ELSEVIER

The Journal of Systems and Software 56 (2001) 165–182

The Journal of Systems and Software

www.elsevier.com/locate/jss

A fast content-based indexing and retrieval technique by the shape information in large image database ☆

Dong-Ho Lee *, Hyoung-Joo Kim

OOPSLA Laboratory, Department of Computer Engineering, Seoul National University, Shilim-Dong, Gwanak-Gu, Seoul 151-742, South Korea

Received 21 November 1999; received in revised form 29 February 2000; accepted 5 April 2000

Abstract

In this paper, we present an efficient content-based image retrieval (CBIR) system which employs the shape information of images to facilitate the retrieval process. For efficient feature extraction, we extract the shape feature of images automatically using edge detection and wavelet transform which is widely used in digital signal processing and image compression. To facilitate speedy retrieval, we also propose the Spherical Pyramid-Technique (SPY-TEC), a new indexing method for similarity search in high-dimensional data space. The SPY-TEC is based on a special space partitioning strategy, which is to divide the d-dimensional data space first into $2d$ spherical pyramids, and then cut the single spherical pyramid into several spherical slices. This partition provides a transformation of d-dimensional space into 1-dimensional space. Thus, we are able to use a B^+-tree to manage the transformed 1-dimensional data. We show that our image indexing method supports faster retrieval than other multi-dimensional indexing methods such as the R^*-tree through various experiments. Finally, we show the retrieval effectiveness of our CBIR system using the measure proposed by the QBIC (C. Faloutsos et al., 1994) system.

Keywords: CBIR system; Multimedia retrieval; Image indexing; High-dimensional index structure

1. Introduction

Recently, with the advances in various multimedia technologies such as high-speed network, compression, multimedia DBMS, visualization, the emphasis on multimedia applications is rapidly increasing. At the same time, many users of large image database are demanding more user-friendly access facilities on images by their content through queries such as "*Retrieve all images that are similar to this image in it's shape*", where *this image* refers to a hand-drawn sketch or an example image provided by users and *the shape* refers to it's content. The system providing such retrieval capabilities is called a content-based image retrieval(CBIR) system. In such a system, the content of database images are (semi-)automatically extracted and stored. A query image's content is extracted during run-time and used to match against those in the database (Ooi et al., 1998). The result of the query is a set of images that are similar to the query image. The content-based indexing and retrieval methods used in a CBIR system can be used as core and basic techniques in various multimedia information retrieval applications.

Most of the existing CBIR systems have two problems as follows. The first is that they have mainly focused on more accurate retrieval rather than speedy retrieval. The reason for this problem is that CBIR systems have been largely studied by researchers in pattern recognition and imaging community, whose focuses have been on extracting and understanding features of the image content, and on studying the retrieval effectiveness rather than on efficiency issues (speedy retrieval). However, speedy retrieval have been much more of an important issue because large image database has been the target of recent CBIR applications. The second is that they have mainly focused on the retrieval by the color information because it is analyzed easier and guarantees more accurate retrieval than using other information of images. However, if there are two images which describe half the body of a human and have different color compo-

☆ This work was supported by the Brain Korea 21 Project.

* Corresponding author. Tel.: +82-2-880-1830; fax: +82-2-882-0269.
E-mail addresses: dhlee@oopsla.snu.ac.kr
http://oopsla.snu.ac.kr (D.-H. Lee), hjk@oopsla.snu.ac.kr (H.-J. Kim).

0164-1212/01/$ - see front matter
PII: S0164-1212(00)00095-9

sitions, the system providing the retrieval by only the color may report that these two images are different from each other. But, in fact, they are similar in view of the shape.

To overcome these problems, we propose the System for Content-based imAge Retrieval using waveLET transform (SCARLET) which facilitate speedy retrieval by the shape trend of the entire image rather than the shape of an object in an image. The problem we mainly focus on is the design of a new indexing technique to facilitate speedy retrieval since SCARLET was developed for large image database. We extract the shape features from images automatically using edge detection and wavelet transform. Thus, they are represented as feature vectors in high-dimensional space. Therefore, we can use an existing index structure such as the R^*-tree for indexing feature vectors. However, recent research activities (Berchtold et al., 1996, 1997, 1998; White and Jain, 1996) reported the result that basically none of the querying and indexing techniques which provide good results on low-dimensional data also performs sufficiently well on high-dimensional data. This problem has been called ''curse of dimensionality''. To overcome this problem, we developed a new index structure, the Spherical Pyramid-Technique (SPY-TEC) which can support efficient similarity search in high-dimensional data space.

The SPY-TEC is based on the special space partitioning strategy, which divides the d-dimensional space into $2d$ spherical pyramids, and then cut the single spherical pyramid into several spherical slices. The results of this partition are shaped like the annual ring of a tree in 2-dimensional space. We can transform the given d-dimensional data space into a 1-dimensional value by this partitioning strategy. Thus, we can use a B^+-tree to store and access data items, and take advantage of all the benefits of a B^+-tree, such as fast insert, update and delete operations. We also propose the algorithms for processing hyperspherical range queries on the data space partitioned by this strategy. Finally, we show experimentally that the SPY-TEC clearly outperforms other index structures such as R^*-tree when processing hyperspherical range queries.

The rest of this paper is organized as follows. In Section 2, we review some related work. In Section 3, we explain our methods for the feature extraction and similarity measure. In Section 4, we present the SPY-TEC, especially focusing on its algorithms for processing hyperspherical range queries. Then, in Section 5, we describe the architecture of SCARLET and show the result of content-based queries in that system. In Section 6, we report on the performance study of the proposed schemes and finally, we conclude in Section 7.

2. Related work

This section reviews the past efforts in the areas of content-based image retrieval (CBIR) system and content-based image indexing.

Hirata and Kato (1992) proposed a image retrieval system which facilitates access on images by visual example. In their system, called ''query by visual example'', edge extraction is performed on user queries. These edges are matched against those of the database images in a fairly complex process that allows for corresponding edges to be shifted or deformed with respect to each other (Jacobs et al., 1995). They did not provide any content-based image indexing mechanism.

The QBIC system (Faloutsos et al., 1994), one of the most notable systems for querying by image content, had been developed at IBM. It allows a user to compose a query based on a variety of different visual properties such as color, shape, texture which are semi-automatically extracted from images. It partially used the R^*-tree as an image indexing method.

Jacobs et al. (1995) proposed an image match criteria and system which uses spatial information and visual features represented by dominant wavelet coefficients. Although their system provides for improved matching over image distance norms, it did not support any index structure. In fact, they mainly focused on efficient feature extraction by using wavelet transform rather than an index structure to support speedy retrieval.

The VisualSEEk (Smith and Chang, 1996) is a content-based image query system that enables querying by color regions and spatial layout. They devised an image similarity function which contains both color feature and spatial components. For image similarity matching, they use intrinsic part such as colors, region size and spatial location, and derived part such as relative spatial locations. However, the evaluations for derived parameter are very complex operations. They used spatial Quad-tree or R-tree for single region query and 2D-string to represent the spatial relationship in an image for multiple region query. These index structures were devised respectively for each query and were not integrated into a content-based indexing.

The VIPER (Ooi et al., 1998) is another image retrieval system that employs both the color and spatial information of images to facilitate the retrieval process. They first extract a set of dominant colors from image and then derive the spatial information of the regions bounded by these dominant colors. Thus, in their system, two images are similar in terms of color and spatial information if they have a few large clusters with the same color that fall in the same image

space. A valuable point of their work is that they proposed the SMAT as an index structure to facilitate speedy retrieval by the color-region information. The SMAT is a hierarchical index structure in which each layer consists of multidimensional index structure such as the R^*-tree. The SMAT is an integrated index structure to support the retrieval by both the color and the spatial information.

3. Shape-based image retrieval

In this section, we describe our approach to extract the shape feature of an image by using wavelet transform and explain similarity function used for image retrieval.

3.1. Extraction of the shape feature

Humans are sensitive to the shape trend of the entire image. If there are two images in which one describes the landscape of the mountain and the other represents half the body of a man, these two images are not similar although their color distributions are similar. However, if there are two images in which they describe half the body of a man, but have different color distributions, human recognize that two images are similar in view of the shape trend regardless of their color distributions. Therefore, it is significant to provide the retrieval by the shape information in CBIR systems.

Our method in extracting the shape feature is basically similar to that of (Jacobs et al., 1995). In (Jacobs et al., 1995), authors applied wavelet transform to a full color image having three channels (R,G,B) and then keep m wavelet coefficients with the large magnitude, respectively, for each channel. They used these wavelet coefficients to facilitate image retrieval. However, the number of wavelet coefficients used in retrieval processing is too large to use an existing index structure. In fact, they mainly focused on retrieval effectiveness rather than retrieval efficiency. Thus, they did not mention any index mechanism in this paper.

To facilitate image retrieval by only the shape, we need not to apply wavelet transform directly to full color images. Instead, as depicted in Fig. 1, we convert a full color image into a gray-scale image and extract edge information from a gray-scale image in lines 5 and 6 of Algorithm 1. And then, we apply discrete wavelet transform to this edge information in line 7. Finally, we can compose a feature vector by normalizing wavelet coefficients and keeping f coefficients with the largest magnitude in lines 8–10 of Algorithm 1. Since these feature vectors are generated by applying wavelet transform to edge information, they represent the shape trend of the entire image rather than the shape of an object within an image.

Algorithm 1 (*Feature extraction*).

1. *INPUT* : full color images
2. *OUTPUT* : f-dimensional feature vectors
3.
4. **for** all images in databases **do**
5. Convert a full color image to a gray-scale image
6. Extract edge information from a gray-scale image
7. Apply *Discrete Wavelet Transform* to edge information
8. Normalize wavelet coefficients so that each coefficient has a value between 0 and 1.
9. Keeping f coefficients with the largest magnitude
10. Compose a feature vector using truncated f coefficients
11. **end for**

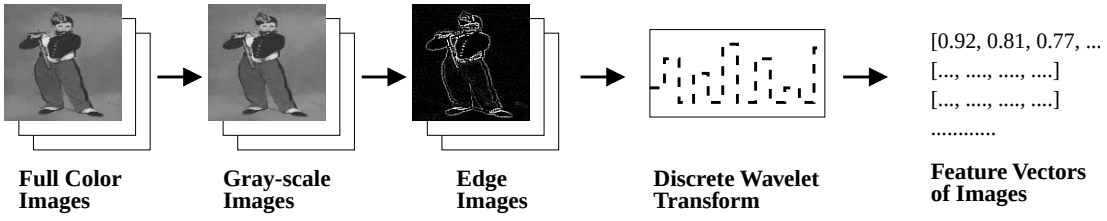


Fig. 1. Example of feature extraction.

In Algorithm 1, we use *Haar* wavelet transform because it is the fastest to compute and has been found to perform well in practice (Jacobs et al., 1995; Apostol et al., 1999). The representation through numerical vectors such as some coefficients of Fourier transform (Faloutsos et al., 1994) or wavelet transform (Mallat, 1989) is one of the most popular methods in shape representation (Korn et al., 1998). The feature vectors generated by Algorithm 1 are invariant to affine transform such as translation, scaling, and rotation (Graps, 1995; Chan, 1995; Stollnitz et al., 1995). Therefore, users can consistently retrieve image database by the shape.

3.2. Similarity measure

To measure the (dis-)similarity between feature vectors of two images, we use Euclidean distance which is defined as follows:

$$D(\vec{Q},\vec{T}) = \left(\sum_{i=0}^{f-1} |Q_i - T_i|^2\right)^{1/2}.$$

We have chosen the Euclidean distance, because it is preserved under orthonormal transforms such as wavelet transform. Thus, wavelet transform is often called distance preserving transform. This property is important in proving that there is no false dismissals for range queries in a CBIR system. In (Faloutsos et al., 1994; Faloutsos, 1995), authors showed that the distance of feature space always lower-bounds the actual distance of images in a system which employs orthonormal transforms for feature extraction and Euclidean distance for similarity function, such that there is no false dismissals for range queries.

Since we use orthonormal transforms for feature extraction and Euclidean distance for similarity function, we can prove that there is no false dismissal for range queries in our CBIR system. However, we would omit this proof because it is the same as that of (Faloutsos et al., 1994; Faloutsos, 1995). See either the Section 5 of (Faloutsos et al., 1994) or Lemma 3 of (Faloutsos, 1995) for details.

4. Content-based image indexing

The feature data produced by our feature extraction method is represented as a high-dimensional vector that can be mapped into a point in high-dimensional data space. Thus, we can use an existing index structure such as R^*-tree to support efficient retrieval. However, as depicted in Section 1, the performance of these index structures degrade rapidly as the dimension increases.

To overcome this drawback, the Pyramid-Technique (Berchtold et al., 1998) proposed a special partitioning strategy which divides the data space first into 2-dimensional pyramids and then, cuts the single pyramid into several partitions. They also proposed the algorithms for processing hypercubic range queries on the space partitioned by this strategy. Fig. 2 intuitively shows the space partitioning strategy of the Pyramid-Technique in a 2-dimensional example. In Fig. 2(a), the space has been divided into four triangles which all have the center of the data space as the top and one edge of the data space as the base. In Fig. 2(b), these four triangles are split again into several partitions, each corresponding to one data page of the B^+-tree. However, this axes-parallel partitioning strategy is not appropriate for similarity queries that are frequently used in CBIR systems because the shape of similarity queries is not a hypercube but rather a hypersphere in high-dimensional data space. Thus, when processing hyperspherical range queries with the Pyramid-Technique, there is a drawback which exists in all index structures based on the bounding rectangle. That is, they have to first perform the query using the minimum boundary rectangle (MBR) of hyperspherical range queries,

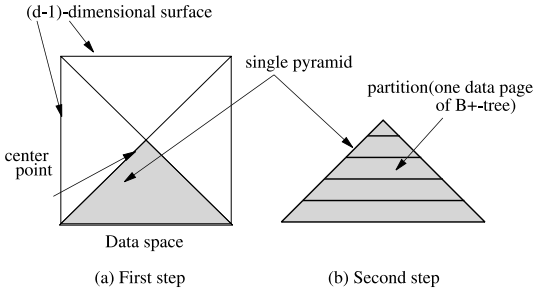


Fig. 2. Partitioning strategy of the Pyramid-Technique.

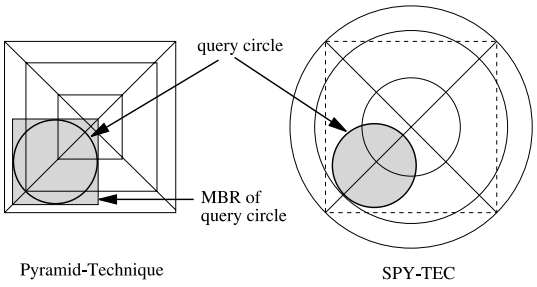


Fig. 3. Partitioning strategies.

and then perform post-processing which determines whether the object inside the MBR of the hyperspherical range query is actually lying in the hyperspherical region (Faloutsos, 1995). This processing needs unnecessary data page accesses and deteriorates the overall performance (Faloutsos et al., 1994; Faloutsos, 1995).

In this paper, we propose a new special space partitioning strategy and also propose the algorithms for processing hyperspherical range queries on the data space partitioned by this strategy.

Fig. 3 depicts the partitions resulting from an axes-parallel split of the Pyramid-Technique and a split of the SPY-TEC, which is shaped-like the annual ring of a tree, in a 2-dimensional example. Given a query circle as depicted in Fig. 3, the Pyramid-Technique accesses unnecessary data pages because it has to first perform the query using an MBR of the query circle. Thus, it has to access eight pages out of a total of 12 pages. However, the SPY-TEC can process the query with accessing only four pages. Thus, four page accesses are saved in the SPY-TEC.

4.1. The SPY-TEC

The main idea of the SPY-TEC is based on the observation that spherical splits will be better than axes-parallel splits of the Pyramid-Technique for similarity search. This observation is due to the fact that the shape of the queries used in similarity search is not a hypercube, but rather a hypersphere. The SPY-TEC is to transform the d-dimensional data points into 1-dimensional values and then store and access the values using the B^+-tree as the Pyramid-Technique does. Also, we store a d-dimensional point *plus* the corresponding 1-dimensional key as a record in the leaf nodes of the B^+-tree. Therefore, we do not need an inverse mechanism for this transformation. The transformation itself is based on a specific partitioning of the SPY-TEC. To define the transformation, we first explain the data space partitioning strategy of the SPY-TEC.

The SPY-TEC partitions the data space in two steps: In the first step, we split the d-dimensional data space into $2d$ spherical pyramids having the center point of the data space $(0.5, 0.5, \ldots, 0.5)$ as their top and a $(d-1)$-dimensional curved surface of the data space as their bases. In the second step, each of the $2d$ spherical pyramids is divided into several spherical slices and each slice corresponds to one data page of the B^+-tree. Fig. 4 shows the data space partitioning of the SPY-TEC in a 2-dimensional example. First, the 2-dimensional data space has been divided into four spherical pyramids resembling a fan. All of these spherical pyramids have the center point of the data space as their top and one curved line of the data space as their bases. In the second step, each of these four spherical pyramids is further split into several data pages which are shaped-like the annual ring of a tree. As a sphere of dimension d has $2d$ $(d-1)$-dimensional curved hyperplane as a surface, we obviously obtain $2d$ spherical pyramids such as in the case of the Pyramid-Technique.

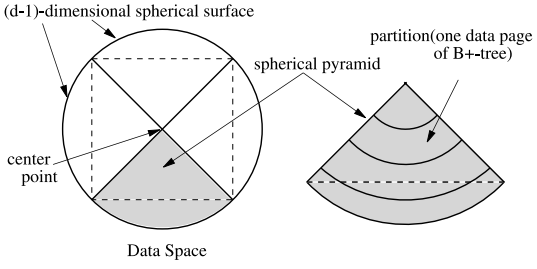


Fig. 4. Partitioning strategy of the SPY-TEC.

Numbering the spherical pyramids is the same as in the Pyramid-Technique. Given a point v, we have to find the dimension i having the maximum deviation $|0.5 - v_i|$ from the center to determine the spherical pyramid which the point v belongs to. If v_i is greater than or equal to 0.5, then the spherical pyramid which the point v belongs to is sp_{i+d}. If it is smaller than 0.5, the spherical pyramid which the point v belongs to is sp_i. Although the formal expression about this processing is the same as (Berchtold et al., 1998), we redefine it formally to help understand the partitioning strategy of the SPY-TEC.

Definition 1 (*Spherical pyramid of a point v*). A d-dimensional point v is defined to be located in spherical pyramid sp_i.

$$i = \begin{cases} j_{\max} & \text{if } v_{j_{\max}} < 0.5, \\ (j_{\max} + d) & \text{if } v_{j_{\max}} \geqslant 0.5, \end{cases}$$

$$j_{\max} = \{j | (\forall k, 0 \leqslant (j,k) < d,\ j \neq k : |0.5 - v_j| \geqslant |0.5 - v_k|)\}.$$

The left side of Fig. 5 shows the numbering of the spherical pyramids according to Definition 1. In order to transform d-dimensional data into a 1-dimensional value, we have to determine the location of a point v within its spherical pyramid. The Pyramid-Technique uses the height of the point within the pyramid as the location of the point. However, we use the distance from the point to the center point of the data space as the location of the point. The right side of Fig. 5 shows the process of determining the distance of the point v as the location within its spherical pyramid. We use the Euclidean distance to determine the distance of v, which complies with the definition of the similarity function in Section 3.2. More formally:

Definition 2 (*Distance of a point v*). Given a d-dimensional point v, the distance d_v of the point v is defined as

$$d_v = \sqrt{\sum_{i=0}^{d-1} (0.5 - v_i)^2}.$$

According to Definitions 1 and 2, we are able to transform a d-dimensional point v into a 1-dimensional value($i\lceil\sqrt{d}\rceil + d_v$). In this 1-dimensional value, i is the number of the spherical pyramid which the point v belongs to, d is the dimension of the point v, and d_v is the distance from the point v to the top of its spherical pyramid. More formally:

Definition 3 (*Spherical pyramid value of a point v*). Given a d-dimensional point v, let sp_i be the spherical pyramid which v belongs to according to Definition 1 and d_v be the distance of v according to Definition 2. Then, the spherical pyramid value spv_v of v is defined as

$$\mathrm{spv}_v = (i \cdot \lceil\sqrt{d}\rceil + d_v).$$

Note that i is an integer in the range $[0, 2d)$, d_v is a real number within the range $[0, 0.5\sqrt{d}]$ and $\lceil\sqrt{d}\rceil$ is the smallest integer not less than or equal to $\sqrt{d}$. Therefore, every point within a spherical pyramid sp_i has a value in the interval of $[i\lceil\sqrt{d}\rceil,\ (i\lceil\sqrt{d}\rceil + 0.5\sqrt{d})]$. In order to make the sets of spherical pyramid values covered by any two spherical pyramids sp_i and sp_j to be disjunct, we multiply i by $\lceil\sqrt{d}\rceil$. Note further that this transformation is not injective, i.e., two points v and v' may have the same pyramid value, but, as mentioned above, we do not need an inverse transformation

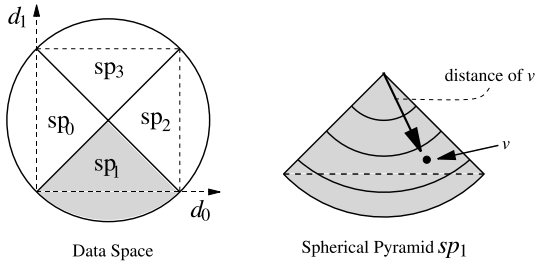


Fig. 5. Numbering of spherical pyramids and distance of a point.

as we store a d-dimensional point in addition to the corresponding 1-dimensional key as a record in the leaf nodes of the B^+-tree.

4.2. Index creation on the SPY-TEC

It is a very simple task to build an index using the SPY-TEC as for the Pyramid-Technique. Given a d-dimensional point v, we first determine the spherical pyramid value spv_v of the point and then insert the point into a B^+-tree using spv_v as a key. Finally, we store the point v and spv_v in the according data page of the B^+-tree. Update and delete operations can be done similarly.

4.3. Query processing on the SPY-TEC

There are two types of queries which are used in similarity search (White and Jain, 1996). One is the k-nearest neighbor query that returns the k most similar objects to the query object. The other is the hyperspherical range query that returns all objects within a threshold level of similarity to the query objects. In fact, the shape of these two queries is a hypersphere in high-dimensional spaces.

In this section, we propose the algorithms for processing hyperspherical range queries. The task of processing this query using the SPY-TEC is a very complex operation in contrast to the insert, delete and update operations.

For hyperspherical range queries, two parameters are given. The first parameter is a query point which indicates the query object and the second is a threshold level of similarity which indicates the limit of distance from a query point. The problem is defined as follows:

"*Given a query point and a threshold*(ε) *of similarity*

$$\text{query point}(Q) : [q_0, q_1, \ldots, q_{d-1}], \qquad \text{distance} : \varepsilon,$$

find the points in the database which are within the distance(ε) *from the query point*."

Note that the geometric correspondence of this similarity query is a hypersphere having the query point as the center point and ε as the radius of the sphere. To process this hyperspherical range query, we have to transform the d-dimensional query into 1-dimensional interval queries on the B^+-tree. However, as demonstrated by the simple 2-dimensional example depicted in Fig. 6(left), a query circle may intersect several spherical pyramids and the computation of the area of intersection is not trivial.

We process a hyperspherical range query in two main steps: In the first step, we have to determine which spherical pyramids are affected by the query, and then in the second step, we have to determine the ranges inside the spherical pyramids. The test to see whether a point is within the ranges or not is based on a single attribute criterion (d_v between two values). Therefore, determining all such objects is a 1-dimensional indexing problem, i.e., a B^+-tree indexing problem. Objects outside the ranges are guaranteed not to be contained inside the query circle. Points lying inside the ranges, are candidates for a further investigation. As depicted in Fig. 6(left), points lying between d_{low} and d_{high} are candidates. Some of the candidates are hits, others are false hits. Thus, in the refinement step, we have to investigate whether a point is inside the query circle or not. However, in order to investigate whether or not a point is inside the query circle, we have to calculate the distances from the query point to all of the points of the candidates. Through our experiments, we found that this task consumes a lot of CPU time. Thus, before starting the refinement step, we perform the filtering step, which tests whether or not the ith coordinate of a point is in the interval $[q_i - \varepsilon, q_i + \varepsilon]$. This

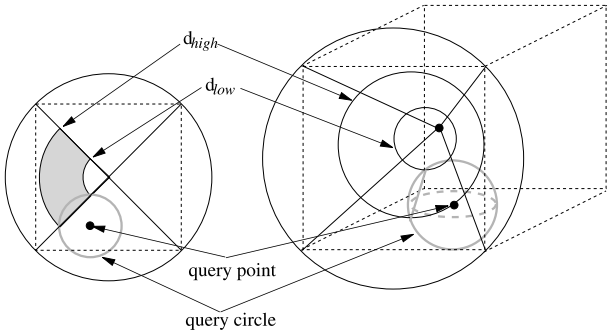


Fig. 6. Transformation of hypersphere range queries.

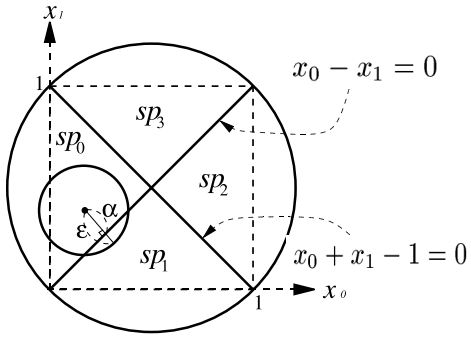


Fig. 7. Intersection of a spherical pyramid and a query circle.

is a simple comparison operation so that CPU time is almost unneeded. Therefore, we can eliminate a large amount of false hits before the refinement step and thus save a lot of CPU time.

For the sake of simplicity, we focus on the description of the algorithm only on spherical pyramids sp_i, where $i < d$. However, our algorithm can be extended to all spherical pyramids in a straightforward manner. Given a query point and an ε, we have to determine whether or not a spherical pyramid sp_i is affected by a given query area in the first step of our algorithm. We are able to solve this problem using a simple formula, which calculates the distance from a point to a hyperplane, used in geometry (Martin, 1996). As depicted in Fig. 7, if the distance (α) from the query point to the closest side plane of the adjacent spherical pyramid (sp_1) is shorter than ε, the query region and the adjacent spherical pyramid have been intersected. As the example of Fig. 7, the following Lemma 1 is to determine whether or not a spherical pyramid sp_i intersects the query circle using a simple formula.

Lemma 1 (Intersection of a spherical pyramid and a query circle). *Given a query point* ($Q = [q_0, q_1, \ldots, q_{d-1}]$) *and an* ε, *let* sp_j ($j < d$) *be a spherical pyramid to which a query point belongs and* sp_i *be a spherical pyramid which will be tested for an intersection. The intersection of a query region and a spherical pyramid* sp_i *is defined as*

Case 1: ($i = j$).

Case 2: ($|i - j| = d$). *Let* β *be the distance from the query point to the center of the data space.*

$\beta - \varepsilon \leqslant 0$.

Case 3: ($i < d$).

$$\frac{|q_j - q_i|}{\sqrt{2}} \leqslant \varepsilon.$$

Case 4: ($i \geqslant d$)

$$\frac{|q_j + q_i - 1|}{\sqrt{2}} \leqslant \varepsilon.$$

Proof. Given a point ($[q_0, q_1, \ldots, q_{d-1}]$) and a hyperplane ($k_0x_0 + k_1x_1 + \cdots + k_{d-1}x_{d-1} + C = 0$), the distance from the point to the hyperplane is defined as

$$\text{Distance} = \frac{|k_0q_0 + k_1q_1 + \ldots + k_{d-1}q_{d-1} + C|}{\sqrt{k_0^2 + k_1^2 + \ldots + k_{d-1}^2}}. \tag{1}$$

We are able to prove Cases 3 and 4 using this formula.

1. sp_i is a spherical pyramid to which a query point belongs because $i = j$. Therefore, the query region and spherical pyramid sp_i is intersected.

2. sp_i is a spherical pyramid in the opposite side of sp_j. Thus, if $\beta \leqslant \varepsilon$, then the center point of the data space is included in the query region. Therefore, sp_i is affected by the query region.

3. In formula (1), the index k_n and the constant C have discrete values $[-1, 0, 1]$ because of unit space. If $i < d$, an equation for the closest side plane of a spherical pyramid adjacent to the query point is $k_jx_j + k_ix_i = 0$ as depicted in the 2-dimensional example (Fig. 7). This formula can be extended to d-dimensional data space in a straightforward way. Given a d-dimensional space instead of the 2-dimensional space, the side plane of a spherical pyramid is not a

1-dimensional line, but a $(d-1)$-dimensional hyperplane, and the equation for this $(d-1)$-dimensional hyperplane has the common property that all indices except k_i and k_j are 0. In this case, $k_j = 1$ and $k_i = -1$ because $i < d$. Thus, the distance from the query point to the closest side plane of an adjacent spherical pyramid sp_i is $|q_j - q_i|/\sqrt{2}$. Therefore, if $|q_j - q_i|/\sqrt{2} \leqslant \varepsilon$, then a part of the query region and sp_i is intersected.

4. If $i \geqslant d$, an equation for the closest side plane of a spherical pyramid adjacent to the query point is $k_j x_j + k_i x_i - 1 = 0$ (refer to Fig. 7). In this case, $k_j = 1$ and $k_i = 1$ because $i \geqslant d$. Thus, the distance from the query point to the closest side plane of adjacent spherical pyramid sp_i is $|q_j + q_i - 1|/\sqrt{2}$. Therefore, if $|q_j + q_i - 1|/\sqrt{2} \leqslant \varepsilon$, sp_i is affected by the query region. □

In the second step, we have to determine which spherical pyramid values inside an affected spherical pyramid sp_i are affected by the query. Thus, we should find an interval $[d_{\mathrm{low}}, d_{\mathrm{high}}]$ in the range of $[0, 0.5\sqrt{d}]$ so that the spherical pyramid values of all points inside the intersection of the query circle and spherical pyramid sp_i are in the interval $[i\lceil\sqrt{d}\rceil + d_{\mathrm{low}}, i\lceil\sqrt{d}\rceil + d_{\mathrm{high}}]$. We are able to determine the values of d_{low} and d_{high} through the *Pythagoras Theorem* (Martin, 1996).

For a case in which the center point of the data space is in the query region, the value d_{low} is always 0. Thus, we have only to determine the value d_{high}. In Fig. 8(I)(a), d_{high} can be determined by using the distance from the center point of the data space to the query point and ε as a radius of query circle. If we let β be the distance from the center to the query point, then d_{high} is $(\beta + \varepsilon)$. Also, in Fig. 8(I)(b) or (d), if we let α be the distance from the query point to the closest side plane of an adjacent spherical pyramid, d_{high} can be determined by using the *Pythagoras Theorem*. The length (δ) of the base line in a right-angled triangle which consists of two sides, α and β, is $\sqrt{\beta^2 - \alpha^2}$, and the length (γ) of the base line in a right-angled triangle which consists of two sides, α and ε is $\sqrt{\varepsilon^2 - \alpha^2}$. In these cases, d_{high} is $(\gamma + \delta)$. Finally, in Fig. 8(I)(c), the spherical pyramid sp_i is in the opposite side of the spherical pyramid sp_j which the query point belongs to. Thus, d_{high} is $(\gamma - \delta)$, but in this case, note that α is the maximum value of the distances from the query point to the closest side plane of all adjacent spherical pyramids.

For a case in which the center point of the data space is not inside the query region, we are able to determine the values of d_{low} and d_{high} analogously. As depicted in Fig. 8(II)(a), in a spherical pyramid which the query point belongs to, d_{low} and d_{high} are determined as follows: $d_{\mathrm{low}} = \beta - \varepsilon,\ d_{\mathrm{high}} = \beta + \varepsilon$. And in a spherical pyramid which the query point does not belong to (Fig. 8(II)(b)), d_{low} and d_{high} are determined as follows: $d_{\mathrm{low}} = \delta - \gamma,\ d_{\mathrm{high}} = \delta + \gamma$. As depicted in an example of Fig. 8, the following Lemma 2 is to determine the interval $[d_{\mathrm{low}}, d_{\mathrm{high}}]$, in which the query circle intersects the spherical pyramids.

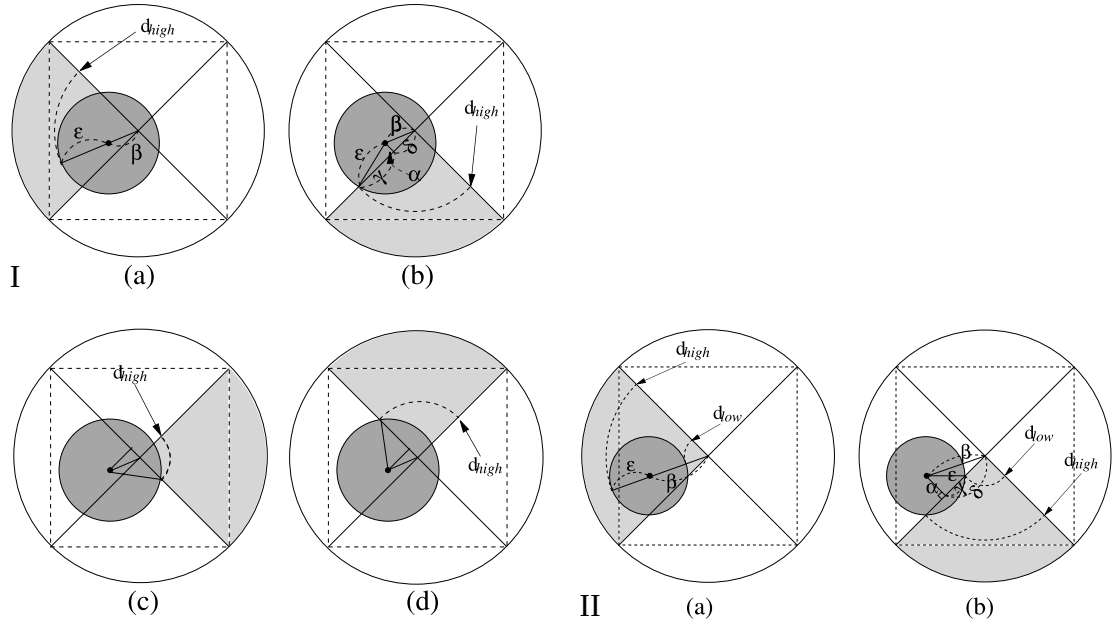


Fig. 8. Interval of intersection of query and spherical pyramid. (I) The center point is inside the query region. (II) The center point is not inside the query region.

Lemma 2 (Interval of intersection of query and spherical pyramid). *Given a spherical pyramid sp_j which the query point belongs to, and a spherical pyramid sp_i which is affected by the query region, the intersection interval $[d_{low}, d_{high}]$ is defined as follows*:

Case 1: *The case in which the center point of the data space is inside the query region.*

Case 1.1: $(j = i)$. *Let β be the distance from the center point to the query point and ε be the radius of the query circle.*

$$d_{low} = 0, \quad d_{high} = \beta + \varepsilon.$$

Case 1.2: $(|j - i| = d)$. *Let α be the maximum value of the distances from the query point to the closest side plane of all adjacent spherical pyramids and δ be the length of the base line in a right-angled triangle which consists of two sides, α and β. Also, let γ be the length of the base line in a right-angled triangle which consists of two sides, α and ε.*

$$d_{low} = 0, \quad d_{high} = \gamma - \delta = \sqrt{\varepsilon^2 - \alpha^2} - \sqrt{\beta^2 - \alpha^2}.$$

Case 1.3: (*Otherwise*). *Let α be the distance from the query point to the closest side plane of an adjacent spherical pyramid and δ, γ be the same as in Case* 1.2

$$d_{low} = 0, \quad d_{high} = \gamma + \delta = \sqrt{\varepsilon^2 - \alpha^2} + \sqrt{\beta^2 - \alpha^2}.$$

Case 2: *The case in which the center point of the data space is not inside the query region.*

Case 2.1: $(j = i)$.

$$d_{low} = \beta - \varepsilon, \quad d_{high} = \beta + \varepsilon.$$

Case 2.2: $(j \neq i)$.

$$d_{low} = \delta - \gamma, \quad d_{high} = \delta + \gamma.$$

Proof. We have to show for any point v which is located inside the query circle and an affected spherical pyramid sp_i that d_v is lying in the resulting query interval $[d_{low}, d_{high}]$. Therefore, we have to prove that

$$d_{low} \leqslant d_v \leqslant d_{high}$$

1. $d_{low} \leqslant d_v$: This holds because the center of the data space is inside the query region so that $d_{low} = 0 \leqslant d_v$.

1.1. $d_v \leqslant d_{high}$: This case corresponds to Fig. 8(I)(a). Let v be the point which has the maximum of the distances of points lying inside the query region and an affected spherical pyramid sp_i. Then, d_v is less than or equal to $(\beta + \varepsilon)$. Therefore, $d_v \leqslant d_{high} = \beta + \varepsilon$.

1.2. $d_v \leqslant d_{high}$: In this case, a spherical pyramid sp_i is in the opposite side of a spherical pyramid sp_j so that there is no side plane of a spherical pyramid adjacent to the query point (refer to Fig. 8(I)(c)). By the *Pythagoras Theorem*, $\delta = \sqrt{\beta^2 - \alpha^2}$ and $\gamma = \sqrt{\varepsilon^2 - \alpha^2}$. Let v be the point which has the maximum of the distances of points lying inside the query region and an affected spherical pyramid sp_i. Then, d_v is less than or equal to $(\gamma - \delta)$. Therefore, $d_v \leqslant d_{high} = \gamma - \delta$.

1.3. In this case, there is the closest side plane of a spherical pyramid adjacent to the query point(refer to Fig. 8(I)(b) and (d)). Let v be the point which has the maximum of the distances of points lying inside the query region and an affected spherical pyramid sp_i. Then, d_v is less than or equal to $(\gamma + \delta)$. Therefore, $d_v \leqslant d_{high} = (\gamma + \delta)$.

2. $d_{low} \leqslant d_v \leqslant d_{high}$: In this case, d_{low} is not 0 and a spherical pyramid in the opposite side of the spherical pyramid which the query point belongs to, is not affected by the query region. Thus, we have to show that $d_{low} \leqslant d_v \leqslant d_{high}$.

2.1. In this case, sp_i is the spherical pyramid which the query point belongs to(refer to Fig. 8(II)(a)). Let v be the point which has the minimum of the distances of points lying inside the query region and an affected spherical pyramid sp_i. Then, d_v is greater than or equal to $(\beta - \epsilon)$, and if v is the point which has the maximum of the distances of points lying inside the query region and an affected spherical pyramid sp_i, then, d_v is less than or equal to $(\beta + \epsilon)$. Therefore, $(\beta - \epsilon) = d_{low} \leqslant d_v \leqslant d_{high} = (\beta + \epsilon)$.

2.2. δ, γ are the same as in Case 1.3. This case corresponds to Fig. 8(II)(b). Let v be the point which has the minimum of the distances of points lying inside the query region and an affected spherical pyramid sp_i. Then, d_v is greater than or equal to $(\delta - \gamma)$, and if v is the point which has the maximum of the distances of points lying inside the query region and an affected spherical pyramid sp_i, then, d_v is less than or equal to $(\delta + \gamma)$. Therefore, $(\delta - \gamma) = d_{low} \leqslant d_v \leqslant d_{high} = (\delta + \gamma)$. □

Algorithm 2 (*Processing the hyperspherical range query*).

1. *INPUT*: qp ← query point, $\varepsilon \leftarrow$ range
2. *OUTPUT*: set of points satisfying the query (*result*)
3.
4. **for** $i = 0$ **to** $2d - 1$ **do**
5. **if** intersect(sp[i], qp, ε)
6. /* *Using* Lemma 1, *we test that a spherical pyramid* sp_i *is intersected by a query sphere having* qp *as the query point and* ε *as the radius of the sphere* */
7. determine_interval(sp[i], qp, ε, d_{low}, d_{high});
8. /* *Using* Lemma 2, *we determine which spherical pyramid values inside an affected spherical pyramid* sp_i *are affected by the query sphere. Namely, we find an interval* $[d_{\mathrm{low}}, d_{\mathrm{high}}]$ */
cs = btree_interval_query ($i\lceil\sqrt{d}\rceil + d_{\mathrm{low}}$, $i\lceil\sqrt{d}\rceil + d_{\mathrm{high}}$);
9. /* *perform* 1*-dimensional interval query on the* B^+*-tree* */
10. **for** c = cs.first **to** cs.end **do**
11. **if** inside_DiameterOfCircle(c, qp, ε) **then**
12. /* *filtering step* */
13. **if** point_in_circle(c, *qp*, ε) **then**
14. /* *refinement step* */
15. *result* ← c;
16. **end if**
17. **end if**
18. **end for**
19. **end for**
20. **end for**

With Lemmas 1 and 2, we can process the hyperspherical range query by Algorithm 2.

For example, consider the query point and the ε in Fig. 8(I). For a spherical pyramid sp_0, we can know that sp_0 intersects the query circle in line 5 of Algorithm 2. In line 7 of Algorithm 2, we can also determine the interval $[d_{\mathrm{low}}, d_{\mathrm{high}}]$ in which the query circle intersects sp_0. And, then we perform 1-dimensional interval query on the B^+-tree using the values of d_{low} and d_{high} in line 9. After performing 1-dimensional interval query on the B^+-tree, we can obtain the points whose d_v's are lying inside the interval $[d_{\mathrm{low}}, d_{\mathrm{high}}]$. For these points, as mentioned above, we determine whether they are really inside the query circle in the **for** loop of lines 11–19. For other spherical pyramids such as sp_1, sp_2 and sp_3, we perform the same process as that of sp_0.

5. Design and implementation of SCARLET

In this section, we describe the design and implementation of SCARLET, a prototype image retrieval system which employs feature extraction method described in Section 3 and the SPY-TEC as an indexing method in Section 4.

SCARLET is implemented on the UNIX system using the X/Motif and C++ programming language. Fig. 9 shows the overall architecture of SCARLET which is operated in two main session as follows.

In the DB population session, feature vectors for the shape information are extracted from images through the feature extraction module, and then inserted into the SPY-TEC structure with their OID's. During this session, original images are stored in image database. In the DB query session, users can sketch a query image or select an example image through the graphical user interface module. And, then the feature vector of a query image is extracted through the feature extraction module, and it is matched against feature vectors stored in the SPY-TEC structure with a constant threshed(ε) for similarity. After this processing, feature vectors and their OID's similar to that for the query image are taken, and they rank in decreasing order of similarity by using formula (2) in the ranking module. Finally, by using the OID's produced in the ranking module, the images similar to the query image are loaded from image database, and showed on the graphical user interface. Formula (2) represents the similarity of images as a percentage that helps users understand the similarity more easily.

$$\mathrm{Similarity}(Q,T) = 100 \times \left(\frac{\varepsilon - D(\vec{Q},\vec{T})}{\varepsilon}\right) \quad (\%). \tag{2}$$

SCARLET supports two visual queries for image retrieval. One is a query by a hand-drawn sketch and the other is a query by an example image selected by users.

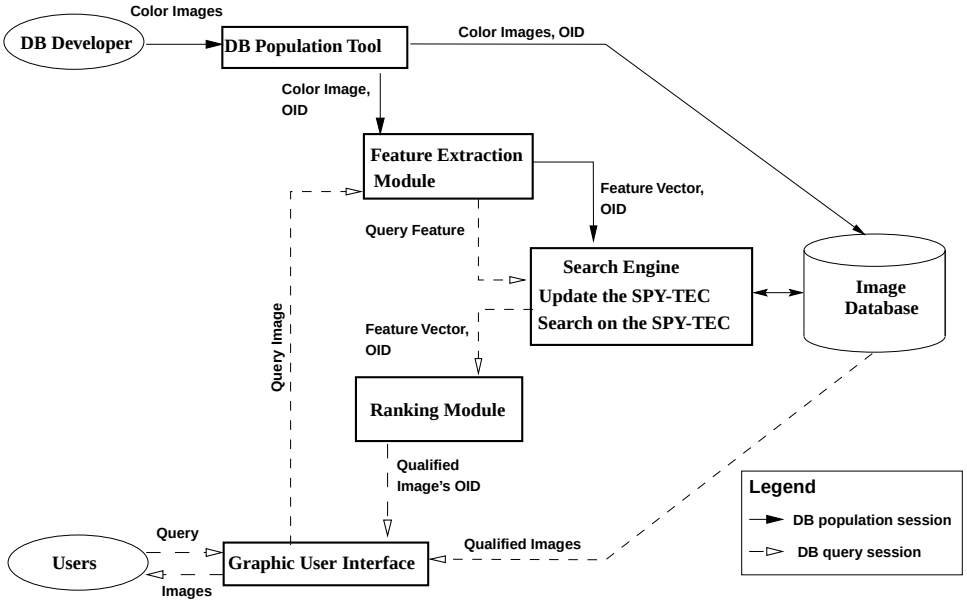


Fig. 9. The architecture of SCARLET.

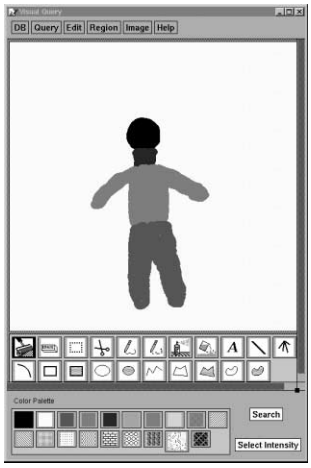


Fig. 10. Query by user-sketch.

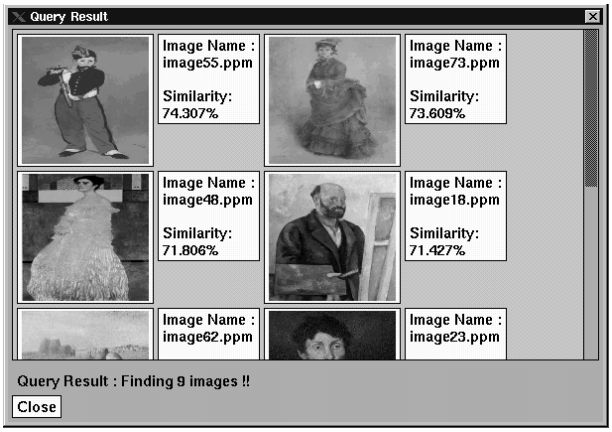


Fig. 11. Retrieval results of Fig. 10.

Figs. 10 and 11 shows the visual query by a hand-drawn sketch of two visual queries and its results. As depicted in Fig. 10, users can draw an image similar to what they want to retrieve through the graphical user interface. And if they click a button "Search", the search processing begins. Fig. 11 is the result of this searching. The top-left of Fig. 11 is the most similar image to the query image, and the top-right is the second most similar. From Figs. 10 and 11, we can observe that SCARLET retrieves images similar to the query in the shape trend.

The query by an example image is similar to that by a hand-drawn sketch, but just different in that users select a particular image as a query in example images supported by SCARLET. By using these two complementary queries, users can retrieve images more easily.

6. Performance evaluation

There are two parameters for measuring the performance of a CBIR system (Faloutsos et al., 1994; Ooi et al., 1998). One is the retrieval efficiency which focuses on the speed of the retrieval. The other is the retrieval effectiveness which focuses on the accuracy of the retrieval. For speedy retrieval, we proposed a new indexing technique called the SPY-TEC. We also proposed a new method for extracting feature vectors from images to improve the accuracy of the retrieval.

In this section, we show the retrieval efficiency and the retrieval effectiveness of SCARLET through various experiments. To demonstrate the impact of the retrieval efficiency, we compared the SPY-TEC to other related index structures. And, in order to evaluate the retrieval effectiveness of SCARLET, we selected five query images and measured how close to the top of the list of retrieved images the set of relevant images appears.

6.1. Retrieval efficiency

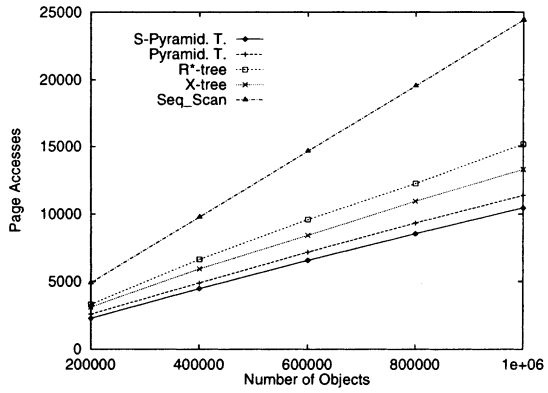
To show the practical impact of the SPY-TEC, we compared it to other index structures such as the Pyramid-Technique (Berchtold et al., 1998), the R^*-tree (Beckmann et al., 1990), the X-tree (Berchtold et al., 1996), and the sequential scan.

For clear comparison, we assume that all relevant information is stored in the various indexes, as well as in the file used for the sequential scan. Therefore, no additional accesses to fetch objects for presentation or further processing are needed in any of the techniques applied in our experiments. Our experiments have been performed on SUN Sparc 20 workstations with 8 GigaBytes of secondary storage. We performed our experiments using both real and synthetic data sets and hyperspherical range queries with a constant radius (ε) in all experiments. The query points were selected randomly from the data space so that the distribution of the queries equals the distribution of the data set itself. Thus, in the case of uniform data distribution, we used 100 query points that are uniformly distributed. The synthetic data set contains 200,000 $\sim$ 1,000,000 uniformly distributed points in 8 $\sim$ 24-dimensional data spaces.

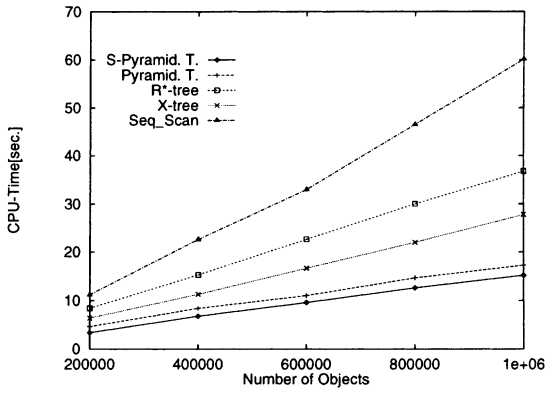
In our first experiment, we measured the performance behavior while we varied the number of objects. We performed hyperspherical range queries with 100 query points and a constant radius for each database size and measured the average result set size, the number of page accesses, CPU time and finally the total elapsed time needed for processing hyperspherical range queries. The page size is 4096 Bytes and the effective page capacity is 41.0 objects per page in all index structures.

Fig. 12 shows the results of our first experiment. Fig. 12(d) shows the radii of query circles used in this experiment and their average result set size. The speed-up with respect to the number of page accesses seems to be almost constant and ranges between 1.09 and 1.13 over the Pyramid-Technique, between 1.44 and 1.49 over the R^*-tree, between 1.27 and 1.36 over the X-tree, and between 2.14 and 2.33 over the sequential scan. The speed-up in CPU time is higher than the speed-up in page accesses, but is only increasing with growing database sizes. Finally, most important is the speed-up in total elapsed time. It is higher than the speed-up in the number of page accesses and CPU time so that reaches its highest value with the largest database. The SPY-TEC with one million objects performs hyperspherical range queries 1.14 times faster than the Pyramid-Technique, 5.81 times faster than R^*-tree, 4.52 times faster than the X-tree, and 3.65 times faster than the sequential scan. Range query processing on B^+-tree can be performed much more efficient than on index structures based on the R^*-tree because large parts of the tree can be traversed efficiently by following the side links in the data pages, and long-distance seek operations including expensive disk head movements have a lower probability due to better disk clustering possibilities (Berchtold et al., 1998). Specially, when processing hyperspherical range query, the SPY-TEC is more efficient than the Pyramid-Technique because its spherical split strategy is more suitable for the shape of queries than the right-angled split strategy of the Pyramid-Technique.

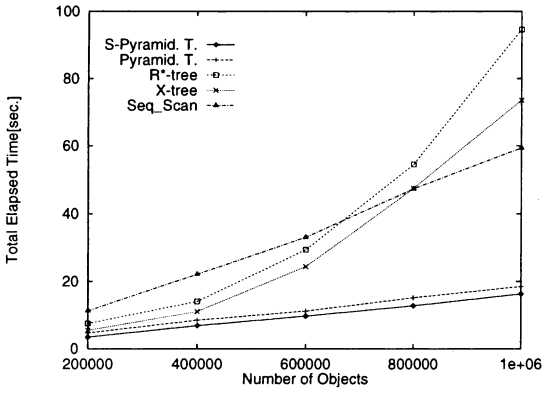
In our second experiment, we measured the performance of query processing while we varied data space dimension. For this experiment, we created five files with the dimensionalities 8, 12, 16, 20 and 24. The database size fixed at



(a) Page Accesses



(b) CPU Time(sec.)



(c) Total Elapsed Time(sec.)

DB Size	Radius(ε)	Average Result Set Size
200,000	0.70	10.92
400,000	0.69	15.67
600,000	0.68	21.34
800,000	0.67	22.26
1,000,000	0.66	22.29

(d) Average Result Set Size

Fig. 12. Performance behavior over database size.

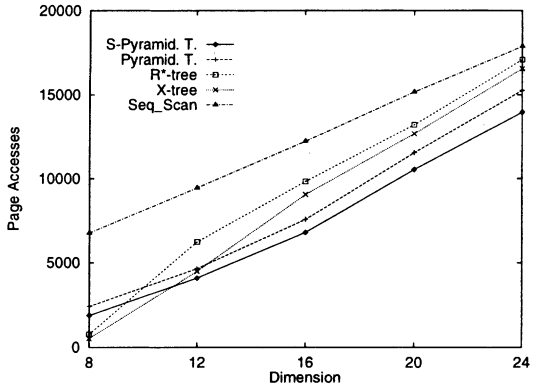
500,000 objects. The page size is 4096 Bytes as in the first experiment and the effective data page capacity depends on the dimension and ranges from 28 to 74 objects per page.

Fig. 13 shows the results of our second experiment. We observed that the R^*-tree is more efficient than our technique including the Pyramid-Technique and sequential scan in 8-dimensional data spaces, but rapidly decreases with increasing dimension up to the 12-dimensional data space. From this point, the page accesses are growing linearly with the index size. In the case of the X-tree, a phenomenon analogous to that of the R^*-tree has appeared except that its efficiency rapidly decreases in 16-dimensional data space. However, the SPY-TEC and Pyramid-Technique have no rapid deterioration of the performance even though their performance decreases slowly with increasing dimension. We also observed that the SPY-TEC clearly outperforms the Pyramid-Technique and the sequential scan in all cases. The speed-up in CPU time is analogous to the speed-up in page accesses, but is higher than it. Finally, in total elapsed time, the SPY-TEC performs hyperspherical range queries 1.25 times faster than the Pyramid-Technique, 3.21 times faster than the R^*-tree, 2.71 times faster than the X-tree, and 1.37 times faster than the sequential scan when the dimension is 24.

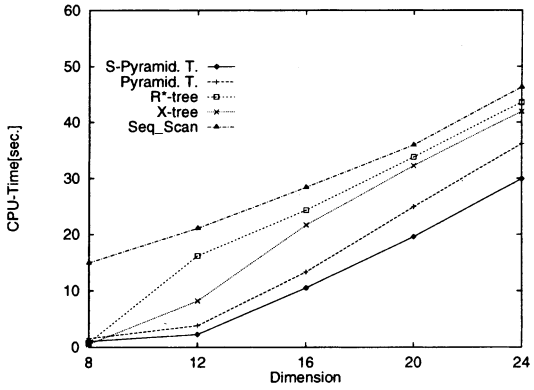
Through our second experiment, we observed that the performance of the R^*-tree or the X-tree is efficient when the dimension is lower than 10, but rapidly decreases with going to higher dimension so that the sequential scan yields better performance than them. We also found out that our new method outperforms the R^*-tree or the X-tree when the dimension is higher than 10, and outperforms the Pyramid-Technique and the sequential scan in all cases.

In the last experiment of this series, we measured the performance of the SPY-TEC on real data sets. Our real data sets is feature vectors extracted from 56,230 images by using our feature extraction method, and the dimension of each feature vector is 16. We varied the radii of the query circles from 0.1 to 0.5 and measured the average result set size, the number of page access, CPU time, and the total elapsed time absorbed to process hyperspherical range queries.

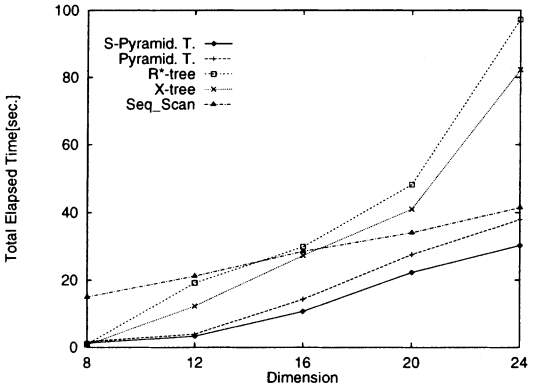
We found out that the real data sets is more clustered than the uniform data distribution through Fig. 14(d). As depicted in Fig. 14, the SPY-TEC clearly outperforms the other index structures for any radius of query circles used in



(a) Page Accesses



(b) CPU Time(sec.)



(c) Total Elapsed Time(sec.)

Dimension	Radius(ε)	Average Result Set Size
8	0.3	55.92
12	0.5	30.37
16	0.7	25.09
20	0.9	23.26
24	0.98	3.47

(d) Average Result Set Size

Fig. 13. Performance behavior over data space dimension.

this experiment. The sequential scan take about 3.2 s to process queries regardless of the radius of query circles. This result is deserved because the sequential scan has to access all pages and all objects stored in database regardless of the radius. We found out that index-based query processing such as the R^*-tree and X-tree outperforms the sequential scan on real data sets. When the radius of query circle is 0.1, the SPY-TEC performs hyperspherical range queries 1.92 times faster than the Pyramid-Technique, 1.59 times faster than R^*-tree, 1.45 times faster than the X-tree, and 31.5 times faster than the sequential scan. And, when the radius of query circle is 0.5, the SPY-TEC performs hyperspherical range queries 1.20 times faster than the Pyramid-Technique, 1.87 times faster than R^*-tree, 1.78 times faster than the X-tree, and 2.85 times faster than the sequential scan.

Through the experiment on real data sets, we could observe that the SPY-TEC clearly outperforms the Pyramid-Technique and other index structures for the reasonable radii of query circles when processing hyperspherical range queries.

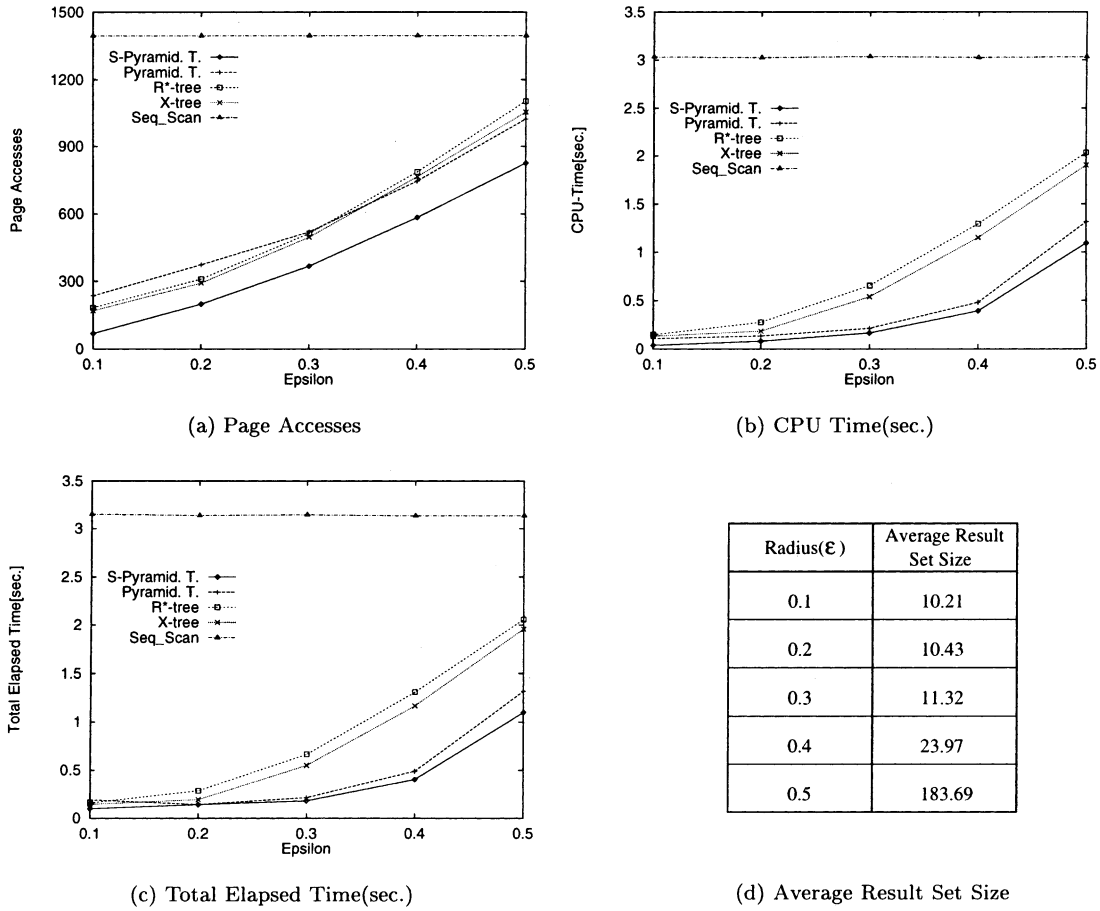
6.2. Retrieval effectiveness

QBIC (Faloutsos et al., 1994) system proposed a variation of normalized precision and recall (described below) to assess the performance of a system that employs similarity-based retrieval. We measured the performance of SCARLET by using the measure proposed by QBIC.

- *AVRR* is the average rank of all relevant, displayed items (the first position is 0th).
- *IAVRR* is the ideal average rank. It is the maximum when all relevant images are retrieved on the top: $\mathrm{IAVRR} = (0 + 1 + \cdots + (T - 1))/T$.

Notice that the ratio of *AVRR* to *IAVRR* gives a measure of the effectiveness of our retrievals. That is, when the ratio of *AVRR* to *IAVRR* achieves 1, the result of the retrievals is the most ideal (Faloutsos et al., 1994).

For the experiments, we selected five query images as depicted in Fig. 15 and decided beforehand which images are relevant for each query image. The number in Fig. 15 refers to the number of relevant images to each query image. The



(a) Page Accesses

(b) CPU Time(sec.)

(c) Total Elapsed Time(sec.)

Radius(ε)	Average Result Set Size
0.1	10.21
0.2	10.43
0.3	11.32
0.4	23.97
0.5	183.69

(d) Average Result Set Size

Fig. 14. Performance behavior on real data.

(a) 9 (b) 8 (c) 6 (d) 12 (e) 4

Fig. 15. Query images and the number of relevant images.

test database consisted of approximately 100 images and the total number of relevant images for five query images is 39. We extract 16-dimensional feature vector from each image and use ε large enough to retrieve all relevant images.

Table 1 shows the values of *IAVRR* and *AVRR*, respectively, for five query images and averaged over the five queries. Notice how close the AVRR = 6.96: IAVRR = 3.4. This implies that the average relevant image appears in the 7th position; given that we displayed 20 images, most of the relevant images were displayed and were also in good ranks. In this experiment, the ratio of AVRR to IAVRR(AVRR/IAVRR) is 2.05 and this value is similar to that of QBIC [1]

[1] In QBIC, AVRR/IAVRR = 1.93 for color-based retrieval, 2.05 for shape-based retrieval.

Table 1
Results of experiments on effectiveness

Query images	(a)	(b)	(c)	(d)	(e)	Average
IAVRR	4	3.5	2.5	5.5	1.5	3.4
AVRR	8.2	7.6	5.4	10.1	3.5	6.96

7. Conclusion

In this paper, we proposed an efficient feature extraction method which employs edge detection and wavelet transform. In particular, to facilitate speedy image retrieval, we proposed the SPY-TEC which is based on a special space partitioning strategy that is suitable for hyperspherical range queries. We also proposed the algorithms for processing hyperspherical range queries on the SPY-TEC.

The SPY-TEC transforms d-dimensional data space to 1-dimensional data space and manages a B^+-tree to store and access the transformed 1-dimensional values as the Pyramid-Technique does, and therefore, we are able to use all of the advantages of a B^+-tree. Through various experiments using both synthetic and real data sets, we showed that the SPY-TEC outperforms other related techniques when processing hyperspherical range queries. Moreover, the SPY-TEC is an optimal high-dimensional index structure for similarity search. Therefore, it should be used in any CBIR applications so far as the feature data of an image can be represented as a vector.

Finally, we designed and implemented SCARLET, a simple prototype system which facilitates image retrieval by the shape trend of an image, using methods proposed in this paper. And, we also showed the retrieval effectiveness of SCARLET.

In future, we plan to study an efficient algorithm for another similarity query, i.e., the k-nearest neighbor query on the SPY-TEC and develop a system that can support the retrieval by various contents of images such as the color or texture information in addition to the shape.

Acknowledgements

We are thankful to Dr. Stefan Berchtold in AT&T Bell laboratory for providing the X-tree code, the Pyramid-Technique code, and helpful comments.

References

Apostol, N., Rajeev, R., Kyuseok, S., 1999. WALRUS: A similarity retrieval algorithm for image databases. In: Proceeding of the ACM SIGMOD International Conference on Management of Data, pp. 395–406.

Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B., 1990. The R^*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 322–331.

Berchtold, S., Böhm, C., Keim, D.A., Kriegel, H.-P., 1997. A cost model for nearest neighbor search in high-dimensional data space. in: ACM PODS Symposium on Principles of Database Systems, pp. 78–86.

Berchtold, S., Böhm, C., Kriegel, H.-P., 1998. The Pyramid-Technique: Towards breaking the curse of dimensionality. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 142–153.

Berchtold, S., Keim, D.A., Kriegel, H.-P., 1996. The X-tree: An indexing structure for high-dimensional data. In: Proceedings of the 22nd International Conference on Very Large Database, pp. 28–39.

Chan, Y.T., 1995. Wavelet Basics. Kluwer Academic Publishers, Dordrecht.

Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D., Equiz, W., 1994. Efficient and effective querying by image content. Journal of Intelligent Information System (JIIS) 3 (3), 231–262.

Faloutsos, C., 1995. Fast searching by content in multimedia databases. Data Engineering Bulletin 18 (4), 31–40.

Graps, A., 1995. An introduction to wavelets. IEEE Computational Science and Engineering 2 (2).

Hirata, K., Kato, T., 1992. Query by visual example-content based image retrieval. In: Advances in Database Technology (EDBT '92), pp. 56–71.

Jacobs, C.E., Finkelstein, A., Salesin, D.H., 1995. Fast multiresolution image query. In: Proceedings of the 1995 ACM SIGGRAPH.

Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., Protopapas, Z., 1998. Fast and effective retrieval of medical tumor shapes. IEEE Transition on Knowledge and Data Engineering 10 (6), 889–904.

Mallat, S., 1989. A theory for multiresolution signal decomposition: the wavelet representation. IEEE Transition on Pattern Analysis and Machine Intelligence 11 (7), 2091–2110.

Martin, G.E., 1996. The Foundations of Geometry and the Non-Euclidean Plane. Springer, Berlin.

Ooi, B.C., Tan, K.L., Chua, T.S., Hsu, W., 1998. Fast image retrieval using color-spatial information. The VLDB Journal 7 (2), 115–128.

Stollnitz, E.J., DeRose, T.D., Salesin, D.H., 1995. Wavelets for computer graphics: A primer, part1, part2. IEEE Computer Graphics and Applications 15 (3-4).

Smith, J.R., Chang, S.-F., 1996. VisualSEEk: a fully automated content-based image query system. In: ACM Multimedia 96, Boston, MA, pp. 87–98.
White, D.A., Jain, R., 1996. Similarity indexing with the SS-tree. In: Proceedings of the 12th International Conference on Data Engineering, pp. 516–523.

Dong-Ho Lee received his BS degree from Hong-Ik University, and MS degree in computer engineering from Seoul National University, Seoul, Korea, in 1995 and 1997, respectively. He is currently enrolled in the PhD program in computer engineering at Seoul National University. His research interests include high-dimensional index techniques, content-based retrieval system, multimedia databases, and object-oriented databases.

Hyoung-Joo Kim received his BS degree in computer engineering from Seoul National University, Seoul, Korea, in 1982 and his MS and PhD in computer engineering from University of Texas at Austin in 1985 and 1988, respectively. He was an assistant professor of Georgia Institute of Technology, and is currently a professor in the Department of Computer Engineering at Seoul National University. His research interests include object-oriented databases, multimedia databases, HCI, and computer-aided software engineering.