Are Window Queries Representative For Arbitrary Range Queries?

Bernd-Uwe Pagel
Department of Computer Science
University of Hagen
bernd-uwe.pagel@fernuni-hagen.de

Hans-Werner Six
Department of Computer Science
University of Hagen
hw.six@fernuni-hagen.de

Abstract

In this paper, we characterize the efficiency of spatial access structures in terms of the expected number of data bucket accesses needed to perform a range query. We derive performance formulas for various kinds of range queries like intersection, containment, and enclosure queries with different geometric shapes. The formulas exhibit that range query performance in general depends on three parameters: the area sum of the data regions, the perimeter sum, and the number of regions. Only the weights of these parameters vary from shape to shape and query type to query type. To a wide extent, our results prove the conjecture that window queries are representative for range queries in general.

1 Introduction

During the last 15 years various spatial access structures have been designed and compared against each other with respect to query performance ([BKSS90, See91] represent a recent selection). Window queries are considered most important, mainly because they can easily be specified even using alpha-numeric input devices and perfectly comply with a presentation on screen. Correspondingly, almost all performance investigations of spatial access structures concentrate on window queries and less attention is paid to more sophisticated query shapes like lines, ellipses, arbitrary rectangles, or polygons (a rare exception is [BF95]). If mentioned at all, these queries are dismissed with the intuitive argument that the window query performance of an access structure serves as an indicator for the quality of the spatial clustering and is therefore representative for range query performance in general. In practice, however, these queries frequently occur and their importance can also be seen from the fact that they are part of the Sequoia 2000 benchmark [SFGM93].

In this paper, we analyse intersection queries, containment queries and enclosure queries with different geometric shapes (e.g. lines, circles, windows and arbitrary rectangles). We focus on the largest group of spatial access structures, namely structures with rectangular, axis-parallel data regions, called box regions. Among them are structures using minimum-bounding-rectangle (MBR) approximations (e.g. the LSD-tree [HSW89] and R-tree variants [Gut84, SRF87, BKSS90]), all grid structures (e.g. [NHS84]) and some further popular structures like the Buddy-tree [SK90].

Applying the approach introduced in [PSTW93, PSW95] we end up in explicit, easy to compute formulas proving that range query performance in general depends on three parameters: the area sum of the data regions, the perimeter sum, and the number of regions. Only the weights of these parameters vary from shape to shape and query type to query type. To a wide extent, our results prove the conjecture that window queries are representative for range queries in general.

2 Framework

As prerequisite of the analysis, we introduce a general framework. After some basic definitions we motivate the performance measure we are using and present some observations which will facilitate the analysis.

2.1 Basic Definitions

Let d be the dimension of the data space, $S_i = [0,1)$, $1 \leq i \leq d$, and $S = S_1 \times S_2 \times \ldots \times S_d$ be the d-dimensional data space in which all geometric objects are defined. Since usually the bounding boxes of real objects are used as geometric keys, we assume that each geometric object g is a d-dimensional interval $g = [g.l_1, g.r_1] \times \ldots \times [g.l_d, g.r_d],\ g.l_i, g.r_i \in S_i,\ g.l_i \leq g.r_i$.

Let us assume that for storing a set $G = \{g_1, \ldots, g_N\}$ of N geometric objects the spatial access structure $AS(G)$ currently consumes m data buckets $b_1, b_2, \ldots, b_m$ with a capacity of c_{dat} objects each. With each object $g \in G$, at least one bucket is associated. The data region $b_i.reg \subseteq S$ is the minimal d-dimensional interval enclosing all objects in b_i. We call $B = \{b_1, \ldots, b_m\}$ a bucket set. The corresponding data regions $DSO(B) = \{b_1.reg, \ldots, b_m.reg\}$ form the organization of the data space.

Assuming a query range q we are interested in three *range query types*:

- An intersection query asks for all objects intersected by q:

$$IQ(q, AS(G)) = \{g \in G \mid q \cap g \neq \emptyset\} \quad (1)$$

- A containment query asks for all objects completely contained in q:

$$CQ(q, AS(G)) = \{g \in G \mid q \cap g = g\} \quad (2)$$

- An enclosure query asks for all objects enclosing q:

$$EQ(q, AS(G)) = \{g \in G \mid q \cap g = q\} \quad (3)$$

For sake of brevity we usually do not distinguish between a query and its range, just talking about q.

Without loss of generality and only for simplicity reasons, we choose $d = 2$ for further considerations. This, for instance, reduces boxes and data regions to 2-dimensional rectangles. By the extensions *.area*, *.perimeter*, and *.centroid* we refer to the area, perimeter, and centroid of a 2-dimensional solid s, respectively.

Finally, we assume an spatial access structure guaranteeing that (1) the access of a data buckets only depends on its data region and (2) unnecessary bucket accesses are avoided. These assumptions are fulfilled by any structure where the directory maintains the data regions, but does not provide other useful information about data buckets (for instance, about the area of the smallest, resp. largest, object of every bucket). Actually, almost any popular structure fulfils these properties.

2.2 Performance Measure

Investigation and fair comparison of different approaches of spatial data management requires a comparative performance measure, which should be independent of access structure and implementation details, and even independent of whether the objects are points or non-point objects. We claim that the expected number of bucket accesses needed to perform a query is such a cost measure, because in practical applications one is primarily interested in the average query cost which is dominated by data bucket accesses. (Data bucket accesses in particular exceed by far external accesses to the paged parts of the corresponding directory concerning frequency and execution time.)

Obviously, the cost measure depends on the actual data bucket set B and the query behavior of the user. Since we deal with expected values, we first have to define the underlying probability model reflecting the expected query behavior of the user. Let us assume $\mathcal{QM}$ to be such a model. Moreover, for a bucket set $B = \{b_1, \cdots, b_m\}$ and a random query Q (a random variable) in model $\mathcal{QM}$ let $Prob(Q \text{ MEETS } B; j)$ be the probability that performing query Q forces exactly j buckets in B to be inspected. Then the expected number of bucket accesses needed to perform query Q – the *performance measure* $\mathcal{PM}$ *for* $\mathcal{QM}$ – is given by

$$\mathcal{PM}(\mathcal{QM}, B) = \sum_{j=0}^{m} j \cdot Prob(Q \text{ MEETS } B; j). \quad (4)$$

In this paper, random queries are denoted by capitals and actual queries which are, for instance, results of random events, by (the corresp.) small letters.

2.3 Range Query Models

Performance measure $\mathcal{PM}$ is based on a probability model for range queries which takes into account query types as well as query shapes. The main problem is to cope with the various geometric shapes, to capture and formally describe them. We consider points, lines, rectangles, ellipses, and polygons. Except for polygons every shape can be characterized by a simple geometric pattern (e.g. a line, ellipse or rectangle), the angle of rotation, size, location and, if necessary, an aspect ratio. In a probability model these five parameters have to be specified appropriately.

Every range query model proposed in this paper fulfils the following assumptions:

- The geometric *pattern* of the queries (captured by the model) is homogeneous (e.g. only windows are considered).
- The *aspect ratio* is a constant (e.g. only squares with aspect ratio 1:1 are considered).
- The query *size* is a constant and denotes either a length (line query) or an area (2-dimensional query).
- The query *location* is given by its centroid. The location is uniformly distributed over the data space, i.e. every part of the data space is equally likely to be requested.
- The smallest angle needed to rotate a query (clockwise) into a congruent x-axis-parallel query is called *orientation* of the query. Orientations are uniformly distributed over the interval $I_\alpha = [\alpha_{min}, \alpha_{max}]$, $0 \leq \alpha_{min} \leq \alpha_{max} \leq 2\pi$. Note that $\alpha_{min} = \alpha_{max}$ is explicitly allowed and indicates a constant orientation.
- Pattern, aspect ratio, size, location, and orientation are assumed *stochastically independent*.

Let PATTERN $\in \{ ⟋, ▭, ⬭ \}$, ASPECT RATIO and SIZE be the constants introduced above and $U[M]$ describe the uniform distribution over set M. Then each query model with these properties can be derived from the general range query model

$$\mathcal{RQM} = (\text{PATTERN}, \text{ASPECT RATIO}, \text{SIZE}, U[S], U[I_\alpha])$$

by "instantiation". In the sequel, we refer to six such "instantiations":

$\mathcal{RQM}_{\bullet} = (▭, \text{-}, 0, U[S], U[0,0])$ models point queries.

$\mathcal{RQM}_{\measuredangle} = (⟋, \text{-}, c_{len}, U[S], U[0,\pi])$ models line queries.

$\mathcal{RQM}_{\bigcirc} = (⬭, 1{:}1, c_{area}, U[S], U[0,0])$ models circle queries.

$\mathcal{RQM}_{\square} = (▭, 1{:}1, c_{area}, U[S], U[0,0])$ assumes square windows.

$\mathcal{RQM}_{⧄} = (▭, 1{:}1, c_{area}, U[S], U[0,\frac{\pi}{2}])$ assumes arbitrary squares (which are not necessarily axis-parallel).

$\mathcal{RQM}_{\diamond} = (▭, 1{:}1, c_{area}, U[S], U[\frac{\pi}{4},\frac{\pi}{4}])$ assumes diamonds, i.e. 45°-oriented squares.

Note that every model uniquely characterizes the query ranges, but does not specify the query type. It is left open whether intersection, containment or enclosure queries are modeled, such that an "interpretation" is needed to apply a model. Since the next section provides results which are independent of "interpretations", we return to query types later.

Thinking about the behavior of real users, the models seem somewhat simple and unrealistic. Uniformly distributed locations and orientations, good-natured aspect ratios and a constant query size at best reflect a user behavior which can sometimes be observed with novice and occasional users. However, dealing with uniform distributions is common practice and meaningful, because problems become feasible and specific conclusions are often confirmed under less simple assumptions. For example, the analysis of different (window) query behavior and their corresp. models exhibits such a tendency [PSTW93, BF95].

2.4 Query-Sensitive Environments

We now turn to the question, how to efficiently compute $\mathcal{PM}$ for a concrete access structure $AS(G)$ with data bucket set B and an "instantiated" range query model $\mathcal{RQM}$. The following Lemma facilitates the computation of $\mathcal{PM}$ for arbitrary query models.

Lemma 1. [PSTW93] Let $Prob(Q \text{ MEETS } b_i)$ be the probability that performing random query Q forces the access of bucket b_i. Then holds

$$\sum_{j=0}^{m} j \cdot Prob(Q \text{ MEETS } B; j) = \sum_{i=1}^{m} Prob(Q \text{ MEETS } b_i).$$

□

Lemma 1 tells us that we have to compute the access probability for every single data bucket $b_i \in B$. To this end, we first determine the *query-sensitive environment* of every bucket b_i ($b_i.reg.env(\mathcal{RQM})$, for short) which is the subset of the data space where queries "meeting" b_i are located. It depends on the actual PATTERN-, ASPECT RATIO- and SIZE-parameters defined in the underlying model $\mathcal{RQM}$ as well as on the data region $b_i.reg$. Second we compute the probability of the queries located in $b_i.reg.env(\mathcal{RQM})$.

The following Lemma assures that a query-sensitive environment is a closed domain in the mathematical sense and, hence, is completely described by its boundary which can be derived from the boundary of the corresponding data region. It provides a notion of query-sensitive environments and – as we shall see in the next sections – strongly facilitates their computation.

Lemma 2. Let b be a data bucket, $b.reg$ a convex domain and $b.reg.env(\mathcal{RQM})$ the query-sensitive environment of b in range query model $\mathcal{RQM}$. Symbol ∂ denotes the boundary operator.

(a) $b.reg.env(\mathcal{RQM})$ is a convex domain.

(b) $b.reg.env(\mathcal{RQM})$ is closed iff $b.reg.env(\mathcal{RQM}) \cap \partial S = \emptyset$, i.e. the environment does not contact the boundary of data space S.

(c) Let p be a boundary point of $b.reg.env(\mathcal{RQM})$. Then there exists at least one query within model $\mathcal{RQM}$ which is located at p and contacts $b.reg$ iff $p \in b.reg.env(\mathcal{RQM})$.

□

Due to space limitations we omit the lengthy and rather technical proof. The interested reader is referred to [Pag95].

3 Intersection and Containment Queries

Spatial query processing consists of a filter and a refinement step. During the filter step, data buckets are identified which eventually contain objects belonging to the answer set. The filter step must guarantee that every object matching the query condition is actually found, and that the search is guided by data regions only.

Concerning an intersection query, a data bucket must be inspected iff its data region intersects the query. Unfortunately, the filter step does not gain from the stronger geometric predicate of containment queries. As for intersection queries, a bucket has to be accessed iff its data region is intersected, because any intersected part may contain complete objects (e.g. point objects). From these observations we immediately get

$$Prob(Q \text{ MEETS } b) = Prob(Q \cap b.reg \neq \emptyset) \tag{5}$$

for both, intersection and containment queries. This means that we treat intersection and containment queries in an equal manner and discuss them in common. To this end, we interpret the query models from above as intersection, resp. containment, models and write $\mathcal{IQM}$ instead of $\mathcal{RQM}$.

3.1 Points

Point queries are window or circle queries with area 0. Nevertheless, we devote a separate section to them, because they are often discussed as separate query type in literature and we use them to smoothly introduce our analysis technique.

A data region $b.reg$ is intersected by a point query $q_\bullet$ iff $q_\bullet \in b.reg$. With $q_\bullet \equiv q_\bullet.centroid$ we conclude

$$b.reg.env(\mathcal{IQM}_\bullet) = b.reg, \tag{6}$$

i.e. data regions and their query-sensitive environments coincide.

The probability of a random point query being located in a certain data region just equals the area of that region, because S is the unit square and in model $\mathcal{IQM}_\bullet$ each location in S is equally likely to be requested. Altogether we get (Eq. 5 and 6)

$$\begin{aligned} & Prob(IQ_\bullet \text{ MEETS } b) \\ = \;& Prob(IQ_\bullet.centroid \in b.reg.env(\mathcal{IQM}_\bullet)) \\ = \;& Prob(IQ_\bullet.centroid \in b.reg) \\ = \;& b.reg.area \end{aligned} \tag{7}$$

and by Lemma 1 and Eq. 4

$$\mathcal{PM}(\mathcal{IQM}_\bullet, B) = \underbrace{\sum_{i=1}^{m} b_i.reg.area.}_{\text{sum of region areas}} \tag{8}$$

Eq. 8 exhibits that the point query performance of a spatial access structure is determined by the sum of the data region areas.

3.2 Lines

Each line query $q_\measuredangle$ is well-defined by its length c_{len}, $q_\measuredangle.centroid \in S$, and orientation $q_\measuredangle.\alpha$ (see Fig. 1). In model $\mathcal{IQM}_\measuredangle$, orientations may vary between 0 and π, such that every possible line of length c_{len} is considered. Hence, the query-sensitive environment

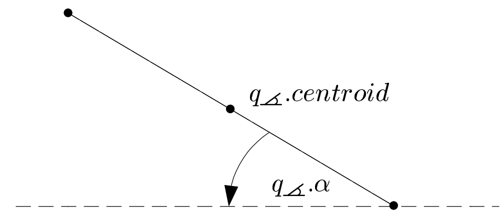

Figure 1: Line query parameters

$b.reg.env(\mathcal{IQM}_\measuredangle)$ of a bucket b is the set of all points in S with a distance to $b.reg$ less than $\frac{c_{len}}{2}$.

Fig. 2 depicts two query-sensitive environments in model $\mathcal{IQM}_\measuredangle$: The environment of bucket b_1 does not cause any conflict with the data space boundaries. On that other hand, the region of bucket b_2 is closely located to the lower right corner of the data space, such that points outside the data space also satisfy the distance condition. Of course, these points are illegal query centroids and not part of the environment.

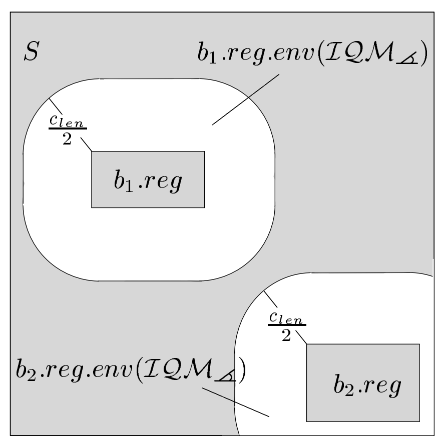

Figure 2: Environments of model $\mathcal{IQM}_\measuredangle$

Whether a line located in the query-sensitive environment actually intersects the region, depends on its current orientation. For this reason, we cannot proceed analogously to point queries, but have to make a little detour. First we compute $Prob(IQ_\measuredangle \cap b.reg \neq \emptyset)$ for lines with an arbitrary but constant orientation α and then integrate over all possible α (from 0 to π).

If we abstain from boundary considerations in favour of readability, then Fig. 3 illustrates the α-dependent query-sensitive environment $b.reg.env(\mathcal{IQM}_\measuredangle, \alpha)$ embedded into the corresp. α-independent environment. The area of the α-dependent environment again equals the bucket's access probability (assuming the projection of $\mathcal{IQM}_\measuredangle$ onto α) and can easily be computed by partitioning the domain into parallelograms as suggested in Fig. 3:

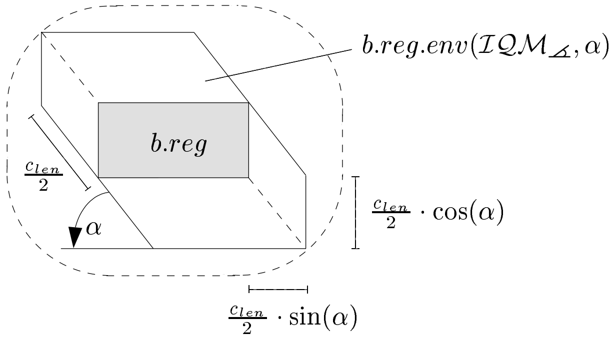

Figure 3: Environment of model $\mathcal{IQM}_{\measuredangle}$ with fixed orientation

$$
\begin{aligned}
& b.reg.env(\mathcal{IQM}_{\measuredangle}, \alpha).area \\
& \quad \approx \quad c_{len} \cdot ((b.reg.u_1 - b.reg.l_1) \cdot |\sin\alpha| \\
& \qquad\qquad + (b.reg.u_2 - b.reg.l_2) \cdot |\cos\alpha|) \\
& \qquad + b.reg.area.
\end{aligned} \tag{9}
$$

Integration over α yields

$$
\begin{aligned}
& Prob(IQ_{\measuredangle} \cap b.reg \neq \emptyset) \\
& \quad \approx \quad \frac{1}{\pi} \cdot \int_0^{\pi} b.reg.env(, \mathcal{QM}_{\measuredangle}\alpha).area \; d\alpha \\
& \quad = \quad b.reg.area + \frac{c_{len}}{\pi} \cdot b.reg.perimeter
\end{aligned} \tag{10}
$$

and summation over all data buckets finally results in the performance formula for line queries:

$$
\begin{aligned}
& \mathcal{PM}(\mathcal{IQM}_{\measuredangle}, B) \\
& \quad \approx \quad \sum_{i=1}^{m} b_i.reg.area \\
& \qquad + \underbrace{\frac{c_{len}}{\pi} \cdot \sum_{i=1}^{m} b_i.reg.perimeter.}_{\text{weighted sum of region perimeters}}
\end{aligned} \tag{11}
$$

As for point queries the sum of the data region areas occurs as parameter in the performance formula. In addition, the region perimeters enter the scene and become the more important the longer the query line.

3.3 Circles

The environment $b.reg.env(\mathcal{IQM}_{\bigcirc})$ of a bucket b is the set of all points in S with a distance to $b.reg$ less than the radius $\sqrt{\frac{c_{area}}{\pi}}$ of the query circles (see Fig. 4).

We recognize that models $\mathcal{IQM}_{\measuredangle}$ and $\mathcal{IQM}_{\bigcirc}$ lead to similar environments, but in contrast to line queries, circle queries are orientation free. Each circle located in $b.reg.env(\mathcal{IQM}_{\bigcirc})$ forces the access of b and the probability of such a random query $IQ_{\bigcirc}$ equals the area of the environment. We get

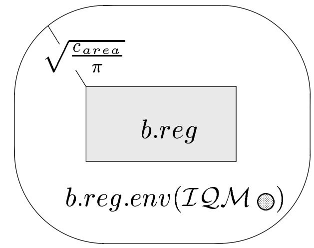

Figure 4: Environment of model $\mathcal{IQM}_{\bigcirc}$

$$
\begin{aligned}
& Prob(IQ_{\bigcirc} \cap b.reg \neq \emptyset) \\
& \quad \approx \quad b.reg.env(\mathcal{IQM}_{\bigcirc}).area \\
& \quad = \quad b.reg.area + \sqrt{\frac{c_{area}}{\pi}} \cdot b.reg.perimeter + c_{area}.
\end{aligned} \tag{12}
$$

Summing up yields

$$
\begin{aligned}
& \mathcal{PM}(\mathcal{IQM}_{\bigcirc}, B) \\
& \quad \approx \quad \sum_{i=1}^{m} b_i.reg.area \\
& \qquad + \sqrt{\frac{c_{area}}{\pi}} \cdot \sum_{i=1}^{m} b_i.reg.perimeter \\
& \qquad + \underbrace{c_{area} \cdot m.}_{\text{weighted number of buckets}}
\end{aligned} \tag{13}
$$

Compared to Eq. 11 we notice a different weight of the perimeter sum and a third summand, the weighted number of regions, resp. buckets. Apart from that, both formulas are identical.

3.4 Windows

Since window query model $\mathcal{IQM}_{\square}$ has been analysed elsewhere [PSTW93, KF93], we briefly recall the results. The query-sensitive environment of a bucket b is just its data region inflated by a frame of width $\frac{\sqrt{c_{area}}}{2}$ and the access probability equals the environment area. Hence,

$$
\begin{aligned}
& \mathcal{PM}(\mathcal{IQM}_{\square}, B) \\
& \quad \approx \quad \sum_{i=1}^{m} b_i.reg.area \\
& \qquad + \frac{\sqrt{c_{area}}}{2} \cdot \sum_{i=1}^{m} b_i.reg.perimeter \\
& \qquad + c_{area} \cdot m.
\end{aligned} \tag{14}
$$

3.5 Rectangles

The same problem as with model $\mathcal{IQM}_{\measuredangle}$ arises for squares with arbitrary orientations: the access probability of a bucket is not equal to the environment area. Fortunately, the idea used in $\mathcal{IQM}_{\measuredangle}$ is also applicable to rectangles.

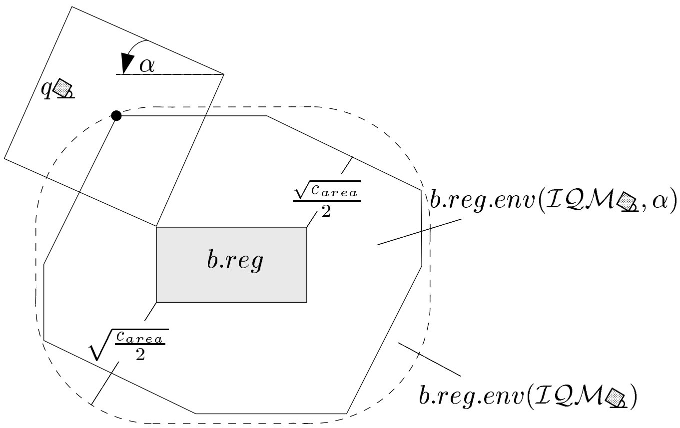

Figure 5: Environment of model $\mathcal{IQM}_{\Box}$ with resp. without constant orientation

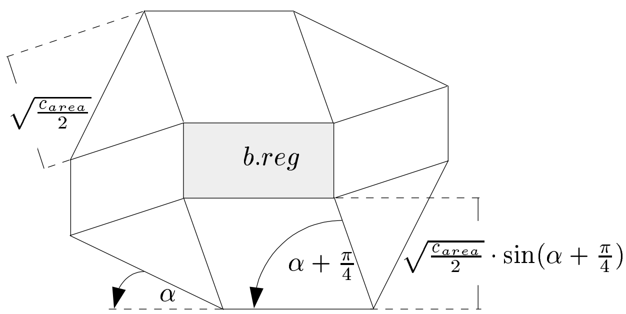

Figure 6: Partitioning $b.reg.env(\mathcal{IQM}_{\Box}, \alpha)$

Fig. 5 shows $b.reg.env(\mathcal{IQM}_{\Box})$ and the corresp. environment assuming an arbitrary but constant orientation α. Computing the area of $b.reg.env(\mathcal{IQM}_{\Box}, \alpha)$ via appropriate parallelogram partitioning yields

$$
\begin{aligned}
&b.reg.env(\mathcal{IQM}_{\Box}, \alpha).area \\
&\quad\approx\ b.reg.area \\
&\quad\qquad + \sqrt{\frac{c_{area}}{2}} \cdot \sin\alpha + \frac{\pi}{4} \cdot b.reg.perimeter \\
&\quad\qquad + c_{area} \\
&\quad=\ b.reg.area \\
&\quad\qquad + \frac{\sqrt{c_{area}}}{2} \cdot (\sin\alpha + \cos\alpha) \cdot b.reg.perimeter \\
&\quad\qquad + c_{area}
\end{aligned}
\tag{15}
$$

and by integration

$$
\begin{aligned}
&Prob(IQ_{\Box} \cap b.reg \neq \emptyset) \\
&\quad\approx\ \frac{2}{\pi} \cdot \int_0^{\frac{\pi}{2}} b.reg.env(, \mathcal{QM}_{\Box}\alpha).area\ d\alpha \\
&\quad=\ b.reg.area \\
&\quad\qquad + \frac{2 \cdot \sqrt{c_{area}}}{\pi} \cdot b_i.reg.perimeter \\
&\quad\qquad + c_{area}.
\end{aligned}
\tag{16}
$$

Summation gives the formula for $\mathcal{IQM}_{\Box}$:

$$
\begin{aligned}
&\mathcal{PM}(\mathcal{IQM}_{\Box}, B) \\
&\quad\approx\ \sum_{i=1}^{m} b_i.reg.area \\
&\quad\qquad + \frac{2 \cdot \sqrt{c_{area}}}{\pi} \cdot \sum_{i=1}^{m} b_i.reg.perimeter \\
&\quad\qquad + c_{area} \cdot m.
\end{aligned}
\tag{17}
$$

3.6 Diamonds

The performance formula for model $\mathcal{IQM}\diamond$ is directly derived from Eq. 15 by substitution and summation. For $\alpha = \frac{\pi}{4}$ we get

$$
\begin{aligned}
&\mathcal{PM}(\mathcal{IQM}\diamond, B) \\
&\quad\approx\ \sum_{i=1}^{m} b_i.reg.area \\
&\quad\qquad + \sqrt{\frac{c_{area}}{2}} \cdot \sum_{i=1}^{m} b_i.reg.perimeter \\
&\quad\qquad + c_{area} \cdot m.
\end{aligned}
\tag{18}
$$

3.7 Discussion

Before turning to enclosure queries, we briefly discuss our formulas for intersection and containment queries. The analysis reveals that for each range query model the performance of spatial access structures is determined by three parameters, namely the sum of the data region areas, the sum of the data region perimeters and the number of regions. What surprises are the regularities of the parameter weights as presented in Table 1.

- In all cases, the area sum is weighted by 1.
- Point query performance depends only on the region areas. With increasing dimensionality other parameters gain importance, namely the region perimeter for line queries and the number of buckets for 2-dimensional queries. Similarly, in 3-dimensional space the surface becomes a fourth performance parameter. In general, each additional dimension is reflected by exactly one additional parameter.

Model	Weight of Area Sum	Weight of Perimeter Sum	Weight of Region Number
$\mathcal{IQM}_\bullet$	1	0	0
$\mathcal{IQM}_\measuredangle$	1	$\frac{1}{\pi} \cdot c_{len}$	0
$\mathcal{IQM}_\square$	1	$\frac{1}{2} \cdot \sqrt{c_{area}}$	c_{area}
$\mathcal{IQM}_\circ$	1	$\frac{1}{\sqrt{\pi}} \cdot \sqrt{c_{area}}$	c_{area}
$\mathcal{IQM}_\boxdot$	1	$\frac{2}{\pi} \cdot \sqrt{c_{area}}$	c_{area}
$\mathcal{IQM}_\diamond$	1	$\frac{1}{\sqrt{2}} \cdot \sqrt{c_{area}}$	c_{area}

Table 1: Weights of the performance parameters for intersection, resp. containment, queries

- The weight of the number of regions equals the query area.
- For 2-dimensional queries, only the perimeter weights vary.
- The weight of the perimeter sum, resp. region number, grows sublinear, resp. linear, with the query size. Hence, these parameters are the more relevant the larger the query.
- Assuming queries of equal size, the weight of the perimeter sum depends on the similarity of query shape and data regions. Window squares, for instance, show the greatest similarity with box regions resulting in the smallest weight. On the contrary, diamond query shapes which are quite different from box regions, result in an approximately 40% greater weight. While it is not completely surprising that the number of bucket accesses depends on the similarity of queries and regions, it is an interesting new fact that similarity affects only the perimeter-dependent parts of the performance formulas.

We conjecture that some regularities also apply to polygon queries, which can not be modeled appropriately due to limitations of stochastic geometry [San76].

4 Enclosure Queries

A query q which is enclosed by an object $g \in b$ is also enclosed by the data region $b.reg$ (which then encloses g). Hence, we have to access every data bucket whose region encloses the query. Thus for an enclosure query EQ we get

$$Prob(EQ \text{ MEETS } b) = Prob(EQ \cap b.reg = EQ). \quad (19)$$

Since the analysis of enclosure queries can be performed analogously to that of intersection queries, we just sketch rectangular queries in detail.

4.1 Points

Obviously, $Prob(IQ_\bullet \cap b.reg \neq \emptyset) = Prob(EQ_\bullet \cap b.reg = EQ_\bullet)$, such that

$$\mathcal{PM}(\mathcal{EQM}_\bullet, B) = \mathcal{PM}(\mathcal{IQM}_\bullet, B). \quad (20)$$

4.2 Windows

The derivation of the performance formula for model $\mathcal{EQM}_\square$ starts as usual with the determination of the query-sensitive environment of a bucket b. Interestingly, computations of $b.reg.env(\mathcal{EQM}_\square)$ and $b.reg.env(\mathcal{IQM}_\square)$ turn out to be dual problems: If we assume that $b.reg$ is the environment $\hat{b}.reg.env(\mathcal{IQM}_\square)$ of a hypothetical bucket $\hat{b}$, then $\hat{b}.reg$ is exactly the environment $b.reg.env(\mathcal{EQM}_\square)$, we are actually looking for. Fig. 7 illustrates this correlation.

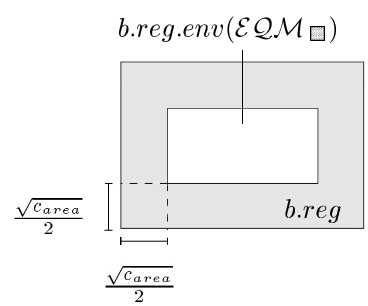

Figure 7: Duality of models $\mathcal{IQM}_\square$ and $\mathcal{EQM}_\square$

This observation can easily be generalized. Environments of enclosure models $\mathcal{EQM}$ fulfilling the assumptions of Lemma 2 are reduced to environments of the corresp. intersection model $\mathcal{IQM}$ as follows:

$$\begin{aligned} & b.reg.env(\mathcal{EQM}) \\ &= \bigcup \{ r \in S \mid r \ domain \\ & \qquad \wedge r.env(\mathcal{IQM}) \subseteq b.reg \}. \end{aligned} \quad (21)$$

In model $\mathcal{EQM}_\square$ window locations are uniformly distributed in the data space and the environment area equals the access probability:

$$
\begin{aligned}
&Prob(EQ_{\square} \text{ MEETS } b)\\
&\quad = b.reg.env(\mathcal{EQM}_{\square}).area\\
&\quad = \max(0\\
&\qquad b.reg.area - \frac{\sqrt{c_{area}}}{2} \cdot b.reg.perimeter + c_{area}).
\end{aligned} \tag{22}
$$

Note that Eq. 22 is always accurate, even if a data region is close to a data space boundary, because $b.reg.env(\mathcal{EQM}) \subseteq b.reg$. It also reflects that no bucket has to be accessed if the query is large compared to the data regions. For sake of brevity, we assume $q.area \ll b.reg.area$ and get

$$
\begin{aligned}
&\mathcal{PM}(\mathcal{EQM}_{\square}, B)\\
&\quad = \sum_{i=1}^{m} b_i.reg.area\\
&\qquad - \frac{\sqrt{c_{area}}}{2} \cdot \sum_{i=1}^{m} b_i.reg.perimeter\\
&\qquad + c_{area} \cdot m.
\end{aligned} \tag{23}
$$

Comparison with the corresp. formula for intersection queries (see Eq. 14) shows a rather astonishing result: Both formulas are identical apart from the sign of the perimeter weight. We will discuss this phenomenon later and first address one further model.

4.3 Rectangles

Model $\mathcal{EQM}_{\boxslash}$ is handled analogously to model $\mathcal{IQM}_{\boxslash}$, i.e. we fix the orientation α, compute the area of the α-dependent environment, integrate over α, and sum up. Fig. 8 depicts the environment $b.reg.env(\mathcal{EQM}_{\boxslash}, \alpha)$ of a bucket b. It also shows the maximum environment inside $b.reg$ in the corresp. intersection model which has been used for construction according to Eq. 21.

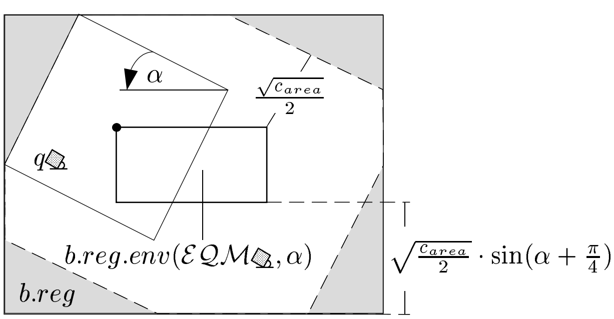

Figure 8: Query-sensitive environment $b.reg.env(\mathcal{EQM}_{\boxslash}, \alpha)$

The area of $b.reg.env(\mathcal{EQM}_{\boxslash}, \alpha)$ is given by

Model	Weight of Area Sum	Weight of Perimeter Sum	Weight of Region Number
$\mathcal{EQM}_{\bullet}$	1	0	0
$\mathcal{EQM}_{\measuredangle}$	1	$-\frac{1}{\pi} \cdot c_{len}$	$\frac{2}{\pi} \cdot {c_{len}}^2$
$\mathcal{EQM}_{\square}$	1	$-\frac{1}{2} \cdot \sqrt{c_{area}}$	c_{area}
$\mathcal{EQM}_{\bigcirc}$	1	$-\frac{1}{\sqrt{\pi}} \cdot \sqrt{c_{area}}$	$\frac{4}{\pi} \cdot c_{area}$
$\mathcal{EQM}_{\boxslash}$	1	$-\frac{2}{\pi} \cdot \sqrt{c_{area}}$	$(1 + \frac{2}{\pi}) \cdot c_{area}$
$\mathcal{EQM}_{\Diamond}$	1	$-\frac{1}{\sqrt{2}} \cdot \sqrt{c_{area}}$	$2 \cdot c_{area}$

Table 2: Weights of the performance parameters for enclosure queries

$$
\begin{aligned}
&b.reg.env(\mathcal{EQM}_{\boxslash}, \alpha).area\\
&\quad = b.reg.area\\
&\qquad - \frac{\sqrt{c_{area}}}{2} \cdot (\sin\alpha + \cos\alpha) \cdot b.reg.perimeter\\
&\qquad + c_{area} \cdot ((\sin\alpha + \cos\alpha))^2.
\end{aligned} \tag{24}
$$

Integration and summation finally results in

$$
\begin{aligned}
&\mathcal{PM}(\mathcal{EQM}_{\boxslash}, B)\\
&\quad \approx \sum_{i=1}^{m} b_i.reg.area\\
&\qquad - \frac{2 \cdot \sqrt{c_{area}}}{\pi} \cdot \sum_{i=1}^{m} b_i.reg.perimeter\\
&\qquad + \left(1 + \frac{2}{\pi}\right) \cdot c_{area} \cdot m.
\end{aligned} \tag{25}
$$

Compared to intersection queries, the sign of the perimeter weight again has alternated. In addition, we notice the increased weight of the number of regions.

4.4 Discussion

Similar considerations for the remaining query models reveal that all formulas are determined by the three well-known performance parameters, namely the area sum, weighted perimeter sum and weighted number of the data regions. Table 2 gives an overview of the model-dependent weights.

We briefly discuss the results and in particular relate them to the corresponding formulas for intersection queries. Some interesting regularities come to light:

- Formulas for enclosure and intersection queries have a lot in common. The area sum is weighted by 1, with increasing dimensionality new parameters enter the scene, and (the absolute value of) the weights of the perimeter sum, resp. region number, grow sublinear, resp. linear, with the query size.

- Apart from the sign the weights of the perimeter sum equal the according weights for intersection queries. Hence, in contrast to intersection queries where small data region perimeters lead to good performance, long and narrow data regions are of advantage. This effect is intuitively clear: Favorable queries have to be very small in order to fit into narrow regions.

- The weights of the region number are no longer identical, but vary with the query shapes. We observe a certain tendency: the more similar regions and queries, the smaller the weights. Hence, for enclosure queries perimeter and region number weights behave in opposite ways with respect to the similarity of regions and queries.

We conjecture that some regularities also apply to polygon queries.

5 How Representative Are Window Queries?

So far we have derived explicit performance formulas for intersection, containment and enclosure queries assuming a couple of different query shapes. We are now in a position to address the central question of the representativeness of window queries. We start with some theoretical considerations based on the performance formulas and end up with some experimental results.

5.1 Analytical Considerations

From the constellation of the parameter weights it is immediately clear that the different formulas are not directly proportional. Formally speaking, window query performance is therefore not representative for other kinds of range queries. However, the following investigation allows for some more sophisticated conclusions. Consider two arbitrary spatial access structures $AS_1(G)$ and $AS_2(G)$ with bucket sets B_1 and B_2. Now we ask under what circumstances $AS_1(G)$ outperforms $AS_2(G)$ with respect to a window query model and at the same time loses to $AS_2(G)$ with respect to another query model. For example, considering models $\mathcal{IQM}_{\square}$ and $\mathcal{IQM}_{\Diamond}$ the following system of inequalities has to be solved:

$$\begin{aligned} \mathcal{PM}(\mathcal{IQM}_{\square}, B_1) &< \mathcal{PM}(\mathcal{IQM}_{\square}, B_2) \\ \wedge\ \mathcal{PM}(\mathcal{IQM}_{\Diamond}, B_1) &> \mathcal{PM}(\mathcal{IQM}_{\Diamond}, B_2). \end{aligned} \tag{26}$$

Let $\Delta m = m_1 - m_2$ denote the difference of the cardinality of B_1 and B_2, $\Delta area$, resp. $\Delta perimeter$, denote the difference of the area, resp. perimeter, sums. Rearranging Eq. 26 yields

$$\begin{aligned} & \Delta perimeter > 0 \\ \wedge\ & \frac{1}{2} < -\left(\frac{\Delta area + c_{area} \cdot \Delta m}{\sqrt{c_{area}} \cdot \Delta perimeter}\right) < \frac{2}{\pi}. \end{aligned} \tag{27}$$

Since almost all promising spatial access structures provide the same bucket utilization of about 70% we assume $\Delta m \approx 0$ and get

$$\begin{aligned} & \Delta perimeter > 0 \\ \wedge\ & \frac{1}{2} < \frac{-\Delta area}{\sqrt{c_{area}} \cdot \Delta perimeter} < \frac{2}{\pi}. \end{aligned} \tag{28}$$

We see that the window query performance is representative if $\Delta perimeter \leq 0$ or $\Delta area$, $\Delta perimeter \geq 0$. Moreover, window query performance tends to be representative if query and data region size differ significantly (c_{area} small or large). Similar conditions hold for the other intersection query models [Pag95].

Assuming $c_{area} \ll b.reg.area$ on average, analogous considerations for enclosure models $\mathcal{EQM}_{\square}$ and $\mathcal{EQM}_{\Diamond}$ yield:

$$\begin{aligned} & \Delta perimeter < 0 \\ \wedge\ & \frac{1}{2} < \frac{-\Delta area}{\sqrt{c_{area}} \cdot \Delta perimeter} < \frac{2}{\pi} \end{aligned} \tag{29}$$

which is very likely to be satisfied (for $c_{area} \ll b.reg.area$). If $c_{area} \approx b.reg.area$ or $c_{area} \geq b.reg.area$ the representativeness of window query performance is straightforward:

$$\mathcal{PM}(\mathcal{EQM}_{\square}, B_1) = 0 = \mathcal{PM}(\mathcal{EQM}_{\square}, B_2). \tag{30}$$

Focusing on spatial access structures avoiding the creation of long and narrow data regions (this holds for all promising access structures like LSD-tree, R-tree etc.) and exploiting the inherent correlation of area and perimeter, three main observations can be derived:

1. Considering queries which are small or large compared to the average region size, window query performance can widely be regarded as representative for other range queries of equal size.

2. Considering queries which are approximately as large as an average data region, the representativeness of window query performance is not guaranteed, but the performance of other range queries will not differ noticeably.

3. The performance ranking of two structures with respect to large intersection queries will be inverted

if small enclosure queries are considered – except both structures provide identical perimeter sums.

5.2 Experimental Results

In this section, we complement our theoretical investigation with some experimental results. We use two data sets, namely the complete Sequoia 2000 regional benchmarks [SFGM93] for points and non-point objects. Buckets may accommodate 512 objects each.

Our first experiments deal with point data. To this end, we have implemented the LSD-tree for points [HSW89] with four different split heuristics, namely the well-known MEAN- and MEDIAN-heuristics [HS91], the RADIX-heuristic which continuously halfs the data regions, and the MINVARIANCE-heuristic which distributes objects in such a way that the variance of their coordinates in the split dimension becomes minimal.

(Bounding boxes of) Non-point objects are maintained in R-trees as well as LSD-trees for rectangles using the transformation technique [PST93]. More precisely, we have chosen the R-tree with the classical QUADRATIC-split heuristic [Gut84] and the LSD-tree with the MINVARIANCE-heuristic applied to bounding box centers, the CENTER-split described in [PST93], and the R*-heuristic adapted from [BKSS90].

The charts in Figures 9 and 10 show the intersection, resp. enclosure, performance of these access structures according to our six query models and midsized, resp. small, queries. Notice that in the 2-dimensional case the performance rankings remain stable if intersection queries on the on hand and enclosure queries on the other hand are considered. The rankings are nearly reversed, however, when comparing them against each other. These experimental results substantiates our theoretical observations.

6 Final Remarks

We have analysed the performance of spatial access structures with respect to intersection, containment, and enclosure queries of different shapes and derived performance formulas which are independent of access structure and implementation details. The formulas significantly facilitate performance comparisons in practice, because the performance of a structure can easily be computed, i.e. it is no longer necessary to generate and perform a whole bunch of test queries.

The analytical and experimental investigations reveal that the performance of spatial access structures depends much more on the query size and whether intersection or enclosure queries are considered, than on the query shape. To a wide extent, our results prove the conjecture that window queries are representative for range queries in general. However, a fair comparison of two access structure still requires a carefully chosen spectrum of query sizes to be tested.

Strictly speaking, our recommendations only hold for access structures using box regions. Similar analysis for other kinds of data regions is still an open problem. However, we do not expect any surprise here.

Acknowledgments. The authors wish to thank Elisa Kwon and James Frew for making the Sequoia 2000 benchmark available to them.

References

[BF95] A. Belussi and C. Faloutsos. Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In *Proc. 21st Intl. Conf. on VLDB*, pages 299–310, Zurich, 1995.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proc. ACM SIGMOD*, pages 322–331, Atlantic City, NJ, 1990.

[Gut84] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. ACM SIGMOD*, pages 47–57, Boston, 1984.

[HS91] A. Henrich and H.-W. Six. How to split buckets in spatial data structures. In G. Gambosi, M. Scholl, and H.-W. Six, editors, *Geographic Database Management Systems*, pages 212–244, Capri (Italy), May 1991. Esprit Basic Research Series DG XIII, Springer-Verlag.

[HSW89] A. Henrich, H.-W. Six, and P. Widmayer. The LSD-tree: spatial access to multidimensional point- and non-point objects. In *Proc. 15th VLDB*, pages 45–53, Amsterdam, 1989.

[KF93] I. Kamel and C. Faloutsos. On packing R-trees. In *Proc. 2nd Intl. Conf. on Information and Knowledge Management*, pages 490–499, Washington D.C., 1993.

[NHS84] J. Nievergelt, H. Hinterberger, and K.C. Sevcik. The grid file: an adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, 9(1):38–71, 1984.

[Pag95] B.-U. Pagel. *Analysis and Optimization of Index Structures in Geometric Databases.* PhD thesis, University of Hagen, Germany, August 1995. in german.

[PST93] B.-U. Pagel, H.-W. Six, and H. Toben. The transformation technique for spatial objects revisited. In D. Abel and B.-C. Ooi, editors,

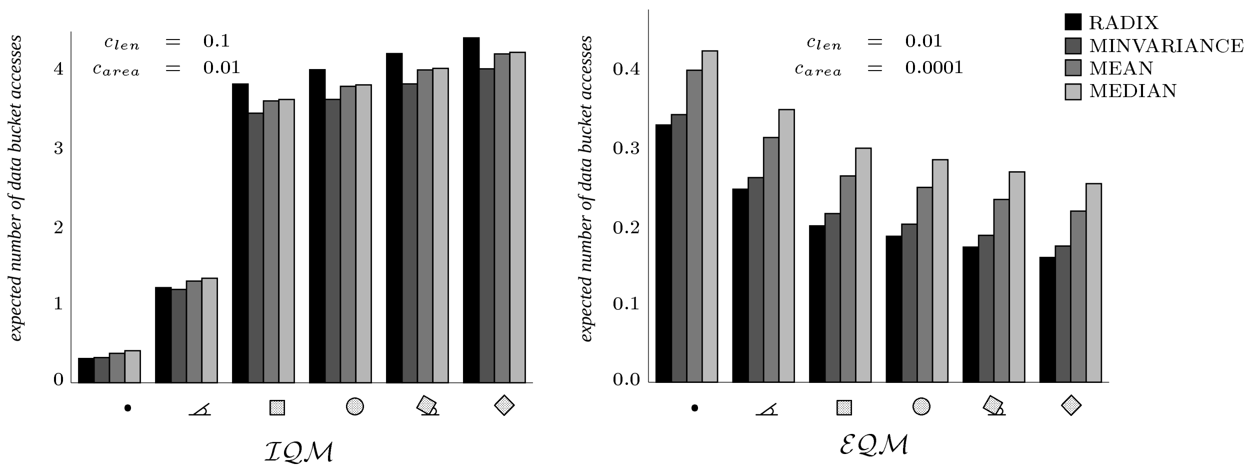

Figure 9: Performance comparison of spatial access structures for points

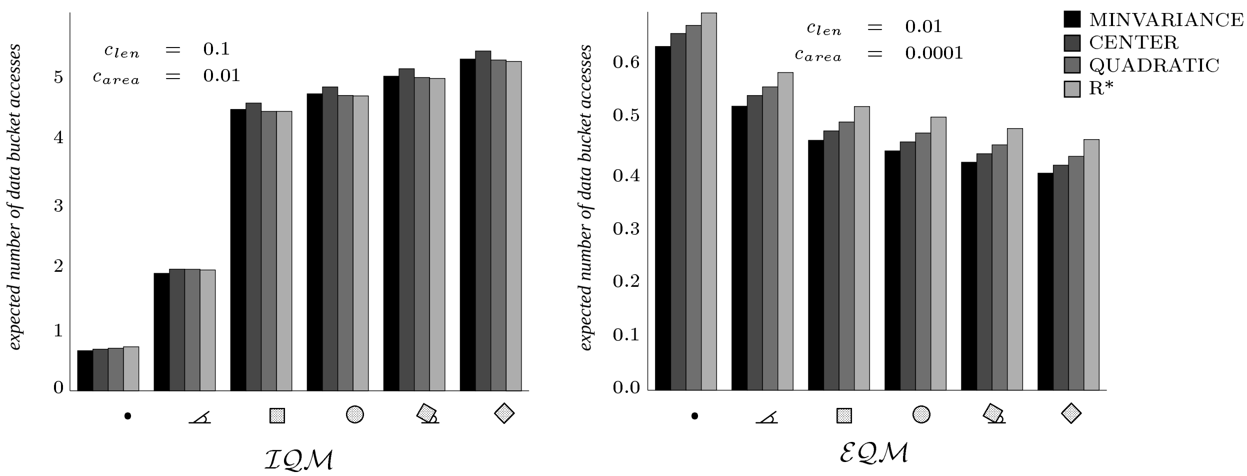

Figure 10: Performance comparison of spatial access structures for rectangles

Proc. 3rd Int. Symposium on Large Spatial Databases (SSD), pages 73–88, Singapore, June 1993. LNCS No. 692, Springer.

[PSTW93] B.-U. Pagel, H.-W. Six, H. Toben, and P. Widmayer. Towards an analysis of range query performance in spatial data structures. In *Proc. 12th ACM PODS*, pages 214–221, Washington, D.C., 1993.

[PSW95] B.-U. Pagel, H.-W. Six, and M. Winter. Window query-optimal clustering of spatial objects. In *Proc. 14th ACM PODS*, pages 86–94, San José, CA, 1995.

[San76] L.A. Santaló. *Integral Geometry and Geometric Probability*. Addison-Wesley, 1976.

[See91] B. Seeger. Performance comparison of segment access methods implemented on top of the buddy-tree. In *Proc. 2cd Symposium on Large Spatial Databases (SSD)*, pages 277–296, Zurich, 1991.

[SFGM93] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith. The SEQUOIA 2000 storage benchmark. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 2–9, Washington, D.C., May 1993.

[SK90] B. Seeger and H.-P. Kriegel. The buddy-tree: an efficient and robust access method for spatial data base systems. In *Proc. 16th VLDB*, pages 590–601, Brisbane, 1990.

[SRF87] T.K. Sellis, N. Roussopoulos, and C. Faloutsos. The r^+-tree: a dynamic access index for multi-dimensional objects. In *Proc. 13th VLDB*, pages 507–518, Brighton, England, 1987.