The VLDB Journal (1998) 7: 275–293

The VLDB Journal
© Springer-Verlag 1998

# Approximate similarity retrieval with M-trees

**Pavel Zezula**[1], **Pasquale Savino**[2], **Giuseppe Amato**[2], **Fausto Rabitti**[1]

[1] CNUCE-CNR, Via S. Maria, 36, 56126 Pisa, Italy; E-mail: zezula@iei.pi.cnr.it, F.Rabitti@cnuce.cnr.it
[2] IEI-CNR, Via S. Maria, 46, 56126 Pisa, Italy, E-mail: {P.Savino, G.Amato}@iei.pi.cnr.it

Edited by T. Özsu and S. Christodoulakis. Received July 7, 1998 / Accepted October 13, 1998

**Abstract.** Motivated by the urgent need to improve the efficiency of similarity queries, approximate similarity retrieval is investigated in the environment of a metric tree index called the M-tree. Three different approximation techniques are proposed, which show how to forsake query precision for improved performance. Measures are defined that can quantify the improvements in performance efficiency and the quality of approximations. The proposed approximation techniques are then tested on various synthetic and real-life files. The evidence obtained from the experiments confirms our hypothesis that a high-quality approximated similarity search can be performed at a much lower cost than that needed to obtain the exact results. The proposed approximation techniques are scalable and appear to be independent of the metric used. Extensions of these techniques to the environments of other similarity search indexes are also discussed.

**Key words:** Access structures – Distance only data – Similarity search – Approximation algorithms – Performance evaluation

## 1 Introduction

The problem of processing collections of data objects so that similarity queries can be answered efficiently has become increasingly important for many application areas, such as data compression, pattern recognition, statistics, learning theory, and above all the multimedia content-based retrieval.

Although several syntactic forms of similarity queries exist, the processing of any of these queries is based on the principle of *sorting* (or *ranking*) objects of a searched database (file) with respect to a user-specified reference query object. Typically, the ranking criterion (a similarity or dissimilarity-based measure) is fixed for a given application, but query objects change according to user needs, which results in a different ordering for different queries. The size of the response to a query is restricted by putting constraints either on the number of best (most similar) objects or on the maximum distance which the retrieved objects can have with respect to the query. However, the response sets always have a form of a *graded set*, because the importance of a retrieved object is determined by its distance from the query object.

Our approach for searching contrasts with the traditional one where Boolean queries, such as the queries known from the relational database systems, are used – the response to a Boolean query is a set where each element is of equal importance. Consequently, it is difficult to apply the traditional content-based indexing technology for similarity retrieval. New approaches are therefore needed.

The problem of executing similarity queries has been addressed by many researchers in both the theoretical and system-oriented communities. Unfortunately, completely satisfactory results have yet to be obtained. Efficiency still remains a problem for large databases, which are typical for multimedia applications. On the other hand, many expert users, or data analysts, develop and test hypotheses on approximate data first and only then do they draw final conclusions on precise information, which is more expensive to obtain. Furthermore, several (iterative) steps for performing atomic similarity queries are required in a typical process of searching in multimedia databases, because specifying an appropriate query object is not always easy. Consequently, an *approximate similarity search* may be very useful in such situations, especially if its execution is much faster.

This paper studies the problem of an approximate similarity search which forsakes some precision in exchange for improved performance. In particular, we propose three approximation hypotheses in the environment of the dynamic metric tree for similarity retrieval (Ciaccia et al. 1997) called the M-tree. The first type of approximation is bound by a user-defined relative error on the distances that the retrieved objects may have with respect to the exact answer. The second type of approximation is based on stochastically defined upper bounds on the number of best cases from which the approximated query response set should be retrieved. The third type of approximation is based on a pragmatic observation that the precise response is obtained through multiple search steps which improve the precision of previous approximate answers; the grades of improvements decrease rapidly during the search time. All three types of approximation are experimentally tested on real and synthetic data sets, and their efficiencies are compared. The results obtained are encouraging, and efficiency improvements of even two orders of magnitude have been achieved. In other words, whereas a precise similarity search may take several minutes, in an approximated search this can be reduced to seconds, while the precision of approximation typically remains quite high. Although better performance improvements are achieved for small sets of nearest neighbors, different metrics can be used, and the larger the files, the better the performance improvements.

This article is organized as follows. In Sect. 2, we discuss the background and objectives of our research. Three approximation methods are specified and independently analyzed in Sects. 3, 4, and 5, respectively. A comparison of the proposed approximations is the subject of Sect. 6, in which also the aspects of extensibility and scalability are studied. Section 7 concludes the paper.

## 2 Background and objectives

Depending on the applications, similarity is expressed by the closeness of objects defined by a *distance function*, $d$, which, for a pair of objects (or their representative features) from the domain $D$, provides a non-negative real value, $d : D^2 \to \mathscr{R}_0^+$. The distance function is generally of arbitrary complexity, and the problem of similarity retrieval can always be solved in linear time through a simple *brute-force* search. After computing distances from a query object to all objects in a given collection, similarity queries can easily be decided by sorting objects in terms of their distances. However, it has always been an important research challenge to find algorithms which would manage, possibly in dynamic data environments, sets of objects so that searches can be performed in a sub-linear time.

Typically, a similarity query has either a form of the *range* or of the *nearest neighbor(s)* query. Though different in their search algorithms, these query types are closely related. In fact, a range query retrieves all objects contained in a specific, query-object-related region. On the other hand a $k$ nearest neighbors query, $k$-NN for short, retrieves $k$ closest objects of the query object; thus it determines the minimum region in which these neighbors can be found. In any case, retrieved sets are generally *graded* (or *ranked*) sets, because some of their objects may be closer to the reference query object, thus more important, than the others.

### 2.1 The theoretical approach

The problem of similarity retrieval has also been one of the fundamental subjects in the field of *computational geometry*. This theoretical area concerns the design and analysis of efficient algorithms for computing various properties and parameters of finite configurations of geometric objects. In this way, it forms foundations of many applications. A survey on geometric range searching can be found in Matousek (1994). However, there are several fundamental differences between the objectives and approaches that are considered as standard in the field of computational geometry and the needs of advanced data-indexing applications such as multimedia. The specific arguments can be summarized as follows.

1. In computational geometry, $n$-dimensional vector spaces with usually not very high $n$ (from 2 to 20) are typical, and the $L_m$*-distance* between objects (points) is used to measure dis-similarities. Notice that $L_1, L_2$, and $L_\infty$ are the well-known Manhattan, Euclidean, and MAX metrics. Though such environments are also relevant in the case of multimedia data indexing, the dimension needed is typically much higher. Moreover, more complex metrics are also relevant in the multimedia environment, such as the quadratic form functions (Seidl and Kriegel 1997), Hausdorff metric over sets of $n$-dimensional points (Huttenlocker et al. 1993), or the Levenshtein (*edit*) distance over strings (Hall and Dowling 1980).
2. Data sets studied in the field of computational geometry are typically static. In particular, it is assumed that the data is known in advance and never changes. However, multimedia data collections change in time; generally growing, but also shrinking, since database objects may be deleted.
3. In theoretical research, the efficiency of algorithms is measured, almost exclusively, by the asymptotic order of growth in their complexity. However, asymptotic efficiencies do not always agree with how fast such algorithms compute in real computer environments. The efficiency of database algorithms is better expressed in terms of elapsed time to perform a transaction, or, since such time is practically impossible to express for complex systems in an analytic way, in terms of (estimated or measured) I/O and CPU costs.

### 2.2 Indexing structures for similarity retrieval

Developing index (or storage) structures which would support efficient evaluation of queries for data from complex geometric spaces has long been a challenge in the database research community. As far as $n$-dimensional vector spaces are concerned, probably the most successful approach is the R-tree by Guttman (1994), with its more advanced modifications, such as the R$^*$-tree (Beckmann et al. 1990), SS-tree (White and Jain 1996), or X-tree (Berchtold et al. 1996).

Generic metric spaces, that is geometric spaces in which only the *positivity*, *symmetry*, and *triangle inequality* distance postulates are necessary, have been considered in designs such as the VP-tree by Chiueh (1994), GNAT by Brin (1995), or MVP-tree by Bozkaya and Ozsoyoglu (1997). Though different in the underlying principles used for data partitioning, all these designs assume static data files. Since processing dynamic files is an important feature of index structures, Ciaccia et al. (1997) have designed a paged, dynamic, and balanced tree, called the M-tree. Notice that collections of generic metric data are sometimes called *distance-only* data, because only distances between pairs of objects can be quantified – data objects are not necessarily embedded in any multi-dimensional space.

## 2.3 Objectives

Alhough there are several index structures that can support the execution of similarity queries, experience with their applications shows that the processing costs are still very high. For example, similarity retrieval in high-dimensional data spaces tends to access most or even all similarity index tree nodes (sometimes designated as the *dimensionality curse*).

Another important fact is that the specification of a query object $Q$, needed as a reference object in similarity queries, is something users find quite difficult. This problem is easy to understand if a space of many dimensions is considered. Consequently, several queries need to be executed before a good query object is found. Initial retrieval steps are needed to find a more suitable query object. This iterative approach to query processing makes the total costs of a query execution even higher, while the need for exact answers is not always relevant.

The vagueness in query specification together with the high query execution costs led us to investigating the following idea.

*Provided an approximate similarity search can be performed much faster than the precise search, the approximate similarity retrieval can play a useful role in the global process of iterative similarity retrieval.*

In other words, since processing exact similarity queries is not always required, approximate answers would suffice, especially when obtained at much lower costs.

Computational geometry researchers (Arya et al. 1994) have also addressed the problem of approximate similarity searching. However, to the best of our knowledge, no dynamic indexing structure for approximate similarity retrieval has been proposed. In this paper, we develop three alternative techniques for the $k$-NN query approximation. Though the asymptotic execution costs of all our approximated search techniques are identical to the cost of the exact similarity search for the same query, we demonstrate through experiments that a high speed-up can be obtained even for quite precise approximations.

Although our proposals are of general validity, we study the problem of approximate similarity search in the context of M-trees, so that different approximation hypotheses can be compared through experiments on actual data sets. Another reason for choosing the M-tree environment is that it is the most general dynamic index structure for similarity retrieval.

Specific hardware configurations can affect experiments by factors such as the simultaneous processing of several jobs, the inability of most operating systems to accurately allocate clock cycles to processes, and caching effects. To avoid problems with the accuracy of measured query elapsed times we measure the query-processing costs in terms of the number of accessed M-tree nodes and the number of distance computations. More precisely, since the number of read nodes and distance computations is correlated (in tree organizations, distances are only computed for objects of accessed nodes), costs in all our graphs represent the number of distance computations. This measurement gives higher numbers, and is thus more precise.

Due to the lack of any standard experimental data (or a benchmark), we choose three different data sets and report on the averages obtained from many test runs in order to increase confidence in the results obtained.

## 2.4 Principles of M-trees

An M-tree, can be seen as a hierarchy of metric (ball) regions. A region is defined by a database object $O_i$ and radius $r(O_i)$, which represents the maximum distance between $O_i$ and any other object, including its region (if any), in the region of $O_i$. An M-tree is a multiway-branching tree; thus, each node can contain several object entries which are all members of a region centered around a *parent object*, $O_p$, stored in a higher level node. Notice that the region of objects from the root is assumed to be the entire universe, because these objects do not have any actual parent object. Each entry is represented by the object's features and, in the case of non-leaf entries, by their region radii which restrict minimum regions in which all descendant objects and/or regions can be found. For efficiency reasons, *child-to-parent* object distances, computed during the tree construction phase, also form a part of the objects' entries.

### 2.4.1 Similarity search strategies

Pruning sub-trees, or reducing search costs by avoiding distance computations and node accesses, is the primary concern of the M-tree similarity search algorithms. As mentioned above, we only consider the more general case of similarity retrieval, that is $k$-NN queries, which can be characterized by the following sketch of the algorithm. Implicitly, the number of objects in the file is assumed to be no smaller than $k$.

```
**k-NN algorithm**
*initialize-set:* consider any sub-file of size $k$ as the response
    set, and assign to $r(Q)$ the distance between $Q$ and the
    $k$-th nearest neighbor of this set (i.e., the largest distance).
    Put a pointer to the M-tree root node (region of objects) into
    the priority queue $PQ$. Pointers in $PQ$ are ordered according
    to the *proximity* of their corresponding regions with respect to
    the query region $(Q, r(Q))$, which is dynamically shrinking
    during the search time.
*purify-set:* while there are entries in $PQ$ with a positive
    proximity (i.e., the query and entry regions intersect), **do**
    **access node:** access a node determined by a pointer located
        at the top of $PQ$;
    **test entry:** for each entry in the accessed node, **do**
        **test object:** if the object of this entry can improve
            the response set, update the response set and adjust $r(Q)$;
        **test region:** if the region of this entry intersects
            $(Q, r(Q))$, put the pointer to this region into $PQ$;
*response set:* the current response set contains $k$ nearest
    neighbors to $Q$.
```

Note that the algorithm consists of two phases, namely the initialization and purification of the required response set. However, though the first phase is typically very fast

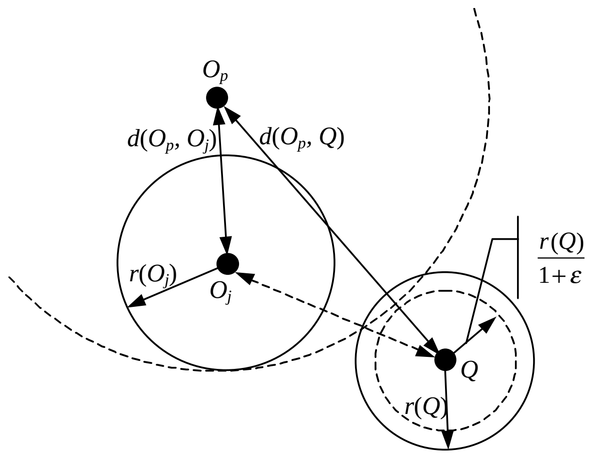


**Fig. 1.** Pruning principles of the M-tree

and a rather rough approximation of the desired response is expected, the second phase typically needs many node reads and distance computations.

To understand these algorithms suppose that the entry for object $O_j$, with its parent object $O_p$ and its region of radius $r(O_j)$, is to be processed (see also Fig. 1). Since the search algorithm is strictly hierarchical – a parent object needs to be processed before its child object can be considered – the distance $d(Q, O_p)$ is known. The distance $d(O_j, O_p)$ is also known because it is a part of $O_j$'s entry.

In principle, $O_j$ may qualify as a new member of the result set, but to decide exactly, the distance $d(O_j, Q)$ must be computed. Also, provided the query and $O_j$'s regions intersect, the pointer to the descendant node of $O_j$ must be put into $PQ$, because it might include better qualifying objects (see the **test object** and **test region** steps of the algorithm above).

To minimize retrieval costs, caused by distance computations and disk (node) reads, the search algorithm of M-trees uses two kinds of (preliminary) tests on object entries. As Ciaccia et al. (1997) explained and proved for correctness, the corresponding tests are the following.

1. $| \, d(O_p, Q) - d(O_j, O_p) \, | > r(Q) + r(O_j)\,.$ (1)
   If this test is satisfied, the region $(O_j, r(O_j))$ can be ignored without even computing the distance to $O_j$.
2. $d(O_j, Q) > r(Q) + r(O_j)\,.$ (2)
   If this test is satisfied, the pointer to the region $(O_j, r(O_j))$ is not inserted into the priority queue, thus the descending node is never accessed[1].

Obviously, provided that $d(Q, O_j) < r(Q)$, then $O_j$ is a new member of the result set and the query radius should appropriately be reduced. Notice, however, that Eq. 1 can also be used as a preliminary test of $O_j$'s qualification by considering $r(O_j) = 0$, that is only considering the object and ignoring its region. The cost minimization principles of the M-tree are also illustrated in Fig. 1.

The execution costs of the $k$-NN algorithm above naturally depend on the query point $Q$ and specific distribution of objects in a given tree. Notice that the same file can result in many still correct, forms of M-trees. However, due to the properties of this algorithm, the following cost-related properties can still be defined.

Given an M-tree, the number of accessed nodes (regions of objects) for $k$-NN search with respect to Q is determined by the number of regions which intersect the query region with radius $r(Q) = d(Q, O_N^k)$, where $O_N^k$ is the $k$-th nearest neighbor of $Q$. More precisely, a region with the parent object $O_p$ is actually accessed from the priority queue if

$$d(Q, O_p) \leq r(O_p) + d(Q, O_N^k)\,. \quad (3)$$

Those interested in M-trees should, besides Ciaccia et al. (1997), also refer to the following articles: Ciaccia and Patella (1998) on algorithms for bulk loading the M-trees; Zezula et al. (1998) presenting practical experiences with a parallel version of the M-tree; Ciaccia et al. (1998a) suggesting an extension of M-tree for processing multiple (single-feature) similarity predicates; Ciaccia et al. (1998b) proposing a cost model for similarity queries in metric spaces.

### *2.5 The simulation testbed*

To evaluate and compare our strategies of approximation, we use three qualitatively different files of 45-dimensional vectors, each one of the size $n = 10,000$. The first file, designated as CHV, represents color features of images and was chosen as a representative of real-life files. Color features are in fact 9-dimensional vectors containing the average, standard deviation, and skewness of pixel values for each of the red, green, and blue channels (see Stricker and Orengo 1995). An image is divided into five overlapping regions, each one represented by a 9-dimensional color feature vector, which gives a 45-dimensional vector as a descriptor of each image. The distance function is based on the Euclidean ($L_2$) metric. Notice that other approaches to image color representations use *color histograms*. In such cases, similarity searches typically use weighted measures of similarity because of the crosstalk between similar colors. In our experiments, we use a method that requires the Euclidean distance in order to be consistent with the metrics adopted for the other data sets.

The other two files contain synthetic data in order to simulate rather extreme and orthogonal cases. Specifically, the second file, designated as UV, contains vectors which are uniformly distributed in a 45-dimensional unit hyper-cube, while the third file, designated as CV, is formed, in the same space, by 10 randomly distributed clusters of vectors with a variance within a cluster of $\sigma = 0.05$. For simplicity, we always consider the Euclidean distance as the measure of similarity.

Query points, again 45-dimensional vectors, are not from the data files, but comply with data distributions which objects in the individual files follow. One hundred different query objects are defined for each of the files. Then, all costs reported are expressed as average values obtained from runs of all queries. We used 10-NN queries in all the experiments.

[1] Notice that this test has a meaning, provided that $O_j$ represents a region, that is when $r(O_j) > 0$.

### 2.5.1 Performance measures

To quantify a degree (or a grade) of excellence which individual approximation techniques may possess, we suggest considering as measures not only an improvement in performance efficiency, but also a quality of approximation. High improvements of the performance typically result in approximations with poorer quality, and vice versa.

The first measure, called the *improvement in efficiency*, $IE$, relates the costs of the exact and approximated searches. It is defined as

$$IE = \frac{cost(O_N^k)}{cost(O_A^k)}, \tag{4}$$

where $O_N^k$ and $O_A^k$ are, respectively, the $k$-th actual and approximated neighbors of Q as found by our search algorithms. Then, $cost(O_N^k)$ and $cost(O_A^k)$ are the corresponding execution costs to retrieve these neighbors.

Contrary to the improvement in performance, the quality of approximation is assessed through two complementary measures. We call them the *precision of approximation*, $P$, and the *relative distance error*, $\overline{\epsilon}$.

In order to define the *precision of approximation*, assume that $O_A^i$, $i = 1, 2, \ldots k$, is the approximated $i$-th nearest neighbor with respect to $Q$. Thus, the precision is defined as

$$P_i = \frac{i}{\#range(Q, d(Q, O_A^i))}, \tag{5}$$

where $\#range(Q, d(Q, O_A^i))$ is the cardinality of the set which is obtained from a file of size $n$ while performing a range query for $Q$ with radius $d(Q, O_A^i)$. Since $d(Q, O_N^k) \leq d(Q, O_A^k)$, it is true that $i \leq \#range(Q, d(Q, O_A^i))$, because the set $range(Q, d(Q, O_A^i))$ contains not only the exact response set of size $i$, but possibly also other objects. For example, when $P_{10} = 0.5$, then the 10th approximated neighbor is, in fact, the 20th actual neighbor of $Q$.

Then, the precision of a $k$-NN search is defined as

$$P = \frac{\sum_{i=1}^{k} P_i}{k} = \frac{\sum_{i=1}^{k} \frac{i}{\#range(Q, d(Q, O_A^i))}}{k}. \tag{6}$$

Notice that, when the approximated response is exact, the precision $P = 1$. On the other hand, the precision tends to 0 in the worst case.

The *relative distance error*, designated as $\overline{\epsilon}$, is defined in a similar way. Let $O_N^i$ be the $i$-th nearest neighbor of $Q$ and $O_A^i$ the $i$-th approximated neighbor of $Q$, which is assumed to be different from any object in the database[2]. Thus, obviously, $0 < d(Q, O_N^i) \leq d(Q, O_A^i)$ for all $i = 1, 2, \ldots, k$, and the expression

$$\overline{\epsilon_i} = \frac{d(Q, O_A^i)}{d(Q, O_N^i)} - 1 \tag{7}$$

gives the relative error of the $i$-th nearest neighbor. The global relative distance error is then defined as

$$\overline{\epsilon} = \frac{\sum_{i=1}^{k} \overline{\epsilon_i}}{k} = \frac{\sum_{i=1}^{k} \frac{d(Q, O_A^i)}{d(Q, O_N^i)}}{k} - 1. \tag{8}$$

[2] The relative error is not defined for the exact match where $d(Q, O_N^i) = 0$

When the approximation gives exact results, the relative error $\overline{\epsilon} = 0$.

A proper evaluation of the quality of approximation entails considering both the measures, $P$ and $\overline{\epsilon}$. In fact, we can consider the precision of approximation as a measure that only takes into consideration the ranking of database objects with respect to the query object. On the other hand, the relative distance error only relates the actual and approximated neighbor distances, irrespective of the rest of the file.

## 3 Approximation through relative distance errors

In this section, we study the case where the approximation of a nearest neighbor with respect to $Q$ is constrained by a user-defined relative distance error $\epsilon$. After outlining the idea, we show how the M-tree pruning rules can be modified and demonstrate by observations from numerous experiments how this method performs.

### 3.1 The idea

Let $O_N$ be the nearest neighbor of $Q$ and $O_A$ some other object in the searched collection. Obviously, provided $0 < d(O_N, Q) \leq d(O_A, Q)$,

$$\frac{d(O_A, Q)}{d(O_N, Q)} = 1 + \epsilon \tag{9}$$

defines that the distance from $Q$ to $O_A$ is $(1 + \epsilon)$ times the distance from $Q$ to $O_N$. Now, assume that $O_A$ is the *approximated* nearest neighbor of $Q$. In such case, $\epsilon$ represents the *relative error* of the distance approximation, that is of considering $O_A$ as the nearest neighbor of $Q$ instead of $O_N$.

Naturally, the relative error of the distance between $Q$ and any database object with respect to $d(Q, O_N)$ is non-negative, but in order to consider an object as the approximate neighbor of $Q$, a user-defined bound on the relative error must be respected. Provided this error is $\epsilon$, the following constraint must hold for $O_A$:

$$\frac{d(O_A, Q)}{d(O_N, Q)} \leq 1 + \epsilon. \tag{10}$$

This idea can be generalized to the case of $k$-NN search, for $1 \leq k \leq n$, where $n$ is the size of the database. Using $O_A^k$ and $O_N^k$ to designate the $k$-th approximated and the nearest neighbors, the constraint should be modified as follows:

$$\frac{d(O_A^k, Q)}{d(O_N^k, Q)} \leq 1 + \epsilon. \tag{11}$$

If this constraint is satisfied, $O_A^k$ is called the $(1 + \epsilon)$ $k$-NN of $Q$. However, though $d(Q, O_N^k)$ is unique for a given $Q$, there may be several objects in the database which, when considered as $O_A^k$, satisfy Eq. 11. That means that the candidate set for approximate results is not necessarily singular. In the limit case, $d(Q, O_A^k) = d(Q, O_N^k)$ or even $O_A^k = O_N^k$.

### 3.2 Approximate pruning constraints

To see how the search pruning tests of M-trees can be relaxed by respecting a tolerable relative error $\epsilon$, consider another form of Eqs. 1 and 2 as follows:

$$\frac{r(Q)}{\mid d(O_p,Q) - d(O_j,O_p) \mid - r(O_j)} < 1 \tag{12}$$

and

$$\frac{r(Q)}{d(O_j,Q) - r(O_j)} < 1\,. \tag{13}$$

Looking at these fractions, the numerators specify, by the query point radius, the distance to the ($k$-th) nearest neighbor of $Q$ discovered so far. For convenience, see also the **k-NN algorithm** in Sect. 2.4.1. Provided the search has not finished, i.e., better objects can still be found, this distance can be considered as the distance to the approximated neighbor. The denominators, on the other hand, put the *lower bounds* (using the corresponding information about distances at hand) on possible nearest neighbors in the region $(O_j, r(O_j))$ with respect to $Q$ (see again Fig. 1). In other words, the denominators represent the minimum distance that an object in the given region might have, with respect to $Q$. Naturally, if the lower bounds (i.e., the denominators) are higher than the current radius of $Q$, the region considered cannot contain any qualifying object, and therefore can be ignored in the search from this point on.

In order to modify these tests to the case of approximate search, that is, when $\epsilon > 0$, the lower bounds can be relaxed by the relative factor $\epsilon$ in the following way:

$$\frac{r(Q)}{\mid d(O_p,Q) - d(O_j,O_p) \mid - r(O_j)} < 1 + \epsilon \tag{14}$$

and

$$\frac{r(Q)}{d(O_j,Q) - r(O_j)} < 1 + \epsilon\,. \tag{15}$$

By analogy, the actual test for an object qualification can be modified as

$$\frac{r(Q)}{d(O_j,Q)} < 1 + \epsilon\,. \tag{16}$$

Naturally, relaxing these tests in the above way can never increase the similarity search costs, because both the number of distance computations and the number of node reads can only be reduced. To be more precise, consider again Eq. 3, which specifies whether a region with its parent object $O_p$ is accessed by the M-tree's search algorithm. To adjust this cost condition to our relative-distance approximation algorithm, two important facts must be reconsidered. First, the distance to the approximated $k$-th nearest neighbor may be larger than the distance to the exact one, thus, restricted by

$$d(Q,O_N^k) \le d(Q,O_A^k) \le d(Q,O_N^k) \times (1+\epsilon)\,. \tag{17}$$

Second, all the pruning tests consider that the actual neighbors, with respect to the approximate, are $(1+\epsilon)$ times smaller, so Eq. 3, which determines whether the region $(O_p, r(O_p))$ is actually accessed or not, can be modified as

$$d(Q,O_p) \le r(O_p) + d(Q,O_A^k)/(1+\epsilon)\,. \tag{18}$$

However, due to the properties of approximated neighbors, see Eq. 17 for the range of their possible values, the approximated search can never access more nodes than the exact search, as the following example illustrates.

*Example 3.1.* Suppose a region with its center $O_p$, radius $r(O_p) = 3$, and distance to the query point $d(O_p,Q) = 4.9$. Provided the distance $d(O_N^k,Q) = 2$ and the precise similarity search is considered, this region must, according to Eq. 3, be accessed, because $4.9 < 3+2$. However, provided the approximate search with $\epsilon = 0.2$ is used, the situation is a bit more complex, because the distance to a possible approximate neighbor is constrained by $2 \le d(Q,O_A^k) \le 2\cdot(1+0.2)$ (see Restriction 17 for verification). Naturally, provided that $d(Q,O_A^k) = 2.4$, i.e., the most distant approximate nearest neighbor is found, the situation is the same as in the case of the precise nearest neighbor, and Inequality 18 orders this region to be accessed. But, if $d(Q,O_A^k) = 2$, Inequality 18 is not satisfied, because $4.9 \le 3+2/(1+0.2)$ is not true. Generally, if we consider the approximate distance $d(Q,O_A^k)$ as a variable and solve the inequality $4.9 \le 3+d(Q,O_A^k)/(1+0.2)$, we find out that our region only has to be accessed if $d(Q,O_A^k) \ge 2.28$.

Due to the fact that the actual savings depend on data as well as the specific structure of a constructed M-tree, we investigate this phenomenon by the following experiments.

### *3.3 Experimentation*

The experiments for the approximate similarity search constrained by relative distance errors $0 \le \epsilon \le 2$ are summarized for all our data files in Fig. 2. Note that the actual relative error $\overline{\epsilon}$ is much smaller than the search approximation parameter $\epsilon$. Though this was true in all the cases, it was more significant for the synthetic data files (UV and CV) rather than the color features. We attribute this behavior to the fact that the (unit) domain space in the case of synthetic data was denser compared to the possible density of our CHV file.

For higher relative errors, the precision obviously decreased. Here, however, the real-life file CHV performed better than the synthetic files, and even for $\epsilon = 0.4$, where $\overline{\epsilon}$ was 0.2, the precision was still around 0.5.

Unfortunately, the observed improvements in efficiency were not very significant. This was true above all for the range of the relative distance errors up to, say 0.5, which is the range of values for which the precision was still quite high. Note that the best improvement in efficiency with $\epsilon = 0.5$ was obtained for the CHV file ($IE = 1.2$), while for the CV file, the improvement was only 1.03, and in the case of the UV file, the improvement was practically negligible.

In summary, an approximate similarity search constrained by a relative distance error $\epsilon$ keeps, by definition, an upper bound on the actual relative error, which is typically much lower than the required constraint. However, since both of the synthetic data files performed similarly in all the aspects considered, it seems that it is the density of the search space, rather than the data distribution, which influences the performance. Less dense spaces can provide better precision and higher improvements in efficiency, but may result in approximations of higher relative errors. Unfortunately, performance improvements are typically not high, whenever the precision is important.

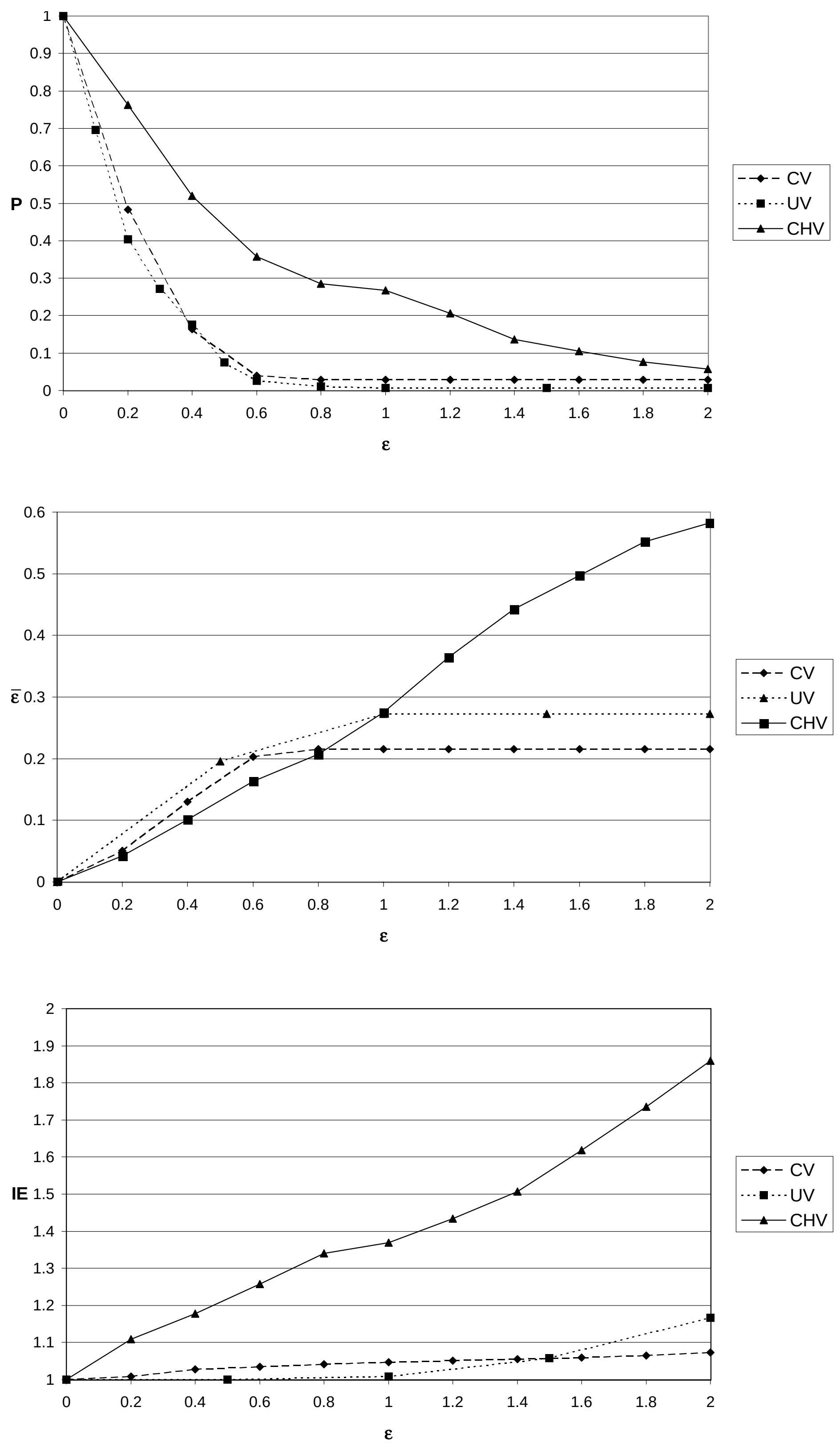


**Fig. 2.** Approximation through the relative error

## 4 Approximate search through distance distributions

Since the existence of a coordinate system, which is used in vector spaces to determine the location of objects, is not necessarily a condition for a metric, the only general way to quantify topological relationships between objects is the distance distribution. As Ciaccia et al. (1998b) argue, this distance distribution is a correct counterpart of the data distribution used for vector spaces; thus, it can be applied as a way to characterize metric data sets. In this section, we investigate a method which exploits characteristics of distance distributions to control a similarity search approximation.

### *4.1 Distance distributions*

Let $\mathscr{M} = (\mathscr{U}, d)$ be a metric space, where $\mathscr{U}$ is the value domain (or the universe), and $d$ is the measure that quantifies distances between pairs of objects from $\mathscr{U}$. The *relative distance distribution* of an object $O_i \in \mathscr{U}$ with respect to the others, or the so called $O_i$'s *viewpoint,* is defined as

$$F_{O_i}(x) = Pr\{d(O_i, \mathbf{O}) \leq x\}, \tag{19}$$

where $\mathbf{O}$ is a random object from $\mathscr{U}$.

By definition, $F_{O_i}(x)$ is a monotonic, non-decreasing function, which, for a given distance $x$, provides the probability that a randomly chosen object from $\mathscr{U}$ is within a distance from $O_i$ which is smaller than or equal to $x$. It is also important to observe that, given two objects, $O_i, O_j \in \mathscr{U}$, $F_{O_i}(x)$ may be different from $F_{O_j}(x)$, because the points of views of objects $O_i$ and $O_j$ need not be the same.

### *4.2 The concept of approximation*

Assume that the query object is $Q$ and that $F_Q(x)$ is the distance distribution of $Q$ with respect to all objects in a given database. According to Eq. 19, $F_Q(x)$ represents the *fraction* of objects in the database for which the distance to $Q$ is less than or equal to $x$. Provided there are $n$ objects in the database, $n \times F_Q(x)$ objects should have a distance to $Q$ not greater than $x$.

Now, consider the $k$-NN search algorithm again and imagine an intermediate retrieval step in which the distance to the $k$-th approximated neighbor $O_A^k$ is $d(Q, O_A^k)$. By using the distance distribution, $F_Q(d(Q, O_A^k))$ determines the fraction of the best cases (with respect to $Q$) to which this current approximate result belongs. For example, provided $F_Q(d(Q, O_A^k)) = 0.01$, the approximate result already belongs to 1% of the best cases in a given object file.

This property can easily be exploited for approximating similarity searches – a user may wish to find any $k$ objects among those belonging to the fraction of the best cases $\rho$. To this aim, the following *stop condition* can be defined

$$F_Q(d(Q, O_A^k)) \leq \rho \tag{20}$$

and used in the $k$-NN algorithm to terminate the search before the exact nearest neighbors are found. Since the search algorithm remains the same, and it is only the stop condition which might terminate the algorithm earlier, the search costs can never be higher than the costs needed to perform the exact similarity retrieval – no search improvements are obtained when $\rho \leq F_Q(d(Q, O_N^k))$.

### *4.3 Approximation through other distance distributions*

So far we have assumed that the distance distribution with respect to $Q$ is known. However, computing and maintaining this kind of information for any possible query object is completely unrealistic. A possible solution would be to find a function $F_r(x)$ that would serve as a *representative distance distribution function* for all *viewpoints*.

Provided that such a representative is found, the stop condition Eq. 20 can be substituted by

$$F_r(d(Q, O_A^k)) \leq \rho \tag{21}$$

for each query object $Q$.

The problems of *viewpoint discrepancies* or, alternatively, *homogeneity of viewpoints* have been studied by Ciaccia et al. (1998b). For instance, experiments performed with large text files have shown that the homogeneity of viewpoints is quite high, which means that distance distributions, measured with respect to different objects, are very similar.

Ciaccia et al. (1998b) also propose a methodology to decide when one distance distribution can be used rather than another. In particular, the so-called *discrepancy* between two relative distance distributions was defined as

$$\delta(F_{O_i}, F_{O_j}) = E[(\mid F_{O_i}(\mathbf{x}) - F_{O_j}(\mathbf{x}) \mid], \tag{22}$$

where $\mathbf{x}$ is a random distance in the interval $[0, d^+]$, and $d^+$ is the maximum distance between two objects of $\mathscr{U}$. The *index of homogeneity of viewpoints HV* of a metric space $\mathscr{M}$ was defined as

$$HV(\mathscr{M}) = 1 - E[\delta(F_{\mathbf{O_1}}, F_{\mathbf{O_2}})],$$

where $\mathbf{O_1}$ and $\mathbf{O_2}$ are random points of $\mathscr{U}$. When $HV(\mathscr{M}) \approx 1$, two different relative distance distributions are likely to behave similarly; thus, any $F_{O_i}(x)$ could be used as the representative $F_r(x)$.

The index of homogeneity is too general for our purposes, since it considers the discrepancy of relative distance distributions over the whole interval $[0, d^+]$. To make an approximation, we need to consider the behavior of a specific function $F_r$ in an interval $[0, d']$ where $d'$ is substantially smaller than $d^+$, since possible values of $\rho$ can typically be smaller than 0.1. Notice, however, that considering a smaller range of distances does not necessarily imply a better discrepancy than the one obtained when the total range of distances is taken into account.

The discrepancy of two relative distance distributions in an interval $[0, x]$ can be obtained by slightly modifying Eq. 22 as follows:

$$\begin{aligned}\delta(F_{O_i}, F_{O_j})(x) &= E_{[0,x]}[\mid F_{O_i}(\mathbf{x}) - F_{O_j}(\mathbf{x}) \mid] \\ &= \frac{1}{x}\int_0^x \mid F_{O_i}(y) - F_{O_j}(y) \mid dy\,.\end{aligned} \tag{23}$$

The *degree of representativeness* $RV^{F_r}$ of a given representative $F_r$ in the interval $[0, x]$ can be defined as follows:

$$RV^{F_r}(\mathscr{M})(x) = 1 - E[\delta(F_r, F_{\mathbf{O}})(x)]\,. \tag{24}$$

When $RV^{F_r}(\mathscr{M})(d') \approx 1$, then we can argue that $F_r$ is a good representative in the interval $[0, d']$.

Let us consider, for instance, as a representative the *average distribution function* defined as follows:

$$F_{avg}(x) = E[F_{\mathbf{O}}(x)]\,. \tag{25}$$

Figure 3 shows a graph of the degree of representativeness of function $F_{avg}$ for our testbed. The graph was obtained by computing for each $\rho$ the value $RV^{F_{avg}}(\mathscr{M})(G_{avg}(\rho))$,

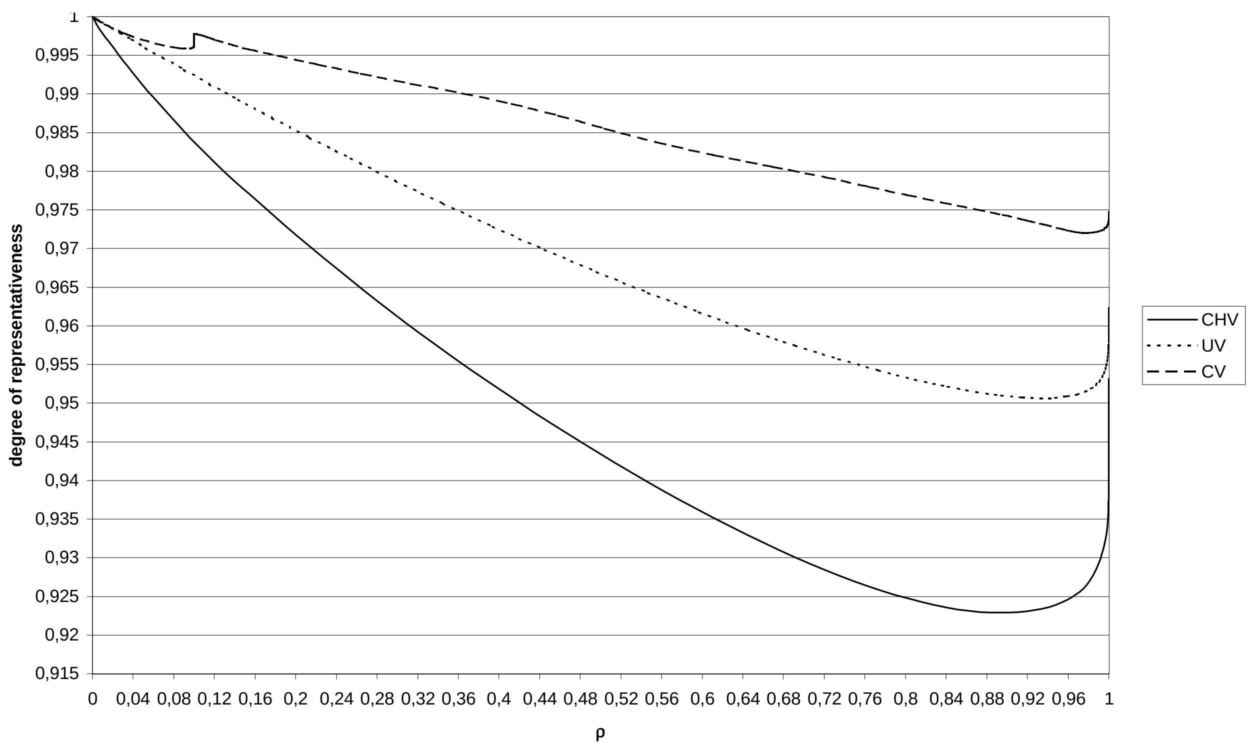


**Fig. 3.** Degree of representativeness of $F_{avg}$

where $G_{avg}$ is the inverse of $F_{avg}$, that is if $\rho$ = $F_{avg}(x)$ then $x$ = $G_{avg}(\rho)$.

It can be seen that, for values of $\rho \leq 0.1$, the degree of representativeness always has values greater than 0.99. Notice that $\rho$ = 0.1 corresponds to 10% of the whole database, so it can be used as a realistic limit for running approximate queries. Observe that, for the CV file, the degree of representativeness remains above 0.99 even for values of $\rho$ up to 0.4.

### *4.4 Experimentation*

The *precision* ($P$), the *relative error* ($\overline{\epsilon}$), and the *improvement in efficiency* ($IE$) were computed for different values of the approximation parameter $\rho$. The results are reported in Fig. 4.

Since realistic values of $\rho$ are lower than 0.1, this approach obtains good precision for all of our files. In particular, the precision seems to be very high using the UV file. In that case, it always remains above 0.2 for values of $\rho$ up to 0.09. Using the CV and CHV files, the precision falls below 0.2 for values of $\rho \geq 0.02$.

While for $\rho \leq 0.1$ the relative error for CV and UV is quite good, we obtain higher values of $\overline{\epsilon}$ for the CHV file. In particular, when $\rho$ is smaller than 0.1, the error is always smaller than 0.4 for CV and UV, while it rises up to 0.7 for the CHV file. Notice that, for the CV file, the error increases rapidly for values of $\rho$ greater than 0.1. This is due to the fact that the CV file contains clusters. Specifically, considering the distance distributions, 10% of the distances are smaller than 0.7, the remaining 90% of the distances belong to the interval [1.9,3.6]. This implies that, when $\rho$ is greater than 0.1, the approximate search may include objects that are suddenly more distant than those considered for values smaller than 0.1.

The last graph shows the measure of the improvement in efficiency. Using CHV and UV files, we obtain good improvements for small values of $\rho$. When $\rho$ = 0.025, we have an improvement of 33 times for UV, and 52 for CHV. On the other hand, for the CV file, the improvement is 34 times only when $\rho$ = 0.1.

In summary, with this method values of $\rho$ can be found that give good values of precision, relative error and improvement in efficiency for all three files considered. For example, if we want to limit the precision to $P \approx 0.3$, then values of $\rho \approx 0.01$ (CHV file), $\rho \approx 0.025$ (CV file), and $\rho \approx 0.05$ (UV file) should be used. With these values, high improvements in efficiency ($35 \leq IE \leq 50$) are obtained and the approximate results have a limited relative error ($\overline{\epsilon} \approx 0.15$ for the CV and UV files, and $\overline{\epsilon} \approx 0.3$ for the CHV file).

## 5 Approximation through the slowdown of distance improvements

The approximation method we present in this section is based on the following pragmatic observation.

**Observation 5.1.** *The M-tree's $k$-NN algorithm determines the response set for $Q$ by gradually improving an initial, approximate and potentially rough, subset of the file's objects of size $k$. Accordingly, the distance to the $k$-th (approximate) nearest neighbor $O_A^k$ shrinks during the query evaluation and finally becomes $d(O_N^k, Q)$ when $O_A^k$ = $O_N^k$. However, the early improvements are typically significant, and with a very*

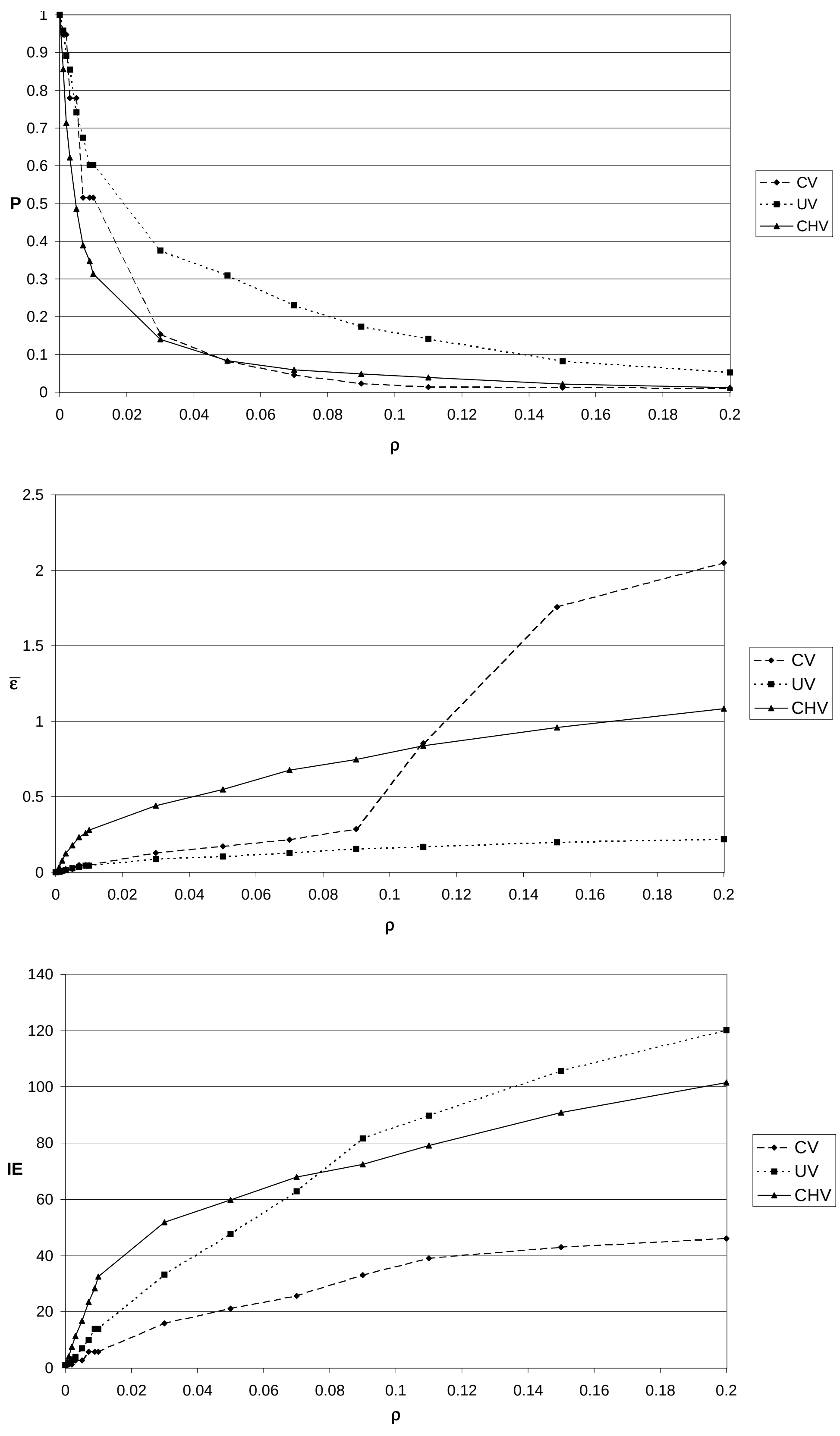


**Fig. 4.** Approximation through the distance distribution

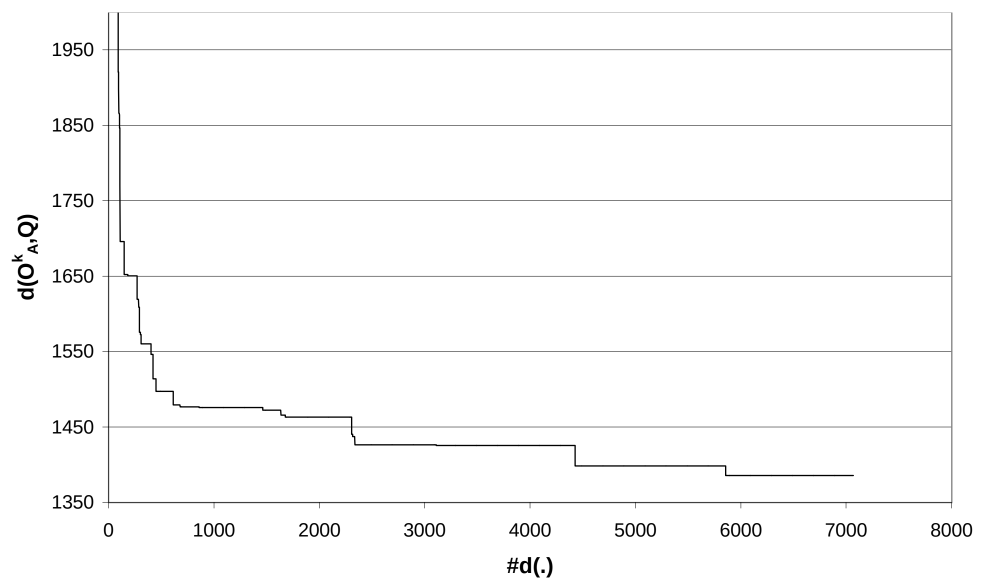


**Fig. 5.** Graph of $d(O_A^k, Q)$ vs the number of distance computations $\#d(\cdot)$

*limited computational effort a good approximation of the exact result can be obtained. On the other hand, this process slows down, and later improvements are usually minor and time consuming.*

The typical situation is illustrated in Fig. 5, which, for a certain $Q$, shows the approximate $k$-NN distance, $d(O_A^k, Q)$, and the number of necessary distance computations $\#d(\cdot)$. Obviously, the minimum value of $d(O_A^k, Q)$ is $d(O_N^k, Q)$ and, for this distance value, we have the maximum of distance computations, which represents the costs for finding the actual $k$-th nearest neighbor. Observe that at the end of this search, more than 1000 distance computations are performed just to be sure that no better nearest neighbor exists.

Since the number of node reads, or the I/O costs, and the number of distance computations are strictly correlated, identical trends have also been observed when the number of node reads instead of the number of distance computations was considered. Furthermore, a similar behavior to the one shown in Fig. 5 was observed in every experiment we performed, when changing not only query points but also the data sets.

### *5.1 The approach to approximation*

Based on Observation 5.1, the approximation method described in this section reduces search costs by executing the standard search algorithm until the variation (i.e., the reduction) of the distance between the query and the approximated $k$-NN objects becomes sufficiently low. In the following, we first clarify the idea and then describe in detail its implementation.

#### 5.1.1 The idea

Assume the function $f : \#d(\cdot) \to d(O_A^k, Q)$ as a strictly decreasing continuous function. In order to simplify the notation, we designate this function as $f(x)$ and its first derivative as $f^{'}(x)$, where $x$ represents the search costs. For convenience, a derivative of degree $\nu$ is designated as $f^{(\nu)}(x)$. Due to the assumed properties of $f(x)$, $f^{'}(x)$ decreases as $x$ increases and its value, in fact, represents the chances of finding better nearest neighbors – the lower the value of $f^{'}(x)$, the lower the possibility of improving the approximation.

Thus, given a constraint $\kappa > 0$, we can stop the approximate search as soon as $f^{'}(x) \leq \kappa$. The quality of this approximation, i.e., the error we get by stopping the search algorithm at a certain point, and the speed-up we obtain, i.e., the reduction in distance computations and I/O operations we gain, are clearly inversely proportional to the value of $\kappa$. Provided that $\kappa$ is sufficiently low and $f^{'}(x) \leq \kappa$, the approximation obtained should already be of a good quality, since the chances of finding better approximations are not high.

#### 5.1.2 Our implementation

The approximation error and the speed-up for a given value of $\kappa$ cannot be measured exactly, because they both depend on the function $f(x)$ which is not known a priori. In reality, the function $f(x)$ does not have the nice properties we have assumed above, because

(a) values of $x$ are discrete and the corresponding values of $f(x)$ only become available (for increasing $x$) as the search algorithm proceeds;
(b) $f(x)$ is a piecewise constant function, monotonically decreasing.

From an implementation point of view, (a) requires a numerical evaluation of $f^{'}(x)$, while (b) requires a careful choice of arguments which are used during the calculation of derivatives. Hereafter, we assume that the function $f(x)$ has known values in $z + 1$ points, $x_0, \ldots, x_z$, with $x_0 < x_1 < \ldots < x_z$. Since $f(x)$ is a piecewise constant and monotonically decreasing function, it has the following properties:

1. $f(x_i) \geq f(x_j)$ for $i < j$;

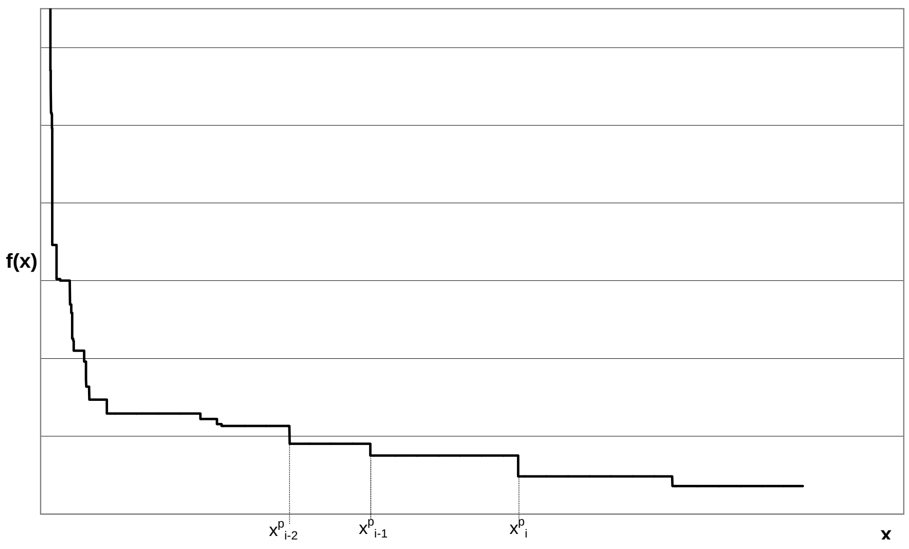


**Fig. 6.** Graph of $f(x)$ showing the potentially optimal points for stopping the k-NN approximate search

2. there exist points $x_0^p, x_1^p, x_2^p, \ldots$ (with $x_0^p < x_1^p < x_2^p < \ldots$ and $f(x_0^p) < f(x_1^p) < f(x_2^p) < \ldots$) such that, given $x_i^p \equiv x_r$, $f(x_{r-1}) > f(x_r)$.

Points $x_i^p$ are called the *potentially optimal points*. In fact, these are the points that provide the highest performance improvement for the same level of the approximation. Figure 6 gives an example of $f(x)$ and of the potentially optimal points.

The numerical evaluation of $f^{'}(x)$ is based on standard numerical analysis techniques (see for example Burden et al. 1979). Accordingly, the derivative of $f(x)$ in $x = x_s$, $s \in [0, z]$ can be calculated as follows:

$$f^{'}(x_s) = L_z^{'}(x_s) + (x_s - x_0) \cdots (x_s - x_{s-1})(x_s - x_{s+1}) \cdots (x_s - x_z) \frac{f^{(z+1)}(\xi)}{(z+1)!}\,, \tag{26}$$

where $L_z(x)$ is the Lagrange polynomial defined as

$$L_z(x) = \sum_{i=0}^{z} f(x_i) l_{z,i}(x)\,, \tag{27}$$

with

$$l_{z,i}(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_z)}{(x_i - x_0)(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_z)} \tag{28}$$

and $\xi \in [x_0, \ldots, x_z]$.

The error term is thus $(x_s - x_0) \cdots (x_s - x_{s-1})(x_s - x_{s+1}) \cdots (x_s - x_z) \frac{f^{(z+1)}(\xi)}{(z+1)!}$. Unfortunately, since we do not have any information about the analytic form of $f(x)$, no assumptions can be made on its derivatives of higher orders and on their errors. We will not take into consideration these errors.

As an example, for $z = 2$, that is when the function is known in the points $x_0, x_1, x_2$,

$$\begin{aligned} f^{'}(x_j) = {} & f(x_0)\left[\frac{(2x_j - x_1 - x_2)}{(x_2 - x_1)(x_0 - x_2)}\right] \\ & + f(x_1)\left[\frac{(2x_j - x_0 - x_2)}{(x_1 - x_0)(x_1 - x_2)}\right] \\ & + f(x_2)\left[\frac{(2x_j - x_0 - x_1)}{(x_2 - x_0)(x_2 - x_1)}\right] \\ & + \frac{1}{6} f^{(3)}(\xi_j) \prod_{i=0, i \neq j}^{2} (x_j - x_i) \end{aligned} \tag{29}$$

for each $j = 0, 1, 2$, where the notation $\xi_j$ indicates that this point depends upon $x_j$.

### 5.1.3 Terminating the approximate search

The condition used to terminate the search algorithm is based on the estimation of $f(x)$ derivatives in points $x_i^p$. We tested the quality of three different methods for calculating $f^{'}(x)$. They are based on the use of two, three, or five points, respectively. All three methods approximate $f^{'}(x)$ by using a limited number of points $x^p$ and the latest value of $x$.

Let us consider two consecutive *potentially optimal points*, $x_{i-1}^p$ and $x_i^p$, and let us also suppose that the search algorithm has run up to a point where the number of distance computations is $x_s$, where $x_s = x_i^p$. We can then calculate the value of $f(x_{s+1})$. By using two points, the derivative in $f(x_i^p)$ is estimated as the slope of a line passing through $f(x_{i-1}^p)$ and $f(x_{s+1})$. If $f(x_{s+1}) = f(x_s)$, then the derivative is compared with $\kappa$; if it is higher than $\kappa$, the function is calculated in a new point $x_{s+2}$ and the procedure is repeated, otherwise it stops. If $f(x_s) > f(x_{s+1})$, a new *potentially optimal point* $x_{i+1}^p \equiv x_{s+1}$ has been found and the procedure is repeated by using this new point. More formally, the procedure can be expressed as follows.

```
**Input:** $x_{i-1}^p$, $x_i^p$, $f(x_s)$ with $x_s = x_i^p$
**Output:** if $f^{'}(x_i^p) < \kappa$, the search terminates, otherwise a
    new *potentially optimal point* $x_{i+1}^p isfound.$
**Step 1:** calculate $f(x_{s+1})$
**Step 2:** if $f(x_{s+1}) = f(x_s)$, then
    { $f^{'}(x_i^p) = \frac{f(x_{i-1}^p) - f(x_{s+1})}{x_{i-1}^p - x_{s+1}}$;
    if $f^{'}(x_i^p) < \kappa$, then stop else $s = s + 1$; goto Step 1; }
**Step 3:** if $f(x_{s+1}) < f(x_s)$, then the new *potentially optimal
    point* $x_{i+1}^p \equiv x_{s+1}$ has been found.
```

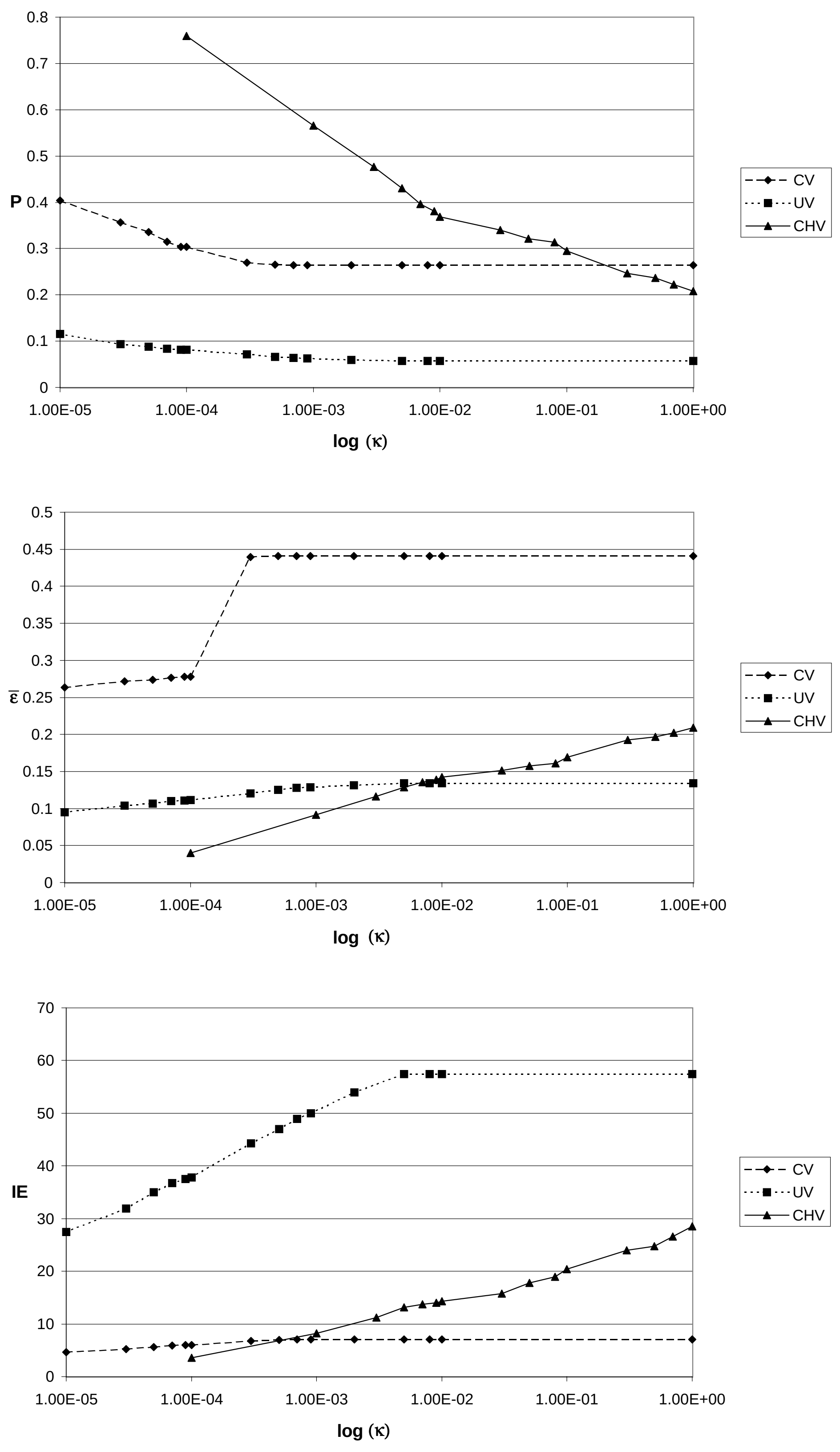
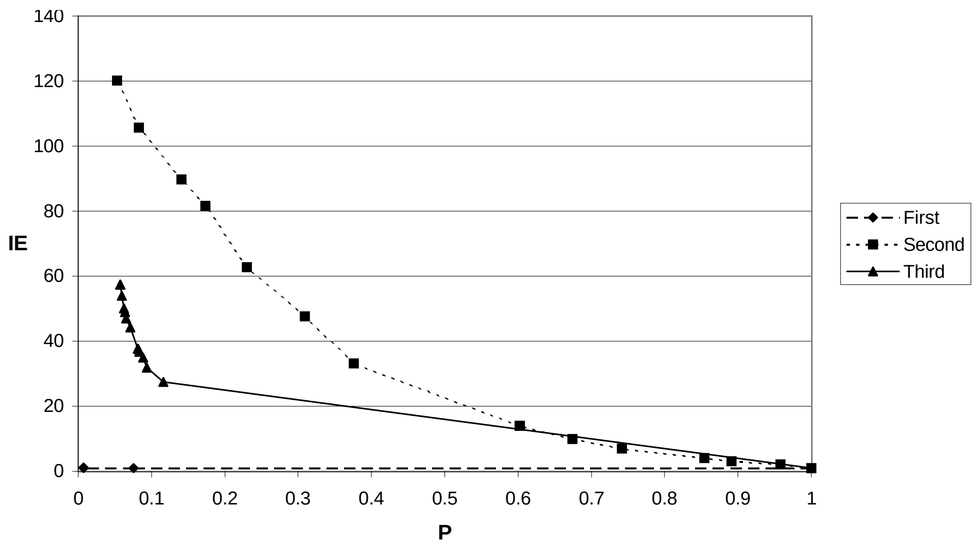


**Fig. 7.** Graphs of the *precision* ($P$), the *relative error* ($\overline{\epsilon}$) and the *improvement in efficiency* ($IE$) as a function of $log(\kappa)$

With three and five points, the derivative is estimated by using $x_{i-1}^p, x_i^p, x_s$ and $x_{i-3}^p, x_{i-2}^p, x_{i-1}^p, x_i^p, x_s$, respectively. The calculation of $f^{'}(x_i^p)$ in **Step 2** is modified, for both cases, according to Eqs. 26-28.

These three methods of numerical derivation have been evaluated by computing the relative error ($\overline{\epsilon}$) and the precision ($P$) as a function of $\kappa$. The experiments obviously entailed running the approximate $k$-NN search and the exact $k$-NN search, and comparing the distance between the query



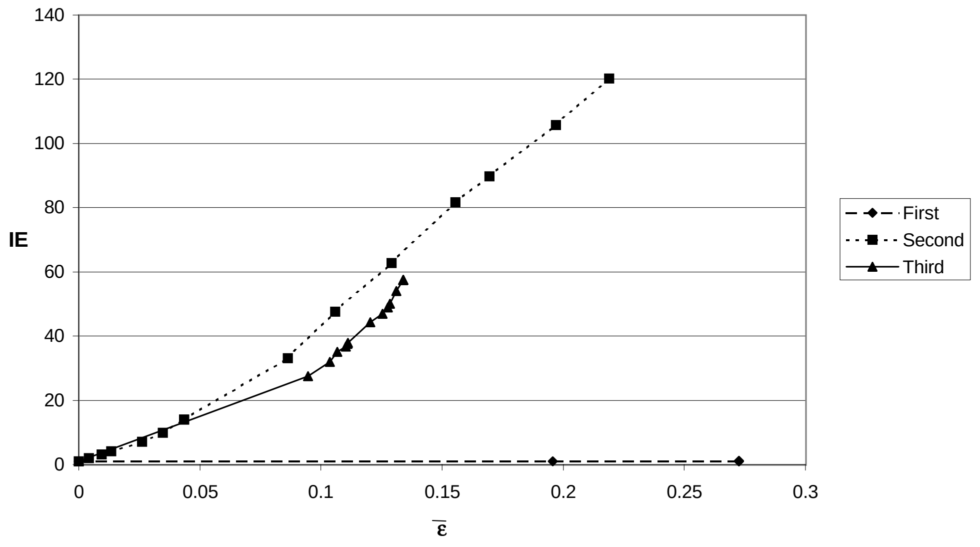


**Fig. 8.** Comparison of approximation techniques for the UV file

and the $k$-th object in the approximated result set with the distance between the query and the $k$-th object in the exact result set. The experiments showed that, although the first method is the simplest, it is far too inaccurate. Although the quality of approximation of the second method is acceptable, the third one, i.e., the method using five points, is significantly better and was adopted in all the experiments described below.

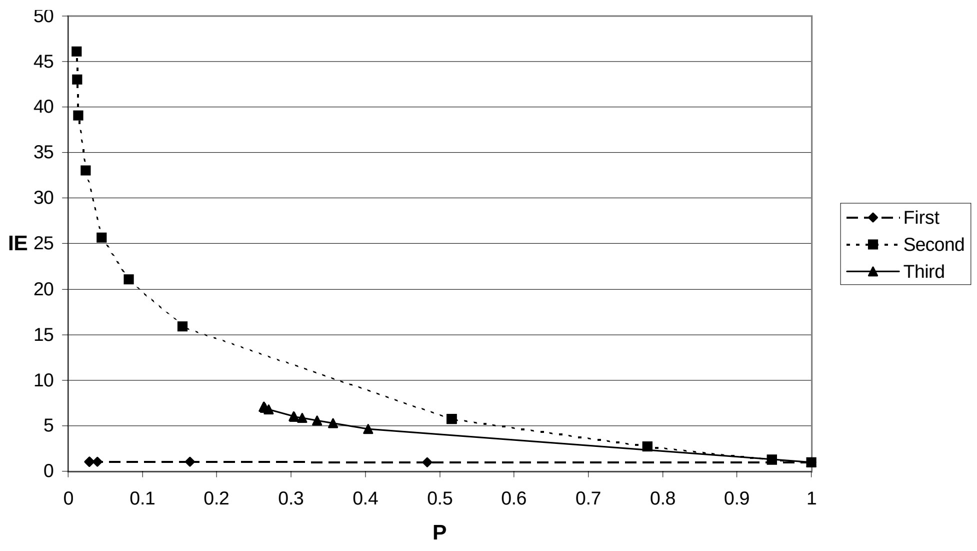
### 5.2 Experimentation

The approach to reducing distance computations described in this section was evaluated by measuring the precision $P$, the relative error $\overline{\epsilon}$, and the improvement in efficiency $IE$ as a function of $\kappa$. The results of this evaluation are reported in Fig. 7. We used a logarithmic scale for $\kappa$, because we considered a wide range of values for this parameter.

In particular, the first graph in Fig. 7 shows how precision varies with $\kappa$ for the three data files (CV, UV, and CHV) used in the experiments. The best results were obtained for the CHV file and the worst for the UV file. Note that, apart from the UV file, the values of $P$ are quite high. For example, $P$ varies in the range $[0.75, 0.20]$ for the CHV file – a very good result. Roughly speaking, we can say that $P = 0.5$ means that the first $k$ approximate nearest neighbors contain $k/2$ of the exact nearest neighbors.

The real relative error $\overline{\epsilon}$ was very good for all three data files. For the CHV file, $\overline{\epsilon}$ is in the range $[0.03, 0.19]$. To give an idea of the quality of the result, $\overline{\epsilon} = 0.1$ means that the approximate results have an average distance from the query point that is 10% higher than the exact result. The second graph of Fig. 7 highlights that the results are quite good even for the $UV$ file, contrary to what happens for $P$. This is due to the fact that in this file the objects are very close each other, which gives a limited relative error even though the precision is low.

Of particular interest are the results reported in the third graph of Fig. 7, which shows the variation of $IE$ with $\kappa$. The method allows an improvement in performance to be obtained, measured with the number of distance computations, up to 50 times. To conclude, this method can generally achieve high performance improvements while still



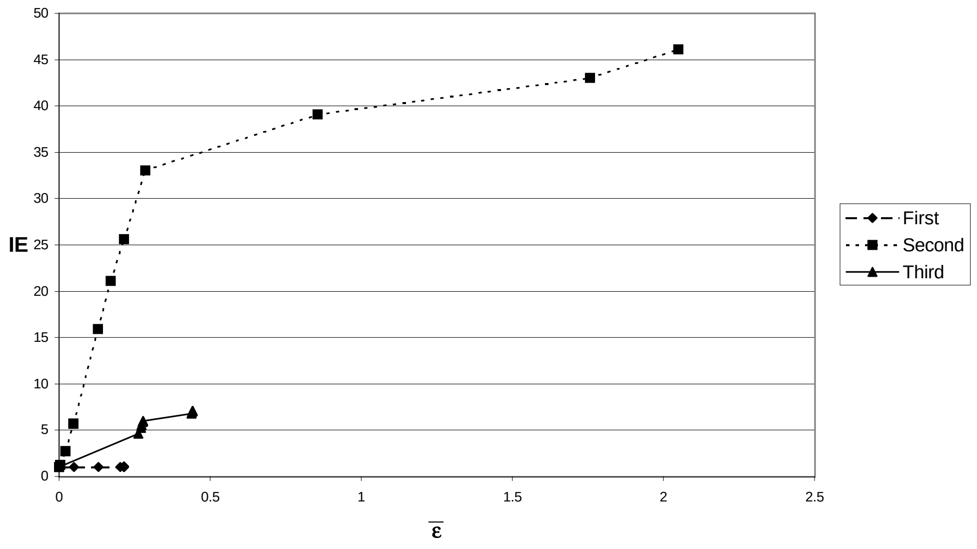


**Fig. 9.** Comparison of approximation techniques for the CV file

maintaining good quality results. For example, for the CHV file, we have $IE$ = 20 with $P$ = 0.3 and $\overline{\epsilon}$ = 0.15.

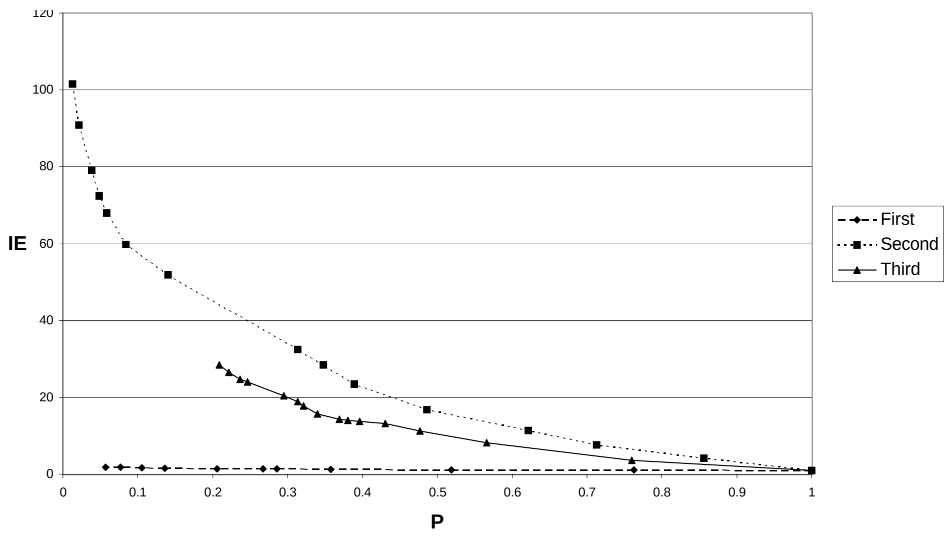
## 6 Comparisons and further performance considerations

So far, we have considered the proposed approximation techniques independently by relating their characteristic approximation parameters ($\epsilon, \rho$, and $\kappa$) with defined performance measures, that is the improvement in efficiency, the search precision, and the actual relative error. To compare our approaches to approximation, we ignore the way the approximations are specified (constrained) and only relate the improvement in efficiency, which was the main objective of our research, with the quality, which is the precision and the relative error obtained. Such comparisons were made for all the files from our testbed, and the results can be found for the uniformly distributed data in Fig. 8, for the clustered data in Fig. 9, and for the color features in Fig. 10.

Although approximation through relative errors explicitly puts upper bounds on possible relative distance errors, this method has not generally proved to be very efficient. For all our data files, the improvements in efficiency, with respect to the same quality, were practically negligible compared to those obtained by the other two approximation methods.

The experiments suggest that the best approach is approximation through distance distribution, which showed the highest improvements in efficiency. This phenomenon was especially significant when precision was considered (see for example Fig. 8, where the approximate search was performed 100 times faster, while the precision was still 0.1). Roughly speaking, we can say that instead of retrieving the exact 10 nearest neighbors, 10 objects from 100 best cases were retrieved 100 times faster. The approximation through distance distribution also performed well with respect to the actual relative distance error. Here, only in the case of the color feature vectors, did approximation through the slowdown of distance improvements achieve better results.

As far as the complexity of the methods is concerned, we can say that all of the proposed approximation methods are quite easy to implement. Except for approximation



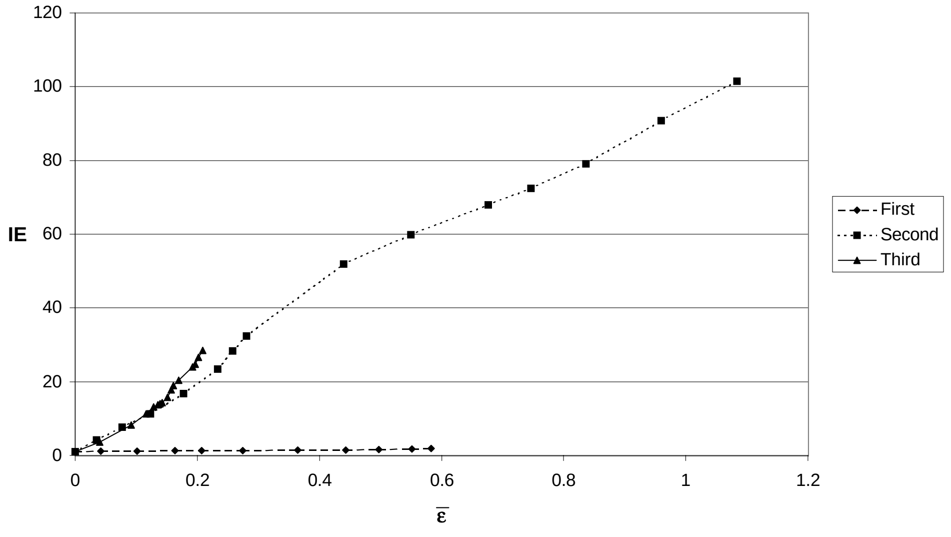


**Fig. 10.** Comparison of approximation techniques for the CHV file

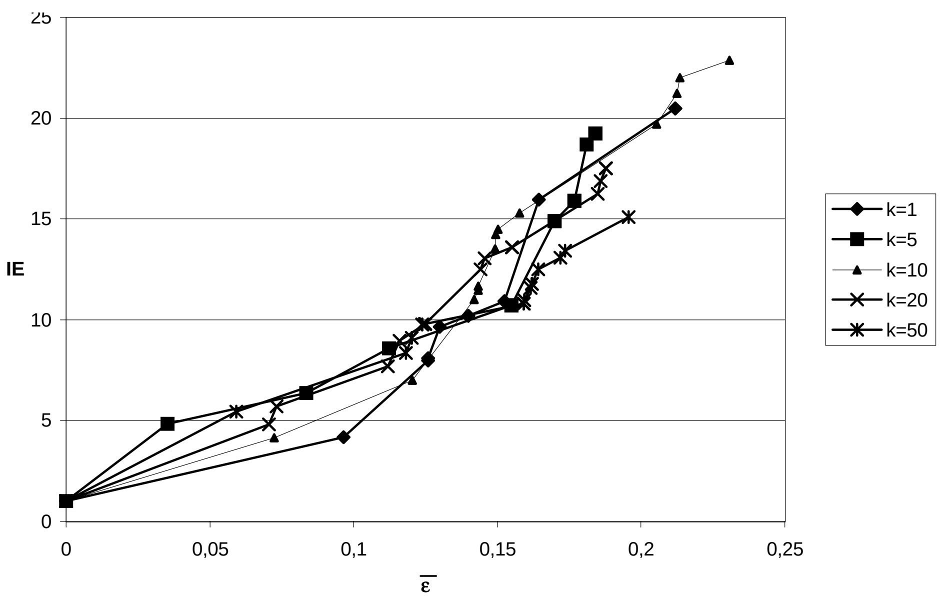
through distance distribution, no additional auxiliary data is required. Obviously, to obtain and maintain distance distributions might be an additional cost for the user or the system administrator; however, distance distributions can be obtained, for example, during the M-tree construction phase, because the insertion of objects into an M-tree requires many distance computations. Another possibility is to acquire the distance distribution by sampling on the data file. However, if a user is not willing to bear the additional costs, or if a good representative distance distribution is difficult to find, then approximation through the slowdown of distance improvements is the best option: it is easy to implement, it has proved to perform well, and it does not need any supporting data.

## 6.1 Extensibility and scalability

Though our testbed was designed to allow a comprehensive evaluation of various approximation techniques, it does not cover all the aspects needed when proposing an index structure. In particular, we have not considered *extensibility*, i.e., how far the technique can be extended to support further forms of queries, and *scalability*, i.e., the reaction of performance parameters to changing sizes of searched files. Because the first approximation technique proved not to be very efficient, hereafter we only consider the second and third approximation techniques.

So far, we have assumed that the number of requesting nearest neighbors is 10. Since the search algorithms are independent of $k$, it is important to know how the precision and efficiency change in relation to $k$. In particular, we investigated the case where $k$ changes from 1 to 50.

We now report the experimental results obtained for the CHV file when the approximation through the slowdown of performance improvements is used. Figure 11 shows two graphs, the first reports the improvement in efficiency ($IE$) as a function of the precision $P$, while the second shows the variation in $IE$ as a function of the relative error $\overline{\epsilon}$. A slight difference can be observed by comparing the results reported here for $k$ = 10 with those given in Fig. 10. This is due to the fact that a different set of queries was used in these two experiments. The graphs in Fig. 11 highlight that the improvement in performance deteriorates (though not dramatically) with increasing $k$. A similar behavior was also



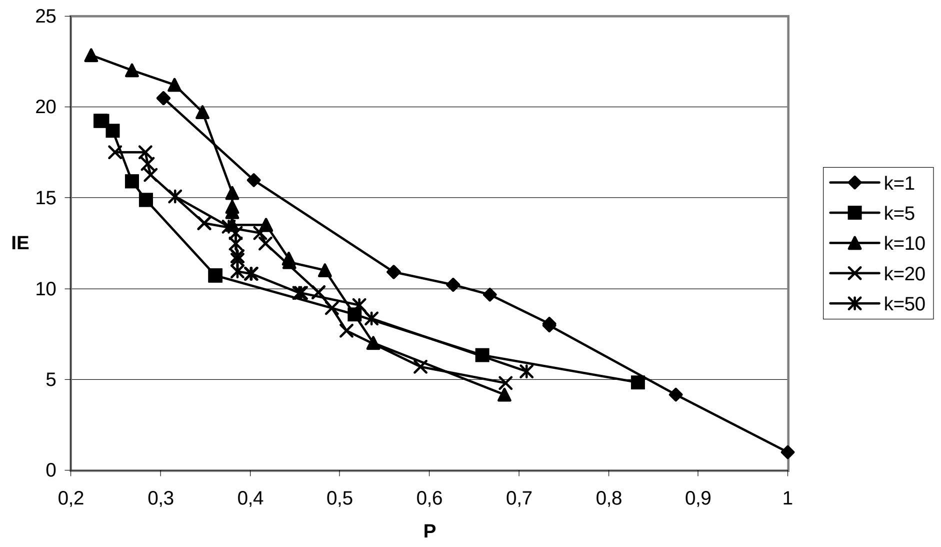


**Fig. 11.** Improvement in efficiency $IE$ as a function of the precision $P$ and of the relative error $\overline{\epsilon}$ for different values of $k$. The "approximation through the slowdown of distance improvements" was used for the CHV file

observed when approximation through distance distributions was used.

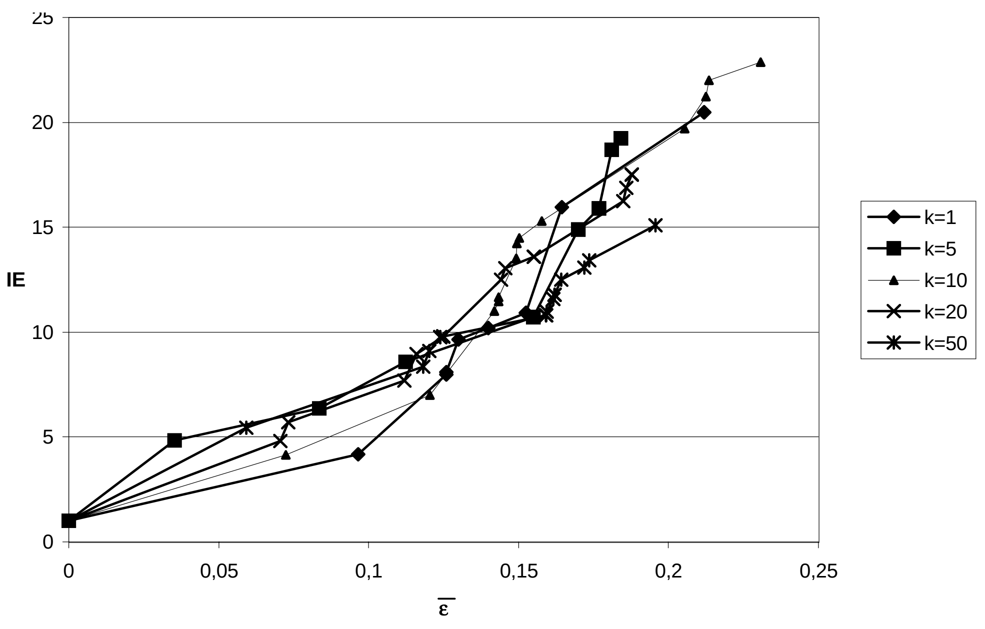
By definition, M-tree is not only able to deal with vector data of different dimensions, but it can also manage non-vector space data. Furthermore, since M-tree organizes search regions exclusively on distances, the actual number of dimension of vector spaces is not important from a performance point of view. What counts is the distribution of distances, and the worst case occurs when the variance is (very) low.

To demonstrate that M-trees are suitable for extreme data distributions as well, we have ran experiments with an Italian dictionary, using the edit distance as the metric. The maximum distance observed was 17, but practically all distances were in the range from 7 to 10. The results obtained for the method based on an approximate search through distance distributions demonstrate performance improvements from $IE$ = 100, when $\rho$ = 0.2, to $IE$ = 20, when $\rho$ = 0.02. These results are quite comparable with those reported in Fig. 4 for vector data.

To demonstrate that our approximation techniques scale well for growing files, we considered files from 1000 to 11000 objects. Figure 12 reports the results of experiments performed on the CHV file by using the "approximation through the slowdown of distance improvements" methods. When a file grows, we observed that efficiency increases for constant values of precision and of the relative error. This is a significant result which proves how useful our techniques are for large files.

## 7 Conclusions

Since multimedia content-based retrieval has to deal with multidimensional or distance-only data, similarity queries have become the most common type of queries in multimedia information systems. Though the idea of similarity



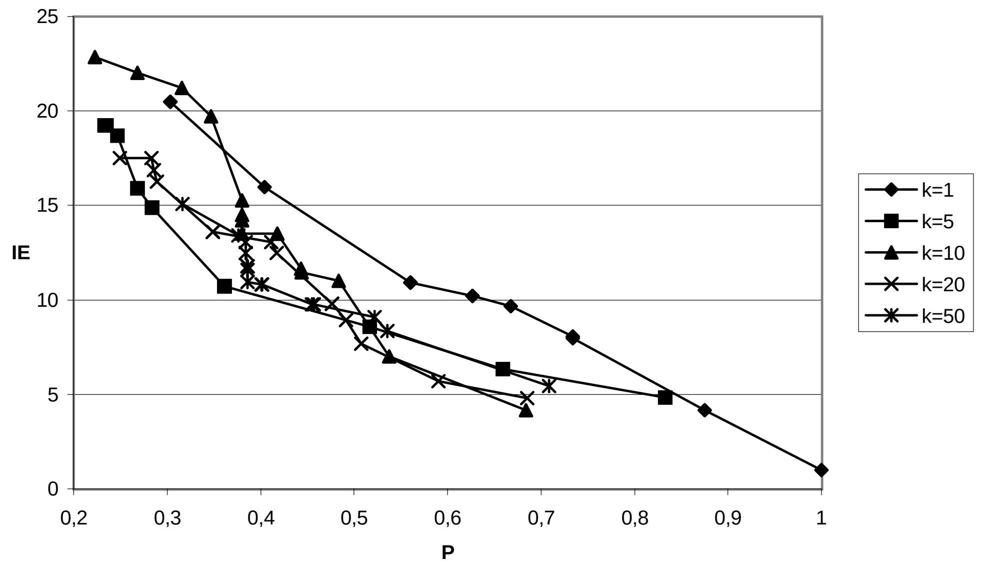


**Fig. 12.** Improvement in efficiency $IE$ as a function of the precision $P$ and of the relative error $\overline{\epsilon}$ for different file sizes $n$. The "approximation through the slowdown of distance improvements" was used for the CHV file

retrieval seems to be clear, it is quite difficult to implement efficiently, and the performance of similarity indexes remains a serious problem. However, since even the concept of similarity typically invokes a certain amount of *semantic* imprecision or vagueness in its definition, we have proposed to forsake some *syntactic* precision in exchange for improved efficiency.

To investigate this idea, we have specified three approximation techniques in the environment of the M-tree. The approximations of the proposed techniques are bound, respectively, by:

– the relative distance error, $\epsilon$;
– the statistically obtained fraction of the searched-file best cases, $\rho$;
– the tangent of the expected search improvement curve, $\kappa$.

We have also defined measures to quantify the performance improvements and the quality of efficiency, and tested all of our approximation techniques by querying on three different multidimensional data files.

Our results show that approximation through relative error is not very efficient. Approximation through distance distribution performs best; in fact, improvements in efficiency of even two orders of magnitude were observed for still quite high precisions. However, a characteristic distance distribution is needed, and the quality of this distribution may influence the performance. If such a distribution is not available, the approximation through the slowdown of distance improvements is strongly recommended, because this method is simple, it does not need additional data, and it also performs very well. Though the techniques of approximation seem to be more efficient for smaller sets of nearest neighbors, approximation techniques also seem to perform well for skewed distance distributions, and they scale well to manage large data files.

In this paper we have only considered the M-tree index. However, other multidimensional indexes, such as the R-tree and the X-tree, could clearly be modified in a similar way, because they use the same, or very similar, $k$-NN search algorithms. Furthermore, we believe that the proposed technology could also be used to process multiple similarity predicates by modifying the search algorithms suggested by Ciaccia et al. (1998a).

Encouraged by the results obtained, our future activity will concentrate on improving the performance of the existing methods and also on developing other approximation techniques. For example, provided more distance distributions in a specific file are available and the one which is most similar to the distribution with respect to the query is used, precision should improve. Another idea might be to consider *region proximity* as the constraint of approximation. We also plan to investigate approximation precision from a human user perspective by running experiments on features of images and then comparing the relevance of the exact and approximated sets. In this way, a user-acceptable precision $P$ could be determined and the approximation parameters correspondingly adjusted. Finally, though the nearest neighbor search seems to be the most important type of similarity query, we also plan to investigate the problem of approximate similarity range queries, which will certainly require new approaches.

*Acknowledgements.* Our work has been funded by the ESPRIT Long-Term Research Program, Project No. 9141, HERMES (Foundations of High-Performance Multimedia Information Management Systems). We would like to thank the members of the HERMES Consortium for fruitful discussions which inspired the idea of approximate similarity search. We would also like to thank the anonymous referees for their helpful comments on earlier drafts of this article.

## References

1. Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY (1994) An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. Journal of the ACM, to appear
2. Beckmann N, Kriegel H-P, Schneider R, Seeger B (1990) The $R^*$-tree: An Efficient and Robust Access Method for Points and Rectangles. In: Garcia-Molina H, Jagadish HV (eds) Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23–25, 1990. SIGMOD Record 19:2, 322–331
3. Berchtold S, Keim DA, Kriegel H.-P.: The X-tree: An Index Structure for High-Dimensional Data. In: Vijayaraman M, Buchmann AP, Mohan C, Sarda NL (eds) VLDB'96, Proceedings of 22nd International Conference on Very Large Data Bases, Mumbai (Bombay), India, September 3–6, 1996. Morgan Kaufmann, San Mateo, Calif., pp 28–39
4. Bozkaya T, Ozsoyoglu M (1997) Distance-Based Indexing for High-Dimensional Metric Spaces. In: Peckham J (ed) Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA, May 13–15, 1997. SIGMOD Rec 26(2): 357–368
5. Brin S (1995) Near Neighbor Search in Large Metric Spaces. In: Dayal U, Gray PMD, Nishio S (eds) VLDB'95, Proceedings of 21st International Conference on Very Large Data Bases, Zurich, Switzerland, September 11–15, 1995. Morgan Kaufmann, San Mateo, Calif., pp 574–584
6. Burden RL, Faires JD, Reynolds AC (1979) Numerical Analysis. Prindle, Weber & Schmidt, Boston, Mass.
7. Ciaccia P, Patella M.: Bulk Loading the M-tree. In: McDonald C (ed) ADC'98, Proceedings of the 9th Australian Database Conference, February 1998, Perth, Australia, pp 15–26
8. Ciaccia P, Patella M, Zezula P (1997) M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In: Jarke M, Carey MJ, Dittrich KR, Lochovsky FH, Loucopoulos P, Jeusfeld MA (eds) VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, Athens, Greece, August 25–29, 1997. Morgan Kaufmann, San Mateo, Calif., pp 426–435
9. Ciaccia P, Patella M, Zezula P (1998a) Processing Complex Similarity Queries with Distance-based Access Methods. In: Schek H-J, Saltor F, Ramos I, Alonso G (eds) Advances in Database Technology – EDBT'98, 6th International Conference on Extending Database Technology. (Lecture Notes in Computer Science, 1377), Valencia, Spain, March 23–27, 1998. Springer, Berlin Heidelberg New York, pp 9–23
10. Ciaccia P, Patella M, Zezula P (1998b) A Cost Model for Similarity Queries in Metric Spaces. In: Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Seattle, Washington, June 1–3, 1998. ACM Press, New York, pp. 59–68
11. Chiueh T (1994) Content-Based Image Indexing. In: Bocca JB, Jarke M, Zaniolo C (eds) VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, Santiago de Chile, Chile, September 12–15, 1994. Morgan Kaufmann, San Mateo, Calif., pp 528–593
12. Guttman A (1984) R-trees: A Dynamic Index Structure for Spatial Searching. Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data. SIGMOD Rec 14(2): 47–57
13. Hall PAV, Dowling GR (1980) Approximate String Matching. ACM Comput Surv 12(4): 381–402
14. Huttenlocker DP, Klanderman GA, Rucklidge WJ (1993) Comparing Images Using the Hausdorff Distance. IEEE Transactions on Pattern Analysis and Machine Intelligence 15(9): 850–863
15. Matousek J (1994) Geometric Range Searching. ACM Computing Surveys 26(4): 421–461
16. Seidl T, Kriegel H-P (1997) Efficient User-Adaptable Similarity Search in Large Multimedia Databases. In: Jarke M, Carey MJ, Dittrich KR, Lochovsky FH, Loucopoulos P, Jeusfeld MA (eds) VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, Athens, Greece, August 25–29, 1997. Morgan Kaufmann, San Mateo, Calif., pp 506–515
17. Stricker M, Orengo M (1995) Similarity of Color Images. In: Niblack W, Jain R (eds) SPIE Proc (Storage and Retrieval for Image and Video Databases III), San Diego, La Jolla, USA, 2420: 381–392
18. White DA, Jain R (1996) Similarity Indexing with the SS-tree. In: Su SYW (ed) Proceedings of the 12th International Conference on Data Engineering, New Orleans, Louisiana, March 1, 1996. IEEE CS Press, Piscataway, N.J., pp 516–523
19. Zezula P, Savino P, Rabitti F, Amato G, Ciaccia P (1998) Processing M-trees with Parallel Resources. In: Silberschatz A, Zhang A, Mehrotra S (eds) Proceedings of RIDE'98 – 8th International Workshop on Research Issues in Data Engineering: Continuous-Media Databases and Applications, Orlando, Florida, February 23–24, 1998. IEEE CS Press, Piscataway, N.J., pp 147–154