Fuzzy Decomposition of Spatially Extended Objects

Hans-Peter Kriegel, Martin Pfeifle
University of Munich, Germany
{*kriegel, pfeifle*}*@dbs.ifi.lmu.de*

Abstract

Modern database applications including computer-aided design, multimedia information systems, medical imaging, molecular biology, or geographical information systems impose new requirements on the effective and efficient management of spatial data. Particular problems arise from the need of high resolutions for large spatial objects. In this short paper, we sketch a new decompositioning approach based on clustering. We propose to describe a voxelized spatial object by a set of Gaussian distribution functions. Based on this decompositioning technique, we propose intersection queries which do not simply return a boolean value for each database object, but assign to each object a probability value indicating how likely an intersection is. The benefit of this approach compared to traditional approaches is that we do not any longer need an expensive refinement step for detecting whether objects intersect exactly on the fine-grained voxel sets.

1. Introduction

The efficient management of rasterized geographical objects has become an enabling technology for many novel database applications. As a common and successful approach, spatial objects can conservatively be approximated by a set of voxels, i.e. cells of a grid covering the complete data space (cf. Figure 1). By means of space filling curves, each voxel (often called pixel in 2D) can be encoded by a single integer and, thus, an extended object is represented by a set of enumerated voxels. As a principal design goal, space filling curves achieve good spatial clustering properties since cells in close spatial proximity are encoded by contiguous integers. Adjacent cell values can be grouped together to intervals, tiles or boxes which are basic datatypes for spatial applications.

By expressing spatial region queries as intersections of these spatial primitives, vital operations for GIS applications can be supported. For these applications suitable index structures, which guarantee efficient spatial query processing, are indispensable. An important new requirement for large spatial objects is a high approximation quality which is primarily influenced by the resolution of the grid covering the data space. A promising way to cope with high resolution spatial data may be found somewhere in between replicating and non-replicating spatial index structures. In the case of *replicating access methods*, e.g. the Relational Interval Tree [7], the number of the simple spatial primitives used to approximate the objects can become very high, resulting in a storage and query processing overhead. On the other hand, many of the *non-replicating access methods*, e.g. R-trees [5], use simple spatial primitives such as rectilinear hyper-rectangles for one-value approximations of extended objects. Although providing the minimal storage complexity, one-value approximations of spatially extended objects often are far too coarse. In many GIS applications, objects feature a very complex geometry. A non-replicating storage of such data causes region queries to produce too many false hits that have to be eliminated by subsequent filter steps. For such applications, the accuracy can be improved by decomposing the objects.

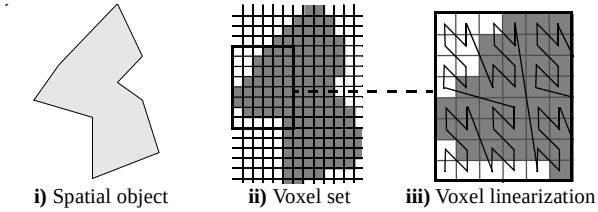

Figure 1. Voxelized Spatial Objects.

In this paper, we propose a new fuzzy decompositioning paradigm for high-resolution objects which is based on the well-known *k*-means clustering algorithm. The basic idea is to describe the voxel set by *k* clusters where the value of *k* depends on the characteristic of the voxel set. Each cluster is described by a few statistical values which are stored in a database. If we carry out collision or window queries, we determine for each object in the database a certain probability value that indicates the likelihood that the object belongs to the result set. This probability value can be computed by exploiting the statistical information describing the *k* clusters of the object and without accessing the fine-grained exact voxel representations. Note that the traditional approach also assigns probability values to the object, i.e. 0 if the object intersects the query object and 1 otherwise. As we omit the refinement step on the exact voxel representations, we can accelerate the complete query process.

The remainder of the paper is organized as follows. In Section 2, we present the related work in the area of spatial object decompositioning. In Section 3, we present our decompositioning approach based on clustering. In Section 4, we show how we can carry out fuzzy intersection queries based on decomposed spatial objects. Finally, we will close the paper in Section 5 with a short summary and a few remarks on future work.

2. Related Work

In this section, we will shortly present the related work in the area of decomposing high resolution voxelized objects.

Gaede pointed out that the number of voxels representing a spatially extended object exponentially depends on the granularity of the grid approximation [4]. Furthermore, the extensive analysis given in [2] shows that the number of voxels is proportional to the surface of the approximated object. Thus, in the case of large high resolution parts, the number of voxels can become unreasonably high.

A common approach to approximate the high resolution voxelized objects is to use their minimum bounding rectangles. Although providing the minimal storage complexity, one-value approximations of spatially extended objects often are far too coarse. In many applications, GIS or CAD objects feature a very complex and fine-grained geometry. The rectilinear bounding box of the brake line of a car, for example, would cover large parts of the data space. A non-replicating storage of such data would cause too many false hits in the *filter step* that have to be eliminated by the *refinement step*.

On top of the resolution of the data space and the clustering properties of the space-filling curve, a more fine-grained control of the trade-off between redundancy and accuracy is desired for many applications. Note that the granularity may have to differ for each individual object rather than to apply the same resolution to all objects. An approach to control this trade-off is the concept of size-bound and error-bound approximation [9] beyond the granularity-bound approximation [4]. A recursive subdivision procedure stops if the desired redundancy (size-bound) or the desired maximum approximation error (error-bound) is reached.

In [10], Kriegel and Schiwietz tackled the problem of "*complexity versus redundancy*" for 2D polygons. They investigated the natural trade-off between the complexity of the components and the redundancy, i.e. the number of components, with respect to its effect on efficient query processing. The presented empirically derived root-criterion suggests to decompose a polygon consisting of n vertices into $O(\sqrt{n})$ many simple approximations.

In [6] a high-resolution spatial object was decomposed based on its linearized voxel sequence (cf. Figure 1c) into gray intervals which cover both object voxel and non object voxel. The hull of the gray intervals was used in the filter step to generate a candidate set. In the refinement step the voxel set covered by a gray interval was evaluated to avoid false hits. The disadvantage of this approach is that the filter step has a rather bad selectivity because much dead space is covered by the gray intervals.

In this paper, we propose a totally new decompositioning approach of voxelized objects based on clustering. Thus, we decompose the spatial objects directly in the original 2D/3D space without linearizing the voxels before by means of space filling curves.

3. Clustering based Object Decompositioning

In the following, we assume that the geometry of a spatial object is described by a set of voxels which in 2D are also known as pixels. Throughout the remainder of this paper, we assume a 3D data space.

Definition 1 (Voxelized Objects)
Let O be the domain of all object identifiers and let $id \in O$ be an object identifier. Furthermore, let IN^3 be the domain of 3-dimensional points. Then, we call a pair $O_{voxel} = (id, \{v_1, ..., v_n\}) \in O \times 2^{N^3}$ a 3-dimensional voxelized object. We call each of the v_i an object voxel, where $i \in \{1, .., n\}$. By v_x, v_y, and v_z we describe the corresponding coordinates of a voxel v.

The idea of our decompositioning approach is to apply the rather simple and well-known *k-means* clustering algorithm [8] to our voxel set. The *k*-means algorithm can be regarded as a simplified version of the more general EM algorithm [1] which describes a dataset by multiple Gaussian distribution functions. In our approach, we regard each voxel as a 3-dimensional feature vector. The clustering algorithm *k*-means divides each voxelized object $o = (id, \{v_1, ..., v_n\})$ into a set of k clusters $C_1^o, .., C_k^o$. Each cluster C_i^o contains a voxel set V_i^o. For these k voxel sets the two following properties hold:

- $\forall i, j \in (1 \ldots k): (i \neq j \Rightarrow (v \in V_i^o \Rightarrow v \notin V_j^o))$
- $\bigcup_{i=1 \ldots k} V_i^o = \{v_1, \ldots, v_n\}$

Each cluster C_i^o is represented by a *centroid* $\overrightarrow{C_i^o}$.

$$\overrightarrow{C_i^o} = \frac{1}{|V_i^o|} \left(\sum_{v \in V_i^o} v_x, \sum_{v \in V_i^o} v_y, \sum_{v \in V_i^o} v_z \right)^t$$

Each voxel of the object is thereby assigned to the closest centroid. An iterative control strategy is used to minimize the squared distances of the voxels to the centroids:

$$sqerr^o = \sum_{i=1}^{k} \sum_{v \in V_i^o} L_2\left(v, \overrightarrow{C_i^o}\right)^2$$

The algorithm starts with a random partition and iteratively reassigns the voxels to the centroids based on the distances between the voxels and the centroids until a convergence criterion is met. For example, the iteration may stop when no voxels are reassigned from one centroid to another one any more, or when the squared error $sqerr^o$ of the clustering ceases to decrease significantly, or after a maximum number of iterations has been performed. Advantages of the *k*-means clustering algorithm are that it is easy to understand and to implement and that the runtime complexity is $O(n \cdot k \cdot l)$ for n voxels, k clusters and l iterations.

The accuracy of our decompositioning algorithm is influenced by the chosen parameters, i.e. the value of k and the initial centroids. In [3] a method based on sampling is introduced which helps to chose these parameters appropriately.

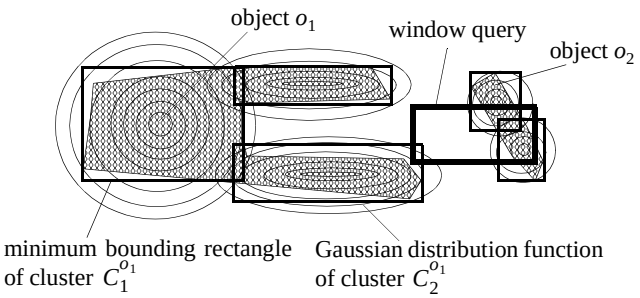

Figure 2. Gaussian distribution functions.

The distribution of the voxels within one cluster C_i^o can accurately be described by the centroid $\vec{C}_i^o = (c_{i,x}^o, c_{i,y}^o, c_{i,z}^o)$ and the standard deviations $\sigma_{i,x}^o$, $\sigma_{i,y}^o$, $\sigma_{i,z}^o$. These standard deviation values can be estimated by using the following formula[1].

$$\sigma_{i,x}^o = \sqrt{\frac{1}{|V_i^o| - 1} \sum_{v \in V_i^o} (v_x - c_{i,x}^o)^2}$$

Obviously, we could also describe the distribution of the voxels V_i^o by means of their covariance matrix Cov_i^o. In order to reduce the storage cost per cluster, we refrain from this more complex description and store only the centroids and the standard deviation values. Thus, we approximate a voxel set V_i^o of a cluster C_i^o by an axis-parallel Gaussian distribution function.

Figure 2 shows an example for describing two 2-dimensional voxelized objects o_1 and o_2 by means of 3 and 2 axis-parallel Gaussian distribution functions, respectively. Note that the problem of finding an appropriate value of k is part of active research in the data mining community. We can benefit from these results for solving the problem of finding the right trade-off between accuracy and redundancy in the case of spatial object decompositioning.

Figure 2, furthermore, shows the minimum bounding rectangles (MBR) of the clusters. These MBRs can be used throughout the filter step to detect true misses. Nevertheless, in the example in Figure 2, for both objects o_1 and o_2 a rather expensive refinement step on the exact voxel representation is necessary to decide whether the objects belong to the result set or not.

We propose to store the MBRs of an object in an R-tree [5] or one of its variants. In order to increase the efficiency of the refinement step, we do not store the exact voxel sets V_i^o of a cluster C_i^o. We only store the centroid values $c_{i,x}^o$, $c_{i,y}^o$, $c_{i,z}^o$ and the standard deviations $\sigma_{i,x}^o$, $\sigma_{i,y}^o$, $\sigma_{i,z}^o$ along with the MBRs of the cluster. Based on this statistical information, we can compute how likely an intersection between an object and the query object is without accessing the detailed information provided by the voxel set. In Figure 2, for instance, we will detect that the probability that o_1 is intersected by the window query is below 50%. On the other hand, based on the Gaussian distribution functions we can compute that it is very likely that o_2 belongs to the result set. In the following section, we will formally introduce this approach.

1. Likewise, we compute the values $\sigma_{i,y}^o$ and $\sigma_{i,z}^o$

4. Fuzzy Intersection Query

The idea to avoid the expensive refinement step, is to assign to each database object represented by an object dependent number of k clusters a probability value indicating the likelihood that the object belongs to the result set. Traditional intersection queries assign a value 1 or 0 to each object in the database indicating whether the object belongs to the result or not. We propose fuzzy intersection queries which order all database objects according to their intersection probability.

Definition 2 Fuzzy Intersection Query

Let DB be a database of voxelized decomposed objects, and let q be a query object. Furthermore, let $p(q \cap o)$ denote the probability that q intersects the object $o \in DB$. Then, the fuzzy intersection query $fuzzy_\cap$: $1..|DB| \rightarrow DB \times [0,1]$ is a function which ranks all objects $o \in DB$ according to their probabilities $p(q \cap o)$, i.e.

$$\forall i, j \in 1..|DB|:$$
$$i<j \wedge fuzzy_\cap(i)=(o_i, p(q \cap o_i)) \wedge fuzzy_\cap(j)=(o_j, p(q \cap o_j)) \Rightarrow$$
$$o_i \neq o_j \quad \wedge \quad p(q \cap o_i) \geq p(q \cap o_j)$$

Note that the result of such a fuzzy intersection query does not necessarily contain less information than the traditional result set. For instance, in Figure 2 the traditional approach only detects that object o_1 does not intersect the window query. Our approach might assign a probability value $p(q \cap o_1) = 0.28$ indicating that the object is quite close to the query object. We suggest that this probability value can be regarded as a meaningful measure for describing the closeness between database objects and query objects. Thus, the fuzzy intersection queries might even provide more information than the binary result *intersection* or *non-intersection.*

The crucial question is now how to compute the probability that an object belongs to a result set. In this short paper, we will exemplarily demonstrate how to compute this value for box volume queries (cf. Figure 2) which are commonly used in many applications, e.g. GIS or CAD applications.

First, we must compute for each cluster C_i^o of an object o the probability that at least one voxel of the corresponding voxel set V_i^o intersects the query. Thereto, we first determine for each dimension $\Delta \in \{x, y, z\}$ individually the probability $p_{i,\Delta}^o$ that the query interval $[l_\Delta, u_\Delta]$ intersects the one-dimensional Gaussian distribution with center $c_{i,\Delta}^o$ and standard deviation $\sigma_{i,\Delta}^o$ (cf. Figure 3). By means of the standard Gaussian cumulative distribution function, conventionally denoted by Φ, we can compute the probability $p_{i,\Delta}^o$ straightforward by the following equation:

$$p_{i,\Delta}^o = p(c_{i,\Delta}^o, \sigma_{i,\Delta}^o, l_\Delta, u_\Delta) = \Phi\left(\frac{u_\Delta - c_{i,\Delta}^o}{\sigma_{i,\Delta}^o}\right) - \Phi\left(\frac{l_\Delta - c_{i,\Delta}^o}{\sigma_{i,\Delta}^o}\right)$$

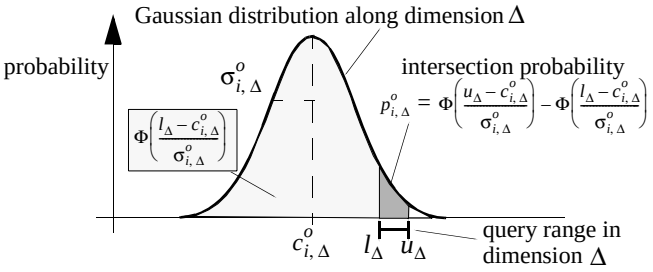

Figure 3. Intersection probability $p^o_{i,\Delta}$ of a query interval $[l_\Delta, u_\Delta]$ and a cluster C^o_i in dimension Δ.

For the voxel set belonging to the current cluster, we can assume that the three dimensions are independent from each other. Note that this assumption has to hold only for the object voxels of one cluster and not for all voxels of an object. Based on this legitimate assumption, we can easily compute the probability *p* that the box query intersects the axis-parallel 3-dimensional Gaussian distribution which describes the cluster *C*.

Lemma 1. Let $q = [l_x, u_x] \times [l_y, u_y] \times [l_z, u_z]$ be a 3-dimensional box query. Furthermore, let C^o_i be a cluster having $|V^o_i|$ many voxels which can be described by a 3-dimensional Gaussian distribution function with centroid $\vec{C}^o_i = (c_{i,x}, c_{i,y}, c_{i,z})$ and standard deviations $\sigma^o_{i,x}$, $\sigma^o_{i,y}$, $\sigma^o_{i,z}$. Then, we can compute the probability that at least one voxel of the cluster C^o_i is within the box query by:

$$p(q, C^o_i) = 1 - \left(1 - \prod_{\Delta \in \{x,y,z\}} p(c^o_{i,\Delta}, \sigma^o_{i,\Delta}, l_\Delta, u_\Delta)\right)^{|V^o_i|}$$

Finally, we can state the following lemma.

Lemma 2. Let $q = [l_x, u_x] \times [l_y, u_y] \times [l_z, u_z]$ be a 3-dimensional box query. Furthermore, let *o* be a voxelized object which is decomposed into *k* clusters $C^o_1, ..., C^o_k$. Then, we can compute the probability $p(q, o)$ that *q* intersects *o* by:

$$p(q, o) = 1 - \prod_{i=1}^{k} (1 - p(q, C^o_i))$$

Obviously, based on this probability value, we can answer the fuzzy intersection query introduced in Definition 2. We traverse the R-tree as usual. On the leaf level of the R-tree, we compute the probability values for each cluster (cf. Lemma 1) and combine these probability values to object probability values according to Lemma 2. Note that if we find one cluster which intersects our window query with high probability, the corresponding object probability value will be close to 1. On the other hand, to those objects for which we have not detected any clusters throughout the tree traversal, we assign a probability value of 0.

5. Conclusion

In this short paper, we sketched a new approach which helps to find an optimal trade-off between complexity and redundancy of object approximations. The proposed decompositioning algorithm is based on the well-known clustering algorithm *k*-means. Thus a voxelized object is described by *k* clusters. We describe each of these clusters by an axis-parallel 3-dimensional Gaussian distribution function and a minimum bounding rectangle of the cluster voxels. We store these minimum bounding rectangles along with statistical information in standard index structures. During query processing, we propose to omit the expensive refinement step on the exact voxel representations. Instead, we assign a probability value to each database object indicating how likely it belongs to the result set. The corresponding probability values are computed by exploiting statistical information describing the multivariate Gaussian distribution functions. In a give-me-more manner the user receives the objects which most likely belong to the result set.

Our first experiments showed that we can accelerate intersection queries considerably while still achieving high quality results. In our future work, we plan a detailed experimental evaluation demonstrating the characteristics and benefits of our fuzzy decompositioning approach.

References

[1] Dempster A.P., Laird N.M., and Rubin D.B.: *Maximum Likelihood from Incomplete Data via the EM algorithm.* Journal of the Royal Statistical Society, Series B, 39(1):1-31, 1977.

[2] Faloutsos C., Jagadish H. V., Manolopoulos Y.: *Analysis of the n-Dimensional Quadtree Decomposition for Arbitrary Hyperrectangles.* IEEE TKDE 9(3), 1997, 373-383.

[3] Fayyad U., Reina C., Bradley P.: *Initialization of Iterative Refinement Clustering Algorithms.* KDD 1998.

[4] Gaede V.: *Optimal Redundancy in Spatial Database Systems.* SSD 1995, pp. 96-116.

[5] Guttman A.: *R-trees: A Dynamic Index Structure for Spatial Searching*. SIGMOD1984, pp. 47-57.

[6] Kriegel H.-P., Pfeifle M., Pötke M., Seidl T.: *Spatial Query Processing for High Resolutions*. DASFAA 2003.

[7] Kriegel H.-P., Pötke M., Seidl T.: *Interval Sequences: An Object-Relational Approach to Manage Spatial and Temporal Data.* SSTD 2001, pp. 481-501.

[8] McQueen J.: *Some Methods for Classification and Analysis of Multivariate Observation.* Proc. 5th Berkeley Symp. on Math. Statist. and Prob., Vol. 1, 1965.

[9] Orenstein J. A.: *Spatial Query Processing in an Object-Oriented Database System*. SIGMOD 1986, pp. 326-336.

[10] Schiwietz M., Kriegel H.-P.: *Query Processing of Spatial Objects*: *Complexity versus Redundancy.* SSD 1993, pp. 377-396.