtree in that it confirmed previous results (i.e., [19, 20]). However, in the case of the PMR and Bucket PMR quadtree for sufficiently high values of the splitting threshold (i.e., ≥ 4) our analysis breaks new ground in that 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 five 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 afore-mentioned quadtree variants. It also contains the results of some experiments with instances of the random lines model. Section 5 describes the application of the analysis to predict the storage requirements for real data as well as the results of extensive experiments that support its validity. Section 6 draws some concluding remarks as well as gives some directions for future research.

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 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 sons of each node represent a partition of that region into four parts. This scheme is useful for representing geometric data at a variable resolution. Quadtree variants exist for representing planar regions [23], collections of points [9, 40], and collections of line segments [31, 32, 33, 42], as well as more complicated objects (e.g., rectangles [22]). Generalization of the quadtree to three and higher dimensions (e.g., octrees [19, 21, 29] and bintrees [24, 41, 47]) have also been investigated. They 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 pre-defined boundaries (i.e., trie-based), and those where the partition is determined explicitly by the data as it is entered into the data structure (i.e. tree-based or data-based). In most cases, use of regular decomposition means that the shape of the resulting data structure is independent of the order in which the data is inserted into the structure when building it. This is not the case for data-based decompositions. In fact, for most applications, regular decomposition works at least as well as the data-based decomposition. Moreover, regular decomposition is easier to implement and analyze. In this paper, we consider only structures based on a regular decomposition. Another distinction is between quadtree variants whose maximum depth (say N) is bounded and those for whom it is not. For the structures that we consider here, only the PMR quadtree has an unbounded maximum depth.

The condition used to determine when a quadtree block should be partitioned is called a *splitting rule*. This rule is usually a function of the data that is associated with the block — that is, the condition is evaluated using local information. The exact formulation of the rule depends on the type of the data being stored. Here we consider the use of the following quadtree variants in the representation of a collection of line segments.

- A region quadtree is usually used to represent planar regions and its splitting rule is such that a block is split if both the depth of its corresponding node is smaller than N and if the region represented by the block is not homogeneous (see Figure 1a which describes the subdivision implied by this quadtree).

- An MX quadtree, although usually defined for points (e.g., [38]), can be used to represent a collection of line segments on the plane. Its splitting rule is such that 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 (see Figure 1b).

- A PM quadtree represents collections of line segments in the plane. It is a vertex-based representation whose splitting rule is that a block is split unless the depth of the corresponding node is N, or only one line segment passes through the block, or all the line segments that pass through the block meet at a point within the block, or the block contains more than one endpoint of a line segment. This is a variant of the PM_1 quadtree [42]. It differs from the PM_1 quadtree by virtue of having a bound on its depth (see Figure 1c).

- We also define a new hierarchical structure called a *Bucket PMR quadtree* which is similar to a PM quadtree with the difference that it is edge-based rather than vertex-based as was the PM quadtree. The Bucket PMR quadtree represents collections of line segments on the plane using a splitting rule such 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. In this case, q is called a *bucket capacity* as this is its role since a block is split as long as it has more that q line segments in it and is not at the maximum depth of the tree. This structure has a prespecified maximum depth (i.e., N). The resulting decomposition does not depend on the order in which the line segments are inserted, and the number of line segments stored in a leaf node whose depth is less than N is guaranteed to be less than or equal to q (see Figure 1d where $q = 4$). At times, we want to express the dependence on q explicitly and use the term *Bucket* $\textit{PMR}_q$ *quadtree* to describe this structure.

- The disadvantage of the Bucket PMR quadtree is that if there are many line segments in one small region, then there would be much decomposition in that region. An alternative representation, termed a *PMR quadtree*, also represents a collection of line segments but is defined in a slightly different way. The data structure is created as a dynamic result of a sequence of insertions of line segments using a splitting rule such that a block is split once, and only once, if the block is both intersected by the new line segment and already contains q or more line segments. In this case, the parameter q is termed a *splitting threshold* to distinguish its use from that in the Bucket PMR quadtree. Note that this structure does not have a prespecified maximum depth. The resulting decomposition depends on the order in which the line segments are inserted. Moreover, the number of line segments represented by any leaf node is not guaranteed to be less than or equal to 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 road maps, rivers, etc.) do not cause a split. Figure 1e illustrates this representation assuming that $q = 4$ and that the lines are inserted in the order marked in the figure. It is important to note that in this example no split takes place until the fifth line segment is inserted, and since only one split has occurred, the `SE` quadrant contains $5 > q$ line segments. Again, as in the Bucket PMR quadtree, at times, we want to express the dependence on q explicitly and use the term $\textit{PMR}_q$ *quadtree*.

A subtle point in the definition of the splitting rules is whether two line segments that intersect in a square should be counted as four line segments or not. The algorithms that create the data structures take a list of line segments as

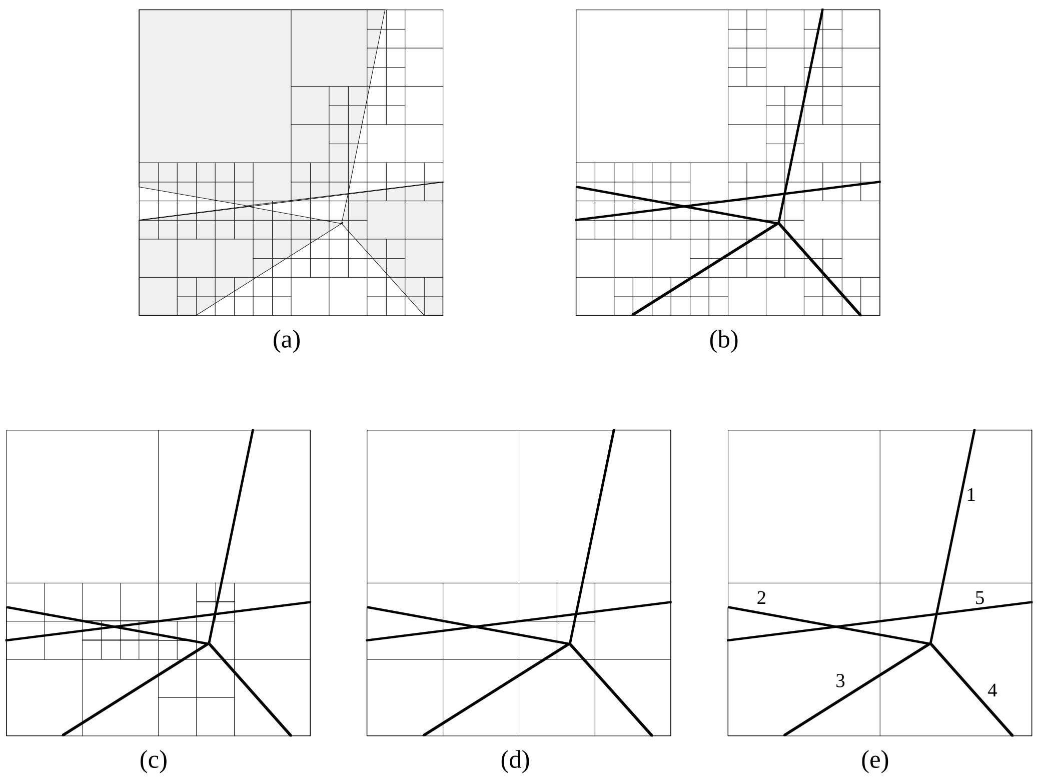


Figure 1: Five types of quadtrees: (a) region quadtree, (b) MX quadtree, (c) PM quadtree, (d) Bucket PMR$_4$ quadtree. (e) and PMR$_4$ quadtree when the lines are inserted in the order 1, 2, 3, 4, and 5.

their input and are not provided with the explicit information on their geometrical intersection. Therefore, every line segment is specified by its two original endpoints and two line segments remain two even if they intersect. Therefore, the data represented in Figure 1b–e consists of exactly five line segments. In fact, this is why in the case of the PM quadtree, we had to get to the maximum depth by decomposing the SW and SE sons of the NE son of the SW quadrant in Figure 1c and likewise for the NE son of the NW son of the SE quadrant in Figure 1c.

3 Random Image Models

In the first part of this paper we assume 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 2a). Although the position of every particular line, $L(\rho,\theta)$, naturally depends on the origin and orientation of the coordinate system, we shall see soon 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 2a illustrates this invariance property. For the rectangular region R

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

let T be the parameter set

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

which includes all the parameter pairs (ρ,θ) 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 ρ and θ are defined (i.e., it is independent of the translation or rotation of the coordinate system [43]). Therefore, it is the natural density function to specify when modeling collections of random lines. As an illustration, see Figure 2b where an instance of the random image containing five lines is described.

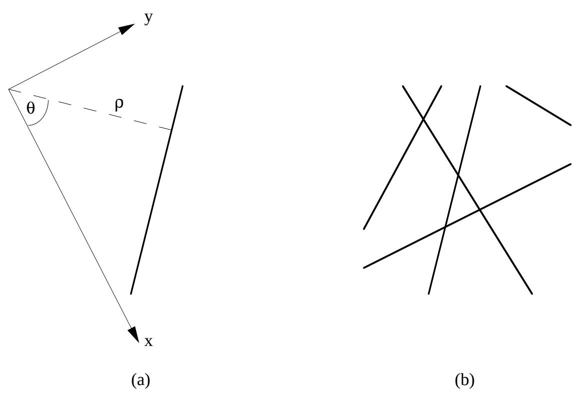


Figure 2: The Random Image Model: (a) The process of generating a random line (note that the coordinate system is arbitrary with respect to the image). (b) A typical instance of the random image constructed for M=5 independently drawn lines.

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 also obeyed by real spatial data such as road maps as a junction of more than four roads is rare. 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) is 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 just 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 correlates with what appears in realistic geometric applications such as road networks. Finding such a correlation is unlikely as realistic geometric data does 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 importantly, the analysis provides us a means to justify claims about the relative qualitative behavior of the different data structures.

The above model is useful for predicting the sizes of all but the region quadtree. The region quadtree represents regions rather than lines segments. This leads us to the following variation of the above model, which we term a *modified* random image model. In this case, each image consists of black and white regions separated by M random lines chosen independently using the uniform density distribution given by (1). The actual colors of the individual regions do not affect the total number of nodes. The resulting image can be interpreted as follows. Choose one of the regions at random and let it be black. Let all of its neighbors be white. Let all of the white node's uncolored neighbors be black. Repeat this process until all regions are colored.

Before starting the analysis, we first present three results from geometric probability which form the basis of our results [43].

Geometric Probability Theorem 1 (GPT1): 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 (GPT2): 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 (GPT3): 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, are random events. We make the following simplifying assumptions: given nodes u and v which exist in the tree, the events that node u is a leaf and that node v is a leaf are statistically independent. Assume further that the conditional probability

$$P_d = p\{v \text{ is a non-leaf node } | v \text{ is a node of the tree}\}$$

depends only on d, the depth of v in the tree. Although the statistical independence assumption does not necessarily hold for all pairs of nodes (e.g., two nodes which are brothers), this is usually the case for most pairs of nodes which correspond to small distant regions intersected by different lines. As we shall see later, the independence assumption, as well as the expressions for the storage requirements which depend on it, hold only when the number of lines is not too small. If this number is too small, then more pairs are intersected by the same line, and the nodes that split no longer correspond to independent events.

Let P'_d denote the probability that the splitting condition is satisfied for a particular square region which corresponds to a node at depth d. Considering the MX quadtree, for example, P'_d is the probability that at least one line pass through this region. The existence of a node v at depth d in the tree implies that the splitting condition holds for the region corresponding to its father node. Thus, v exists in the tree with probability P'_{d-1} and, for $d \geq 1$,

$$P_d = p\{v \text{ is a non-leaf node } | v \text{ exists}\} = \frac{p\{v \text{ has 4 sons}\}}{p\{v \text{ exists}\}} = \frac{P'_d}{P'_{d-1}}. \tag{2}$$

Both P_0 and P'_0 are one for every tree which describes at least one line segment. Let S_d be the number of nodes at depth d, and let S be the total number of nodes in the tree. The expected number of nodes at each depth, as implied by the statistical independence assumption, is given by

$$\begin{aligned} E[S_0] &= S_0 = 1 \\ E[S_1] &= P_0 \cdot E[S_0] \cdot 4 = P_0 \cdot 4 \\ E[S_2] &= P_1 \cdot E[S_1] \cdot 4 = P_0 P_1 \cdot 4^2 \\ E[S_3] &= P_2 \cdot E[S_2] \cdot 4 = P_0 P_1 P_2 \cdot 4^3 \\ &\cdots \\ E[S_d] &= P_{d-1} \cdot E[S_{d-1}] \cdot 4 = \left[\prod_{i=0}^{d-1} P_i\right] \cdot 4^d \end{aligned} \tag{3}$$

and

$$\begin{aligned} E[S] = \sum_{d=0}^{N} E[S_d] &= 1 + \sum_{d=1}^{N} 4^d \cdot \prod_{i=0}^{d-1} P_i \\ &= 1 + P_0 \sum_{d=1}^{N} 4^d \cdot \frac{P'_1}{P'_0} \cdot \frac{P'_2}{P'_1} \cdot \frac{P'_3}{P'_2} \cdots \frac{P'_{d-1}}{P'_{d-2}} \\ &= 1 + \sum_{d=1}^{N} 4^d \cdot P'_{d-1}. \end{aligned} \tag{4}$$

Equation (4), 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 Equation (4) 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 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 4 sons (see Figure 1b).

4.1.1 Analysis

In order to compute the probabilities P_d, we use the following argument. A node at depth d corresponds to a $2^{N-d} \times 2^{N-d}$ square. The Geometric Probability Theorem GPT1 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

$$p_{d,k} = \binom{M}{k} \left(\frac{1}{2}\right)^{dk} \left[1 - \left(\frac{1}{2}\right)^d\right]^{M-k} . \tag{5}$$

The probability that one or more lines pass through this region, thereby implying that it is split, is

$$P'_d = 1 - p_{d,0} = 1 - \left[1 - \left(\frac{1}{2}\right)^d\right]^M . \tag{6}$$

Inserting (6) into (4) we get

$$E[S] = 1 + \sum_{d=1}^{N} 4^d \cdot P'_{d-1} = \sum_{d=1}^{N} 4^d \left[1 - \left[1 - \left(\frac{1}{2}\right)^{d-1}\right]^M\right] . \tag{7}$$

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, as well as consulting their developers. 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 to insert the known parameters (i.e., M and N) into the sum (7) and to evaluate it numerically. These comments are also applicable in the analyses of the rest of the data structures (see Sections 4.3.1 and 4.4).

Note that although the form of the sum (7) 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 (7) from above which means that the term $\left[1 - \left(\frac{1}{2}\right)^{d-1}\right]^M$ should be bound from below and this is not possible with this inequality.

Our technique is based on decomposing (7) 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 additional parameter β (as well as M and N) which is defined as

$$\beta = M \left(\frac{1}{2}\right)^{d_0} . \tag{8}$$

The depth d_0 is chosen so that β 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 β. 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 constraint 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

$$\begin{aligned} E[S] &= 1+\sum_{d=1}^{d_0} 4^d \left[1-\left[1-\left(\frac{1}{2}\right)^{d-1}\right]^M\right] + \sum_{d=d_0+1}^{N} 4^d \left[1-\left[1-\left(\frac{1}{2}\right)^{d-1}\right]^M\right] \\ &= \textstyle\sum_1+\sum_2. \end{aligned} \tag{9}$$

$$\textstyle\sum_1 = 1+\displaystyle\sum_{d=1}^{d_0} 4^d \left[1-\left[1-\left(\frac{1}{2}\right)^{d-1}\right]^M\right] \leq \sum_{d=0}^{d_0} 4^d \cdot 1 \approx \frac{4^{d_0}}{1-\frac{1}{4}} \approx \frac{4}{3}\frac{M^2}{\beta^2}. \tag{10}$$

$\sum_1$ reflects the fact that we assume that the tree is complete at depth less than or equal to d_0. Taking the binomial expansion of $\sum_2$ we get

$$\begin{aligned} \textstyle\sum_2 &= \sum_{d=d_0+1}^{N} 4^d \left[1-\sum_{k=0}^{M}\left(\frac{1}{2}\right)^{kd-k}\binom{M}{k}(-1)^k\right] \\ &= \sum_{d=d_0+1}^{N}\sum_{k=1}^{M} 2^{2d}\left(\frac{1}{2}\right)^{kd-k}\binom{M}{k}(-1)^{k-1}. \end{aligned} \tag{11}$$

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

$$\textstyle\sum_2 = \sum_3+\sum_4+\sum_5 \tag{12}$$

$$\textstyle\sum_3 = \displaystyle\sum_{d=d_0+1}^{N} 2^{2d}\left(\tfrac{1}{2}\right)^{d-1}\tbinom{M}{1} = \sum_{d=d_0+1}^{N} 2^d\cdot 2\cdot M = 4M(2^N-2^{d_0}) = 4M\cdot 2^N - \tfrac{4M^2}{\beta} \tag{13}$$

$$\textstyle\sum_4 = -\displaystyle\sum_{d=d_0+1}^{N} 2^{2d}\left(\frac{1}{2}\right)^{2d-2}\binom{M}{2} = -\sum_{d=d_0+1}^{N} 4\binom{M}{2} = -2M(M-1)(N-d_0) \tag{14}$$

$$\begin{aligned} \textstyle\sum_5 &= \sum_{d=d_0+1}^{N}\sum_{k=3}^{M} 2^k\binom{M}{k}\left(\frac{1}{2}\right)^{(k-2)d}(-1)^{k-1} \\ &\leq \sum_{k=3}^{M}\sum_{d=d_0+1}^{N} 2^k\binom{M}{k}\left(\frac{1}{2}\right)^{(k-2)d} \\ &\approx \sum_{k=3}^{M} 2^k\binom{M}{k}\frac{(\frac{1}{2})^{d_0(k-2)}(\frac{1}{2})^{k-2}}{1-(\frac{1}{2})^{k-2}} \\ &= \sum_{k=3}^{M} 4\binom{M}{k}\left[\frac{\beta}{M}\right]^{k-2}\frac{1}{1-(\frac{1}{2})^{k-2}} \\ &= \sum_{k=3}^{M} 4\cdot\frac{M\cdot(M-1)\cdot(M-2)\cdots(M-k+1)}{k!}\frac{\beta^{k-2}}{M^{k-2}}\frac{1}{1-(\frac{1}{2})^{k-2}} \leq \frac{4}{3}M^2\frac{\beta}{1-\beta}. \end{aligned} \tag{15}$$

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

$$\begin{aligned} E[S] &= \textstyle\sum_1+\sum_3+\sum_4+\sum_5 \\ &\leq 4\cdot M\cdot 2^N - 2\cdot M(M-1)\cdot N + M^2\left[-\frac{4}{\beta}+\frac{4}{3}\frac{1}{\beta^2}+\frac{4}{3}\frac{\beta}{1-\beta}+2\log_2\frac{M}{\beta}\right]. \end{aligned} \tag{16}$$

Figure 3a shows the value of the upper bound given by (16) 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.

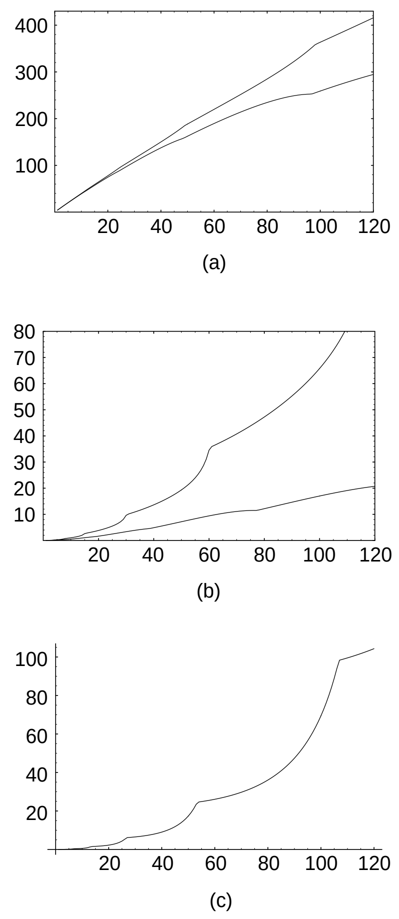


Figure 3: The upper and lower bounds on the number of nodes necessary to store an (a) MX quadtree and a (b) PM quadtree. (c) The upper bound on the number of nodes necessary to store a PMR_4 quadtree. The x and y axes correspond to the number of lines and nodes, respectively, for a tree of depth N=10. The number of nodes shown is divided by 10^3. Note the "rounded staircase-like" behavior resulting from our method of analysis which calculates several bounds (each for different integer values of the d_0 parameter) retaining the tightest one.

4.1.2 Interpretation

The value of $E[S]$ given by (16) is an upper bound on the number of nodes in a MX quadtree. Now the value of β (and the corresponding value of d_0) may be chosen to minimize it. Recall that our motivation for decomposing the calculation of $E[S]$ into two parts was based on the observation that when the number of lines M is large and the depth d is small (i.e., $d < d_0$), then the part of the tree at depth less than d_0 is almost complete, and thus all of its nodes contribute to $E[S]$.

Asymptotically, the dominant contribution to the number of nodes comes from the first term in (16) which may be transformed into a more familiar form using Theorem GPT2 from geometric probability. Letting L_i be the length of the i^{th} line in our geometric structure, the expected total length L of all lines is

$$E[L] = \sum_{i=1}^{M} E[L_i] = M \cdot \pi \frac{(2^N)^2}{4 \cdot 2^N} = \frac{\pi}{4} \cdot M \cdot 2^N. \tag{17}$$

Substituting (17) into the first term of (16) we get

$$E[S] \approx \frac{16}{\pi} E[L]. \tag{18}$$

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 [19, 20].

4.1.3 A Lower Bound

The derivation of $E[S]$ given by (10)–(16) may be used to set a lower bound on the expected number of nodes. $E[S]$ consists of the contributions 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,

$$E[S] \geq \Sigma_3 + \Sigma_4 - \tfrac{4}{3}M^2\tfrac{\beta}{1-\beta}. \tag{19}$$

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 them is a good approximation of $E[S]$ (see Figure 3a).

4.2 Region Quadtree

A region quadtree represents regions 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 smaller than N and if the region represented by the block is not homogeneous. Thus if the block is homogeneous, then it is not subdivided further and its corresponding node is a leaf. Otherwise, it is subdivided and its corresponding node has 4 sons (see Figure 1a).

Assuming the modified image model described in Section 3, the expected number of nodes in a region quadtree follows directly from the derivation in Section 4.1. Consider an MX quadtree corresponding to an image which is an instance of the basic random image model (and thus contains M lines). Consider also a region quadtree representing a second image which consists of regions separated by the lines of the first image. Both trees have the same structure with the only difference being the contents of each of the nodes. It follows that the statistical properties of a region quadtree corresponding to an instance of the modified random image model are exactly the same as the properties of an MX quadtree corresponding to an instance of the basic random image model. Hence, the expected number of nodes in a region quadtree is bounded by (16) and (19).

4.3 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 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 point in the block and all the lines that pass through the block meet at that point. 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 4 sons (see Figure 1c).

4.3.1 Analysis

Let α 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 may be written as

$$P'_d = 1 - p_{d,0} - p_{d,1} - \alpha \cdot p_{d,2} \tag{20}$$

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 α may be inferred from the Geometric Probability Theorem GPT3, 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,

$$\begin{aligned} P'_d = 1 \quad & - \quad \binom{M}{0}\left[1-\left(\frac{1}{2}\right)^d\right]^M \\ & - \quad \binom{M}{1}\left(\frac{1}{2}\right)^d\left[1-\left(\frac{1}{2}\right)^d\right]^{M-1} \\ & - \quad \frac{\pi}{8}\binom{M}{2}\left(\frac{1}{2}\right)^{2d}\left[1-\left(\frac{1}{2}\right)^d\right]^{M-2}. \end{aligned} \tag{21}$$

Inserting (21) in (4) we get an expression for the expected number of nodes in the PM quadtree.

To bound this expression, let us once again, as in the MX quadtree, formulate our analysis of $E[S]$ in terms of an additional parameter β (as well as M and N) which is defined as

$$\beta = M\left(\frac{1}{2}\right)^{d_0}. \tag{22}$$

As before, our analysis is in the form of a calculation of upper and lower bounds on $E[S]$ based on the fact that we have used an integer value of d_0 that ensures that β is less than 1.

The result of inserting (21) in (4) (i.e., $E[S]$) 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, what we do is 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 constraint that d_0 is an integer bounded by N and that $\beta < 1$.

Thus, we have $\sum_1$ and $\sum_2$

$$E[S] = 1 + \sum_{d=1}^{d_0} 4^d P'_{d-1} + \sum_{d=d_0+1}^{N} 4^d P'_{d-1} = \sum_1 + \sum_2. \tag{23}$$

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

$$\sum_1 = 1 + \sum_{d=1}^{d_0} P'_{d-1} 4^d \leq \sum_{d=0}^{d_0} 4^d \approx \frac{4}{3} 4^{d_0} = \frac{4}{3}\frac{M^2}{\beta^2}. \tag{24}$$

Σ_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

$$\Sigma_2 = \sum_{d=d_0+1}^{N} \sum_{k=2}^{M} C_k \left(\frac{1}{2}\right)^{kd-k} \cdot 4^d, \tag{25}$$

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 $k = 2$ (Σ_3) and $k > 2$ (Σ_4) cases yields:

$$\Sigma_2 = \Sigma_3 + \Sigma_4 \tag{26}$$

$$\Sigma_3 = \sum_{d=d_0+1}^{N} C_2 \left(\frac{1}{2}\right)^{2d-2} 4^d = 2(1 - \frac{\pi}{8})M(M-1)(N-d_0) \approx 1.215M(M-1)(N-d_0) \tag{27}$$

$$\begin{aligned} \Sigma_4 &= \sum_{k=3}^{M} C_k 2^k \sum_{d=d_0+1}^{N} \left(\frac{1}{2}\right)^{d(k-2)} \approx \sum_{k=3}^{M} C_k 2^k \frac{(\frac{1}{2})^{(k-2)(d_0+1)}}{1-(\frac{1}{2})^{k-2}} \\ &= 4 \sum_{k=3}^{M} \frac{C_k}{1-(\frac{1}{2})^{k-2}} \left[\frac{\beta}{M}\right]^{k-2}. \end{aligned} \tag{28}$$

Now, let us examine the coefficients C_k.

$$\begin{aligned} C_k &= (-1)^{k-1} \left[\binom{M}{k} - \binom{M}{1} \cdot \binom{M-1}{k-1} + \binom{M}{2} \cdot \binom{M-2}{k-2} \frac{\pi}{8} \right] \\ &= (-1)^{k-1} \left[\binom{M}{k} - M \cdot \frac{k}{M} \cdot \binom{M}{k} + \binom{M}{2} \frac{k(k-1)}{M(M-1)} \cdot \binom{M}{k} \frac{\pi}{8} \right] \\ &= (-1)^{k-1} \binom{M}{k} \left[1 - k + \frac{k(k-1)}{2} \frac{\pi}{8} \right]. \end{aligned} \tag{29}$$

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

$$-0.137M^k \leq C_k \leq 0.027M^k. \tag{30}$$

Inserting (30) into (29) and accounting for the worst cases leads to

$$-1.1M^2 \frac{\beta}{1-\beta} \leq \Sigma_4 \leq 0.22M^2 \frac{\beta}{1-\beta}. \tag{31}$$

Therefore, the contribution of Σ_4 to $E[S]$ is $O(M^2)$. Collecting the contributions of Σ_1, Σ_3, and Σ_4, we have

$$\begin{aligned} E[S] &= \Sigma_1 + \Sigma_3 + \Sigma_4 \\ &\leq 1.215M(M-1)N + M^2 \left[\frac{4}{3}\frac{1}{\beta^2} + 0.22\frac{\beta}{1-\beta} - 1.215\log_2 \frac{M}{\beta} \right]. \end{aligned} \tag{32}$$

Once again, d_0 is chosen to minimize (32), subject to the conditions that d_0 is an integer less than N and that β (as defined in (22)) is less than 1. Note that the first term in (32) is exact (it corresponds to Σ_3) while the second term (i.e., M^2 and its multiplier) is only a bound which is not very tight. Even this nontight bound (i.e., the second term in (32)) makes a negative contribution to the total number of nodes when the number of lines exceeds some modest threshold (e.g., for $M = 16, 32, \ldots$ and $\beta = 0.5$), in which case the simpler expression of $1.215M(M-1)N$ is also an upper bound. Figure 3b shows the value of the upper bound given by (32) 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.

4.3.2 Interpretation

The number of possible line pairs in the image is $\binom{M}{2}$. Multiplying it by α 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 (32), which is roughly proportional to $N \cdot M^2$, 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. It can be explained by the observation that the maximal depth of a PM quadtree depends, in part, on factors such as the minimum separation between two non-intersecting lines, the minimum distance between a vertex and a line, and the minimum distance between two vertices [42].

The linear dependence of the upper bound for the expected storage requirements (i.e. $E[S]$) on the maximum depth N is in agreement with the fact that in the worst case, all the vertices created by a random configuration of lines could appear in nodes at the maximum level. Furthermore, this linear dependence shows that the probability of the occurrence of such "bad" line sets is not zero. This behavior was confirmed by our simulations, which found that in most cases the randomly generated PM quadtree is small, but in some rare cases it can be very large. For images generated using the random image model, it is not surprising to find many of the vertices at the maximum level of the tree, as unlike real data (e.g., road networks), the factors that lead to the maximum depth (e.g., two vertices or non-intersecting lines being very close to each other, or a vertex and a line being very close) are more likely to arise. However, as M increases for a fixed value of N, the first term of the bound in (32) becomes very loose as the storage requirements are bounded by the number of nodes in a complete tree (i.e., $4/3 \cdot 2^{2N}$), thereby implying that for our analysis to be meaningful, M should be considerably smaller than 2^N.

4.3.3 A Lower Bound

The expected value of $E[S]$ consists of three terms $\sum_1$, $\sum_3$ and $\sum_4$. $\sum_3$ is an exact value, $\sum_1$ is positive (i.e. bounded from below by 0), and $\sum_4$ is bounded from below by $-1.1\frac{\beta}{1-\beta}M^2$. Hence, $\sum_3 - 1.1\frac{\beta}{1-\beta}M^2$ is a lower bound for $E[S]$. Note that attempting to improve the bound by choosing a small value of β would fail as this requires that d_0 have a higher value which means that $\sum_3$ has a lower value. For very large values of N, $\sum_1$ and $\sum_4$ are negligible in comparison with $\sum_3$ in which case the difference between the upper bound and the lower bound is small implying that each of these two bounds is itself a good approximation of $E[S]$.

4.4 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 4 sons (see Figure 1d).

Consider a Bucket PMR$_2$ quadtree. The probability that the splitting conditions are satisfied may be written as

$$P'_d = 1 - p_{d,0} - p_{d,1} - (1-\alpha) \cdot p_{d,2}. \tag{33}$$

This splitting probability is clearly identical to that of the PM quadtree, apart from changing a multiplicative constant from α to $1-\alpha$. Notice that we don't split if the two lines do not intersect. However, if they do intersect, then the result is that there are 4 line segments in the block and thus it is split. Therefore, the expected number of nodes is given by an expression similar to (32) and increases linearly with the maximal depth N.

For the case that $q = 3$ the splitting probability here is the same as in (33) with the subtraction of a term corresponding to the small probability that 3 line segments intersect the region, but not each other. Thus, the splitting probabilities for $q = 2$ and for $q = 3$ are expected to be similar.

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

$$P'_d < 1 - p_{d,0} - p_{d,1} - p_{d,2}. \tag{34}$$

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 3 or 4 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. From the expression (4), it is clear that if the probabilities P_i are lower, then the expected number of nodes is also lower. Therefore, for deriving the upper bound, we may substitute the upper bound (34) for the probability P'_d. Using the same techniques as in Section 4.3, we define β and d_0 as in (22) and decompose the sum corresponding to $E[S]$ into two sums $\sum_1$ (24) and $\sum_2$ (25). Here, however,

$$\begin{aligned} C_k &= (-1)^{k-1}\left[\binom{M}{k} - \binom{M}{1}\cdot\binom{M-1}{k-1} + \binom{M}{2}\cdot\binom{M-2}{k-2}\right] \\ &= (-1)^{k-1}\left[\binom{M}{k} - \binom{M}{1}\cdot\frac{k}{M}\cdot\binom{M}{k} + \binom{M}{2}\cdot\frac{k(k-1)}{M(M-1)}\cdot\binom{M}{k}\right] \\ &= (-1)^{k-1}\binom{M}{k}\left[\frac{(k-1)(k-2)}{2}\right], \end{aligned} \tag{35}$$

implying that $C_2 = 0$ and that

$$-\frac{1}{8}M^k \le C_k \le \frac{1}{6}M^k. \tag{36}$$

The bounds (36) 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,

$$\Sigma_4 = \sum_{k=3}^{M} C_k 2^k \sum_{d=d_0+1}^{N}\left(\frac{1}{2}\right)^{d(k-2)} \approx \sum_{k=3}^{M} C_k 2^k \frac{(\frac{1}{2})^{(k-2)(d_0+1)}}{1-(\frac{1}{2})^{k-2}} \tag{37}$$

$$= 4\sum_{k=3}^{M}\frac{C_k}{1-(\frac{1}{2})^{k-2}}\left[\frac{\beta}{M}\right]^{k-2} \le \frac{4}{3}M^2\frac{\beta}{1-\beta}. \tag{38}$$

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

$$E[S] \le \frac{4}{3}M^2\left[\frac{1}{\beta^2} + \frac{\beta}{1-\beta}\right]. \tag{39}$$

Figure 3c shows the value of the upper bound given by (39) as a function of M at a maximal depth of $N = 10$. 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 β, 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

$$E[S] \le 6.37M^2 \qquad (q = 4). \tag{40}$$

The bound (40) 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 Section 4.3.2), 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 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 non-intersection of two lines. It is clear, however, that the splitting probability decreases as q increases, and therefore for $q \geq 4$, the bound (39) 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 model, all vertices have 4 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.

4.5 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 a block b is split once, and only once, if b is both 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 is entered into 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. Since the maximal depth of a PMR$_q$ quadtree is not known in advance, a rule that repeatedly splits each block until no more than q line segments intersect with it may lead to a PMR$_q$ quadtree of infinite depth. This situation arises when more than q line segments intersect at a given point. It is interesting to observe that for our model of a random line image, the previous situation cannot arise for $q \geq 4$ as the probability that three or more of the random lines intersect at a point is zero. Regardless of the configuration of the random lines (i.e., whether or not the intersection of three or more random lines is permitted), the creation of an infinitely deep Bucket PMR$_q$ quadtree is precluded from occurring by our definition of the Bucket PMR$_q$ quadtree to have a limit on its depth (i.e., N).

For the case that $q \geq 4$, we may use the upper bound (39) 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. To see this, we observe that since we cannot have the situation that 3 or more random lines meet at a point, when $q \geq 4$ we cannot have a PMR$_q$ quadtree of infinite depth. This means that given a set of M random lines, the PMR$_q$ quadtree for them is of a finite depth with a limit D that can be calculated based on the minimum horizontal and vertical separations between the intersection points of the lines. Therefore, we can construct a Bucket PMR$_q$ quadtree v with a parameter $N = D$ so that regardless of the order in which the M lines are inserted into the PMR$_q$ quadtree t, the nodes in the PMR$_q$ quadtree t will always be a subset of the nodes in the Bucket PMR$_q$ quadtree v for the M lines. Recall that the bound on the number of nodes in a Bucket PMR$_q$ quadtree that we obtained was independent of the depth N of the Bucket PMR$_q$ quadtree as we let N go to inf when we computed $\sum_4$ in (38). Thus the bound that we obtained in (39) is also good for Bucket PMR$_q$ quadtrees of depth D. Therefore, it is also applicable to PMR$_q$ quadtrees subject to $q \geq 4$.

For $q = 2$, we recall that the number of nodes in the Bucket PMR$_2$ quadtree is similar to this number in the PM quadtree which is given by (32). Unfortunately, this does not provide a bound on the storage required by a PMR$_2$ quadtree because the upper bound on the number of nodes in the PM quadtree is linear in the maximum depth while the depth of the PMR$_2$ quadtree may be as high as the number of random lines M, thus giving an unrealistically high upper bound. Such a situation can occur when two random lines intersect at point p as they result in four line segments. In the worst case, the block b that currently contains p would be split each time a new random line i is inserted provided i passes through b. In contrast, the decomposition for the Bucket PMR$_2$ quadtree ceases once the block containing p is at depth N. Thus, the node set associated with the Bucket PMR$_2$ quadtree is guaranteed to be a superset of the node set associated with the PMR$_2$ quadtree (as well as for all values of q) only if its depth is trivially M or higher. Unlike the case when $q \geq 4$, when $q = 2$, the upper bound on the number of nodes in the Bucket PMR$_2$ quadtree does not converge when the depth N increases, and thus the bound obtained is very loose. Therefore, obtaining an upper bound on the number of nodes in a PMR$_2$ quadtree is an issue left for future research, although, as mentioned before, the case of $q \geq 4$ is more interesting from a practical standpoint as we do not want the common situation of a road junction (i.e., when 4 line segments meet at a point) to cause an arbitrarily large amount of splitting.

4.6 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.

$$\begin{aligned}
\textit{Region quadtree } E[S] &= O(M \cdot 2^N) \qquad (41)\\
\textit{MX quadtree } E[S] &= O(M \cdot 2^N)\\
\textit{PM quadtree } E[S] &= O(M^2 \cdot N)\\
\textit{PMR}_q \textit{ quadtree } E[S] &= O(M^2) \quad (q \geq 4)\\
\textit{Bucket PMR}_q \textit{ quadtree } E[S] &= O(M^2 \cdot N) \quad (q = 2)\\
\textit{Bucket PMR}_q \textit{ quadtree } E[S] &= O(M^2) \quad (q \geq 4)
\end{aligned}$$

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 the Bucket PMR$_2$ quadtree. For $q \geq 4$, for both the PMR$_q$ quadtree and the Bucket 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 (and region) quadtree, the number of nodes is proportional to the total length of the line segments (or the region boundaries). This conclusion confirms a similar result obtained by Hunter and Steiglitz [19, 20]. For both the PM quadtree and the Bucket 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. The lower bounds on the size of the MX quadtree and the PM quadtree include the dominant components of order $O(M \cdot 2^N)$ and $O(M^2 \cdot N)$, respectively, thereby demonstrating that the upper bounds reflect the real growth rate of the structures. For both the PMR$_q$ quadtree and the Bucket 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 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 (16), (32) and (39), which depend on a parameter β. It is worth re-emphasizing that the values d_0 and β 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 β which minimizes them while satisfying relation (8) (with d_0 being an integer bounded by N). Fixing the value of β 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 β is not justified because it usually implies a non-integer 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.

These bounds hold for all values of M and N. However, they become trivial when $M \geq 2^N$. In this case, the parameter d_0 approaches the value of the maximal depth N, and the value of the bounds approach the number of nodes in the complete quadtree of depth N which is a trivial bound on any quadtree of depth N as there cannot be any more nodes. For the datasets that we considered (with 25, 50, 75, and 100 random lines), this was the case for $N = 6$ and thus bounds for such a depth are omitted from the results described in Section 4.7 and the appropriate tables.

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.

4.7 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 PMR$_4$ quadtrees of 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. They usually agree with the analytical predictions, and, in particular, the upper bounds for all of the quadtree variants always hold. The exceptions occur when the number of lines is too low relative to the size of the complete quadtree (i.e., at the higher depths as we see for the PM quadtrees of depth 14). In such a case, we observe that we can no longer assume

M	N	MX				PM				Bucket PMR_4		
		MODEL	UB	LB	ER	MODEL	UB	LB	ER	MODEL	UB	ER
25	10	94.9K	97.2K	90.7K	94.9K	3.88K	5.5K	2.48K	2.55K	0.846K	4.33K	0.669K
50	10	179K	188K	162K	181K	12.8K	19.3K	7.17K	9.25K	3.38K	17.3K	2.77K
75	10	255K	268K	225K	256K	25.1K	43.8K	11.5K	19.1K	7.50K	32.4K	6.01K
100	10	325K	364K	259K	325K	39.9K	65.8K	17K	31.3K	13.1K	69.3K	10.4K
25	14	1.63M	1.63M	1.62M	1.62M	6.78K	8.42K	(5.39K)	2.92K	0.863K	4.33K	0.676K
50	14	3.23M	3.24M	3.21M	3.24M	24.5K	31.2K	(19.1K)	11.3K	3.52K	17.3K	2.83K
75	14	4.82M	4.83M	4.79M	4.80M	51.6K	70.8K	(38.4K)	25.0K	7.97K	32.4K	6.30K
100	14	6.40M	6.43M	6.32M	6.37M	87.0K	114K	(65.1K)	44.0K	14.2K	69.3K	11.2K

Table 1: Comparison of the result of expression (4) (MODEL), 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 quadtree. Parenthesized entries indicate results that are inconsistent with the predictions. These inconsistencies only occur for some of the lower bounds whose values are in excess of the actual number of nodes.

that most pairs of regions (i.e., nodes corresponding to blocks) are intersected by different lines, which counters the independence assumption. In this case, the lower bounds may not be valid (i.e., they are too high in the sense that they exceed the observed values). This is especially true for nodes at the higher depths.

To verify this observation we also evaluated the expression (4) for several values of the number of lines M and the maximum depth N of the quadtrees. Recall that the sum (4) depends only on the independence assumption. We found that while this sum closely approximates the actual expected number of nodes for the MX quadtree and by about 25% for the Bucket PMR_4 quadtree, this is not the case for the PM quadtree for which the independence assumption causes the value of (4) to be much higher than the actual value, provided that the depth is high enough. These observations are examples of the limitations on the validity of the independence assumption.

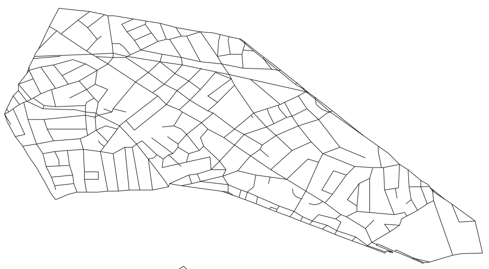
The upper bounds obtained for the MX quadtree were consistently very close to the observed node counts. In contrast, the upper bounds for the PM and Bucket PMR_4 quadtrees consistently exceed the observed node counts by factors as high as 4 and 5, respectively. This difference can be attributed to both the possible inappropriateness of the independence assumption and 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.

5 Predicting Storage Requirements for Real Data

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

We conducted our tests using real data corresponding to road maps in the US that are part of the TIGER files used by the Bureau of the Census (see Figure 4). We used maps ranging from a small map having only 585 road segments (Falls Church county) to the largest map including 39,719 segments (Montgomery county). The actual data for the maps such as the depth (N), number of vertices, segments, non-shape vertices (*NSV* and described below), the normalized length (*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.

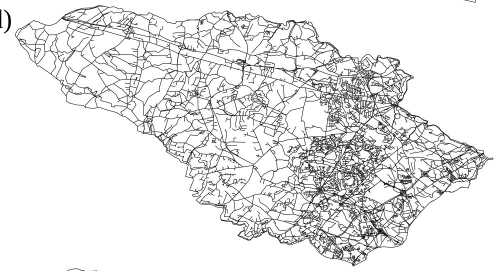
Our approach to applying the expected node count predictions to a real map r depends on finding, for each map r, a class c of a random line images which shares some property with r. The expected number of nodes required to represent a)



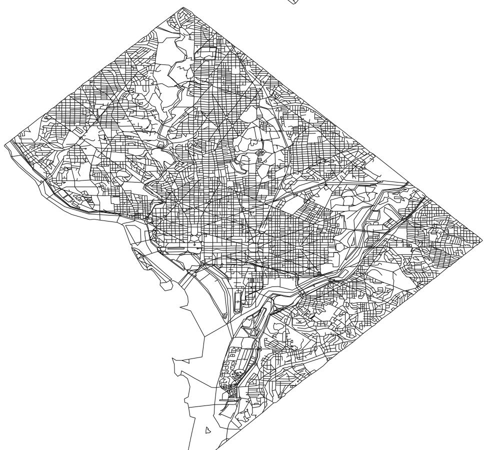
b)

c)

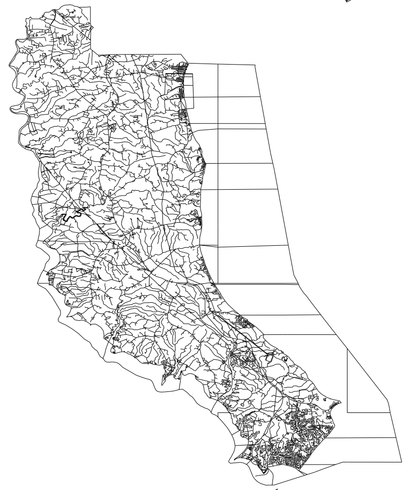
d)



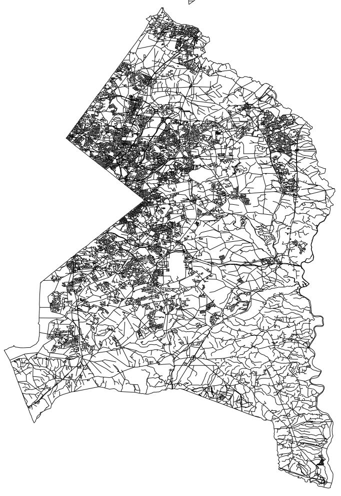
e)



f)



g)



h)

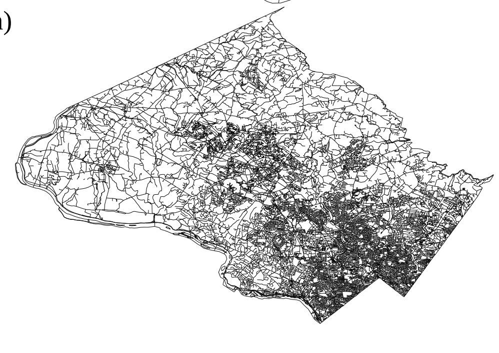


Figure 4: Eight maps used to test the estimates on the number of nodes in the representing quadtrees.

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 Georges	10	50161	18055	117.55	59551	315289	157977	88485
Prince Georges	12	50161	18055	117.55	59551	1937553	277209	104993
Prince Georges	14	50161	18055	117.55	59551	8993017	318889	107889
Prince Georges	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

Table 2: Description of the TIGER files maps (see Figure 4), used in the experiments as well as the corresponding actual storage requirements for the MX, PM, and Bucket PMR_4 quadtrees.

a random image from class c is taken to be the estimate for the number of nodes required to represent the map r.

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 (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 infered from the image r and specfied the fraction retained. Thus, we are assuming that the number of nodes is directly proportional to the ratio of the nonempty area, and 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 random images class 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 line model, the expected line length $E[L]$ is $\pi/4 \cdot M \cdot 2^N$. This is the result of (17) which follows directly from Theorem GPT2. 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 shown in Figure 4a) that we used is $16.49 \cdot 2^N$ 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.

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* [6]) 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 *non-shape-vertex (NSV) estimator*. Clearly, there are many cases for which this heuristic is inapplicable. 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 as *S-based estimator*. Like the L-based and V-based estimators, it is based on replacing the given map by 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 four. 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 $\sqrt{\frac{8S_{map}}{\pi}}$.

The fourth estimator that we consider is termed a *d-based estimator*. This estimator is based on replacing the given map by a (usually smaller) random line image having the same number of vertices and density (i.e., average line segment length). 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 by the expected number of vertices on the line. We have already seen 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 $\pi M/8$. Therefore, the expected segment length is $2 \cdot 2^N/M$. 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}, yields 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 line 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 non-shape vertices 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 non-uniform 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 one. 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 it in the basic sum (4) 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 obtained by inserting the rounded estimate $\hat{M}$ directly into the upper bounds (16), (32), and (39), although we do not tabulate them here.

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 Georges	10	1.42	0.37	0.32	2.32	1.08	1.14	0.35	0.42	0.63
Prince Georges	12	1.14	0.34	0.30	2.12	1.11	1.18	0.48	0.68	1.01
Prince Georges	14	1.06	0.40	0.30	2.07	1.41	1.20	0.56	1.03	1.16
Prince Georges	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

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 requirement using the estimator and the actual ones given in Table 2.

The particular estimators that we have 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 two 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 the best for the MX quadtrees, while the V-based estimator seems to work the best for the PM and PMR_4 quadtrees. We constructed the d-based (as well as the related S-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 and Steiglitz [19, 20] which 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 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 (42) on the expected number of nodes as a function of the number of line segments in real images and the level of permitted subdivision. The experiments were the same as those described above. In all of the experiments we assume that the number of line segments in the map is proportional to M^2, where M is a parameter related to the effective number of lines in the equivalent random images (recall that there are $O(M^2)$ intersection points for the M infinite lines).

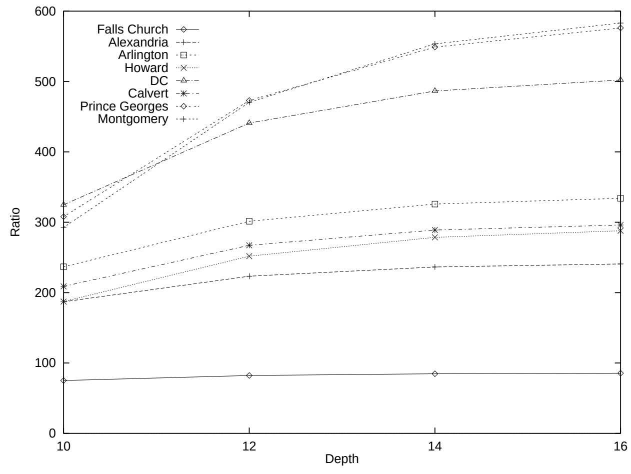


Figure 5: Ratio of MX quadtree nodes to side length.

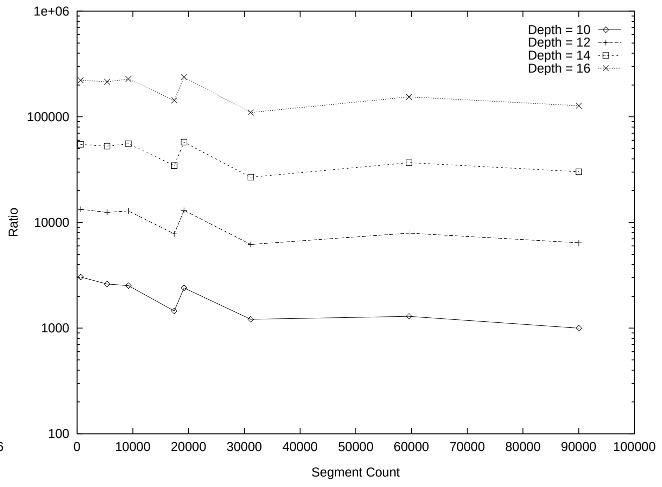


Figure 6: Ratio of MX quadtree nodes to square root of segment count.

We first examine the MX quadtree. Figure 5 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 6 shows the ratios of the node count to the square root of the number of line segments (i.e., M) as a function of the number of line segments (i.e., M^2) for the different depths which, as expected, are close to being constant (i.e., horizontal lines) when the number of line segments relative to the depth is large enough. We used a logarithmic scale in Figure 6 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 7 and 8 show the ratio of the node count to the number of non-shape vertices (i.e., NSV) 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 was also observed to be the case with the

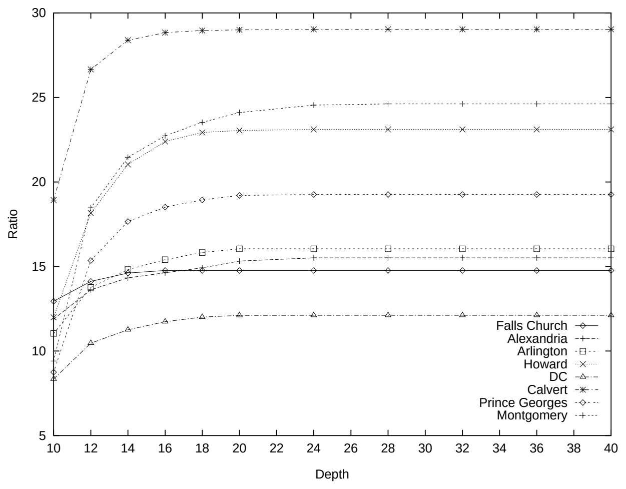


Figure 7: Ratio of PM quadtree nodes to NSV.

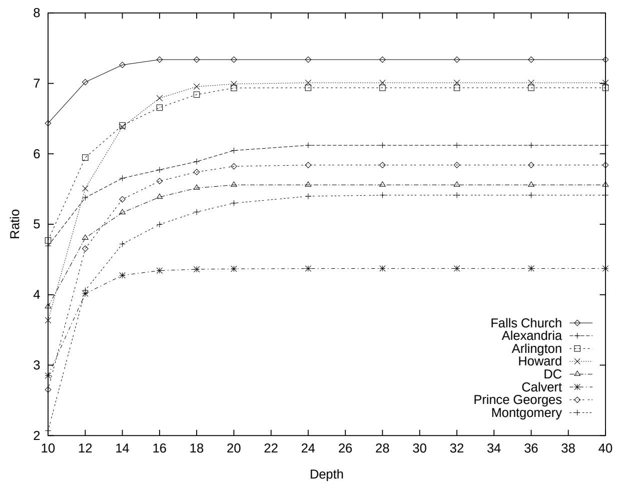


Figure 8: Ratio of PM quadtree nodes to the segment count.

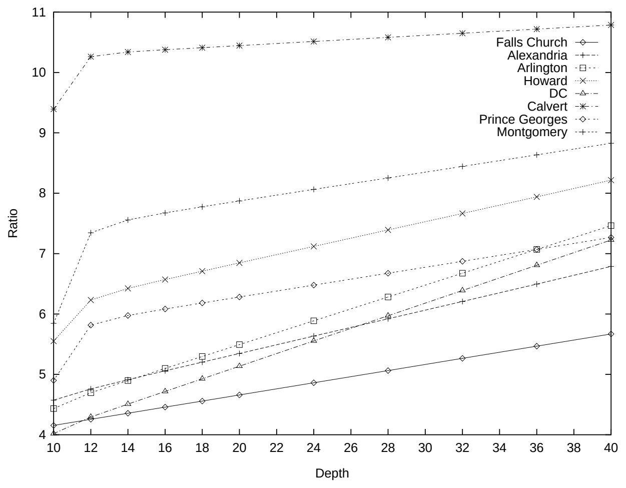


Figure 9: Ratio of PMR quadtree nodes to NSV, q=4.

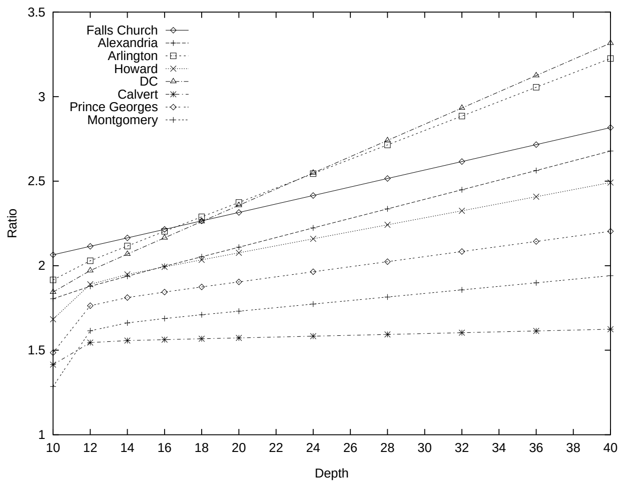


Figure 10: Ratio of PMR quadtree nodes to segment count, q=4.

randomly generated data. As we recall, this is probably because of the invalidity of the independence assumption.

Finally, we examine the PMR_4 quadtree. Figures 9 and 10 show the ratio of the node count to the number of non-shape vertices (i.e., *NSV*) and to the number of line segments (i.e., M^2), respectively, as a function of the depth for the different maps which we would 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 no longer be applied. It is interesting to observe that the lines in Figures 9 and 10 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 where 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

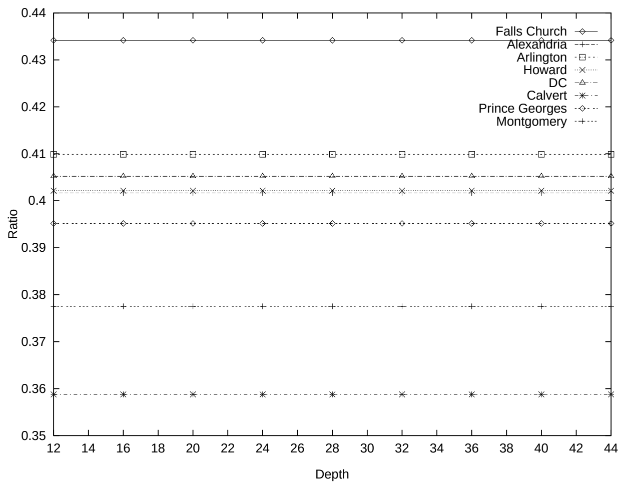


Figure 11: Ratio of PMR quadtree nodes to segment count, q=12.

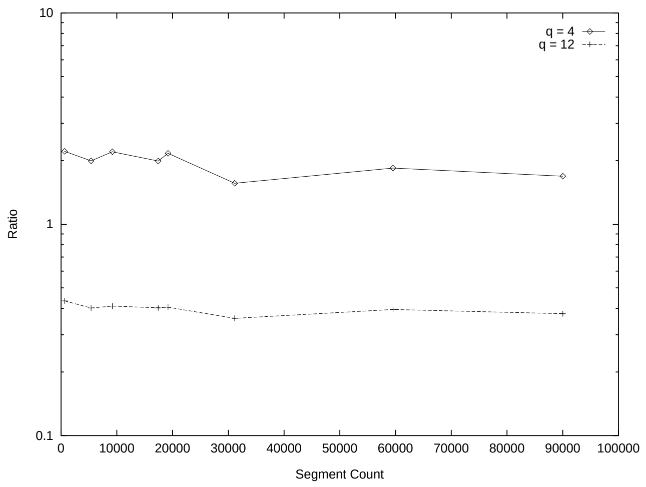


Figure 12: Ratio of PMR quadtree nodes to segment count vs. segment count at a depth of 16.

in Figure 11 for $q = 12$. Figure 12 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 $q = 12$. Notice that the ratios are all within 6% of their average value. Figure 12 also reveals a general trend that 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.

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 random image models. 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 include 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 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) 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 confirming earlier observations on the possible worst case behavior of the PM quadtree [42].

Perhaps our most important result is showing that the space requirements of the Bucket PMR$_q$ and PMR$_q$ ($q \geq 4$) quadtree are asymptotically proportional to the number of line segments. This was shown theoretically for a random line 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 importantly, to show that it is independent of the maximum depth of the tree. It is thus not surprising that the PMR$_q$ quadtree has found use in experimental systems (e.g., QUILT [45]) as well as commercial systems (e.g., United Parcel Service (UPS) [4]).

7 Acknowledgements

We appreciate the comments of S. K. Bhaskar and Gary D. Knott who made suggestions about the evaluation of the sums.

References

[1] C. H. Ang, Applications and analysis of hierarchical data structures, Computer Science TR-2255, University of Maryland, College Park, MD, June 1989.

[2] W. G. Aref and H. Samet, Optimization strategies for spatial query processing, *Proceedings of the Seventeenth International Conference on Very Large Data Bases*, G. Lohman, ed., Barcelona, September 1991, 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, *Proceedings of the SIGMOD Conference*, Atlantic City, NJ, June 1990, 322–331,

[4] R. Bonefas, personal communication, 1991.

[5] T. Brinkhoff, H. P. Kriegel, R. Schneider, and B. Seeger. Multi-step processing of spatial joins. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 197–208, Minneapolis, MN, June 1994.

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

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

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

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

[10] P. Flajolet, G. Gonnet, C. Puech, and J. M. Robson, Analytic variations on quadtrees, *Algorithmica 10*, 6(December 1993), 473–500.

[11] P. Flajolet and T. Lafforge, Search costs in quadtrees and singularity perturbation asymptotics, *Discrete & Computational Geometry 12*, 2(September 1994), 151–175.

[12] P. Flajolet and C. Puech, Partial match retrieval of multidimensional data, *Journal of the ACM 33*, 2(April 1986), 371–407.

[13] E. Fredkin, Trie memory, *Communications of the ACM 3*, 9(September 1960), 490–499.

[14] G. Graefe, Query evaluation techniques for large databases, *ACM Computing Surveys 25*, 2(June 1993), 73–170.

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

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

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

[18] M. Hoshi and P. Flajolet, Page usage in a quadtree index, *BIT 32*, 3(1992), 384–402.

[19] 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.

[20] G. M. Hunter and K. Steiglitz, Operations on images using quad trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence 1*, 2(April 1979), 145–153.

[21] C. L. Jackins and S. L. Tanimoto, Oct-trees and their use in representing three dimensional objects, *Computer Graphics and Image Processing 14*, 3(November 1980), 249–270.

[22] G. Kedem, The Quad-CIF tree: a data structure for hierarchical in-line algorithms, *Proceedings of the 19th Design Automation Conference*, Las Vegas, June 1982, 352–257.

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

[24] K. Knowlton, Progressive transmission of grey-scale and binary pictures by simple, efficient, and lossless encoding schemes, *Proceedings of the IEEE 68*, 7(July 1980), 885–896.

[25] D. E. Knuth, *The Art of Computer Programming*, vol. 3, *Sorting and Searching*, Second Edition, Addison-Wesley, Reading, MA, 1973.

[26] J. H. Lee, D. H. Kim, and C. W. Chung. Multi-dimensional selectivity estimation using compressed histogram information. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 205–214, Philadelphia, June 1999.

[27] C. Mathieu, C. Puech, and H. Yahia, Average efficiency of data structures for binary image processing, *Information Processing Letters 26*, 2(October 1987), 89–93.

[28] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 448–459, Seattle, June 1998.

[29] D. Meagher, Geometric modeling using octree encoding, *Computer Graphics and Image Processing 19*, 2(June 1982), 129–147.

[30] M. Muralikrishna and D. J. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, pages 28–36, Chicago, June 1988.

[31] R. C. Nelson and H. Samet, A consistent hierarchical representation for vector data, *Computer Graphics 20*, 4(August 1986), 197–206 (also *Proceedings of the SIGGRAPH '86 Conference*, Dallas, August 1986).

[32] R. C. Nelson and H. Samet, A population analysis of quadtrees with variable node size, Computer Science TR-1740, University of Maryland, College Park, MD, December 1986.

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

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

[35] M. Ouksel and P. Scheuermann, Storage mappings for multidimensional linear dynamic hashing, *Proceedings of the Second ACM SIGACT–SIGMOD Symposium on Principles of Database Systems*, Atlanta, March, 1983, 90–105.

[36] 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 Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 214–221, Washington, DC, May 1993.

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

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

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

[40] 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, Computer Science TR-1457, University of Maryland, College Park, MD, November 1984.

[41] H. Samet and M. Tamminen, Efficient component labeling of images of arbitrary dimension represented by linear bintrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence 10*, 4(July 1988), 579–586.

[42] H. Samet and R. E. Webber, Storing a collection of polygons using quadtrees, *ACM Transactions on Graphics 4*, 3(July 1985), 182–222.

[43] L. A. Santalo, Integral geometry and geometric probability, in *Encyclopedia of Mathematics and its Applications*, G. C. Rota, ed., Addison-Wesley, Reading, MA, 1976.

[44] C.A. Shaffer, R. Juvvadi, and L.S. Heath, Generalized comparison of quadtree and bintree storage requirements, *Image and Vision Computing 11*, 7(September 1993), 402–412.

[45] C.A. Shaffer, H. Samet, and R. C. Nelson, QUILT: a geographic information system based on quadtrees, *International Journal of Geographical Information Systems 4*, 2(April-June 1990), 103–131 (also University of Maryland Computer Science TR –1885.1).,

[46] M. Tamminen, Encoding pixel trees, *Computer Vision, Graphics, and Image Processing 28*, 1(October 1984), 44–57.

[47] M. Tamminen, Comment on quad- and octtrees, *Communications of the ACM 27*, 3(March 1984), 248–249.