

FINDING SIMILAR IMAGES QUICKLY USING OBJECT SHAPES

Heng Tao Shen
Department of Computer Science
National University of Singapore
Singapore 117543
shenht@comp.nus.edu.sg

ABSTRACT

Retrieving images from a large image collection has been an active area of research. Most of the existing works have focused on content representation. In this paper, we address the issue of identifying relevant images quickly. This is important in order to meet the users' performance requirements. We propose a framework for fast image retrieval based on object shapes extracted from objects within images. The framework builds a hierarchy of approximations on object shapes such that shape representation at a higher level is a coarser representation of a shape at the lower level. In other words, multiple shapes at a lower level can be mapped into a single shape at a higher level. In this way, the hierarchy serves to partition the database at various granularities. Given a query shape, by searching only the relevant paths in the hierarchy, a large portion of the database can thus be pruned away. We propose the *angle mapping* (AM) method to transform a shape from one level to another (higher) level. AM essentially replaces some edges of a shape by a smaller number of edges based on the angles between the edges, thus reducing the complexity of the original shape. Based on the framework, we also propose two hierarchical structures to facilitate speedy retrieval. The first, called Hierarchical Partitioning on Shape Representation (HPSR), uses the shape representation as the indexing key. The second, called Hierarchical Partitioning on Angle Vector (HPAV), captures the angle information from the shape representation. We conducted an extensive study on both methods to see their effectiveness and efficiency. Our experiments on sets of images, each of which has objects around from 1 to 30, showed that the framework can provide speedy image retrieval without sacrificing on the quality. Both proposed schemes can improve the efficiency by as much as hundreds of times to sequential scanning. The improvement grows as image database size, objects per image or object dimension increase.

Keywords:

Shape-based image retrieval, shape indexing, partitioning, shape representation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'01, November 5-10, 2001, Atlanta, Georgia, USA.
Copyright 2001 ACM 1-58113-436-3/01/0011...\$5.00.

1. INTRODUCTION

For a content-based image retrieval system to be acceptable to users, there are two requirements that must be met. First, the system must be able to capture the content of images so that relevant images can be retrieved. Second, the system must retrieve the relevant images quickly. The first essentially deals with the effectiveness of the system, while the latter addresses the efficiency issue. In this paper, we shall adopt a simple representation of shapes as the content of an image, and address the efficiency issue. Finding images quickly has become increasingly important as the number of images continue to grow at an alarming rate.

In this paper, we propose a framework for fast image retrieval based on object shapes extracted from objects within images. Each object shape is represented as a set of rectilinear edges. The framework builds a hierarchy of approximations on object shapes such that a shape at a higher level serves as a coarser approximation of a shape at a lower level. This is realized using an *angle mapping* (AM) mechanism. AM replaces a sequence of connected edges by a smaller number of edges based on the angle between the edges. Thus, the "dimensionality" of a shape is reduced significantly at each level. In this way, the hierarchy serves to partition the database at various granularities – at higher levels, there are fewer partitions but more (original) shapes share the same approximated shape representation; at lower levels, there are more partitions with fewer shapes sharing the same shape representation. By picking one or more paths along the hierarchy during a search process corresponds to pruning the images indexed by the other paths. Thus, only certain partitions need to be examined.

Based on the framework, we also propose two different index structures. The first, called Hierarchical Partitioning on Shape Representation (HPSR), employs the shape representations as the indexing keys. The second, Hierarchical Partitioning on Angle Vector (HPAV), makes use of the angle information obtained from the shape representations as the indexing keys. While the former is a more accurate mechanism, the latter is a more compact representation and facilitates inexpensive computation.

We implemented the two methods, and evaluated their performance a large collection of images. Our results showed that the proposed framework is indeed efficient. In particular, HPSR can lead to 93% recall, while HPAV can provide 77% recall. Both methods can also outperform the sequential scan method by hundreds times in terms of response time to retrieve the relevant images.

The rest of paper is organized as following. We will discuss some related work in section 2. In section 3, we present the

hierarchical partitioning framework. Sections 4 and 5 present two schemes that are based on the framework respectively. A detailed performance study to compare two methods is reported in section 6. Finally we conclude our paper in section 7.

2. RELATED WORK

In content-based retrieval systems, feature functions are used to map *physical* objects into *logical* representations, like color histogram, 2D-strings [11], symbolic image [12], etc. To do so, a number of strategies for decomposing an image into its individual objects have been introduced. Recently, [1] introduced sliding window, which is efficient and domain-independent, to extract image regions and compute their signatures. For image retrieval, various algorithms have been proposed; the more recent ones can be found in [13 – 15].

As a crucial factor for image database access, several indexing structures have been proposed. The R-Tree [2] is widely used for multidimensional databases. Other structures such as the R+ -Trees [3], R*-Trees [4], TV-Tree [5] and NR-Tree [20] improve the quality of the tree. The STR [6] further utilizes the space usage. [7] used inverted lists to avoid sequential search of the image database. A content-based indexing technique based on the weighted center of mass (WCOM) was introduced [8]. Several indexing techniques on spatial retrieval have also been proposed. Recently, 2SMLSF [10] was introduced as an improvement of 2LSF [11]. However, these structures either explode exponentially with the dimensionality, eventually reducing to sequential scan, or contain tree nodes at larger and larger size from leaves to root, leading the searching at upper level to be slower than at lower level, which slows down the pruning process. Furthermore, most works do not stress on the indexing quality.

Relevant to our work is a wide variety of clustering techniques [16 – 18]. However, they focus on pattern classification and quality assessment. There is little attention on searching.

3. A HIERARCHICAL PARTITIONING FRAMEWORK VIA ANGLE MAPPING

In this paper, we represent the content of an image by the shapes of the objects in the image. When a large number of vertices are used to represent a shape, the shape can be approximately abstracted by rectilinear edges. For the rest of this paper, the image database is assumed to consist of logical image representations (in the form of a sequence of rectilinear edges). Clearly, each logical image may represent several similar shapes, which are the outlines of objects.

Intuitively, comparing two shapes with smaller number of vertices is computationally cheap compared to comparing two shapes with larger number of vertices. Inspired by this observation, we introduce the *angle mapping* (AM) method to map a high dimension shape's representation into its lower-dimension approximation.

As the name implies, AM approximates a shape based on the angles between edges. Given a shape, we note that sharper angles and longer edges carry more important information about shape. Imagine that if an angle between two connected edges is close to 180 degrees, we can actually represent them by a single edge without losing much information. Similarly, a very short edge may not be very useful in identifying the

outline of an object. Therefore, to reduce the shape's dimension, our way is to prune those edges that are less important in identifying the shape. From this point of view, we combine angle between edges and length of edge to prune those less important edges. By repeatedly applying the mapping mechanism, we can derive a hierarchical partitioning framework where shapes at the lowest level of the hierarchy represents the logical shape representation, and shapes at the internal levels represent coarser representations. We note that at different levels, the importance threshold varies. The higher the level is, the larger the importance threshold is, so that fewer dimension's representation can be produced. Eventually, at the final stage, the approximate representation with the least number of dimensions, that can cover most area of the original shape, can be obtained.

We assume that the dominating outline can effectively and approximately represent a shape. Therefore, we simplify the problem to mapping dominating outline.

Definition 3.1: (Dominating Outline)

For a shape with polygon, its dominating outline is defined as the polygon with the biggest area. Otherwise, the longest route between any two end points is treated as its dominating outline.

There are several issues that have to be addressed in the framework. The first issue is the terminating criterion, i.e., when should we stop the process of generating approximations? Our current solution is to provide a tuning knob that controls the maximum number of approximations the system can support. This essentially corresponds to the number of levels in the hierarchy. We denote the number of mapping level as N .

The second issue concerns the choice of edges to merge. As a first cut, we only prune those edges with angle that is greater than 90 degrees. Given N , the *Angle Interval* (AI), which should be mapped to produce level i representation at each level, is defined as:

$$AI[i] = (90+90*(i-1)/N, 90+90*i/(N)) \text{ or } (270-90*(i-1)/N, 270-90*i/(N))$$

For example, if $N=3$, then to produce level-3 representation, the edges with angle in interval $(90+60, 90+90]$ should be reduced. Similarly, to produce level-2 and level-1 representations, the AI becomes $(90+30, 90+60]$ and $(90, 90+30]$ respectively.

Finally, we need to determine what is a "short" edge. In this work, we define this length as *Prune Length Threshold* (PLT), and can be tuned by the system for different applications. Algorithm 3.1 shows the framework with the AM algorithm:

Algorithm 3.1: A Hierarchical Partitioning Framework

1. Find the shape's dominating outline and initialize it as level N representation.
2. For level i from N to 1 do
 - 2.1 Check any two connected edges. If their angle lies in $AI[i]$ then map two edges into one or three edges into two. Compute the new edge's length and angles, and then update representation.
 - 2.2 If any edge $<$ PLT, choose the shorter connected edge and prune both edges. Then connect two different ends of two edges to produce a new edge. Update representation.

- 2.3 Go back 2.1 if any angle lies in AI[i] or go back to 2.2 if any edge < PLT.
- 2.4 Get level-i representation.
- 2.5 If there are too many shapes at level i compared to user specified threshold
 - 2.5.1 Cluster similar shapes
 - 2.5.2 Identify a representative from the shape

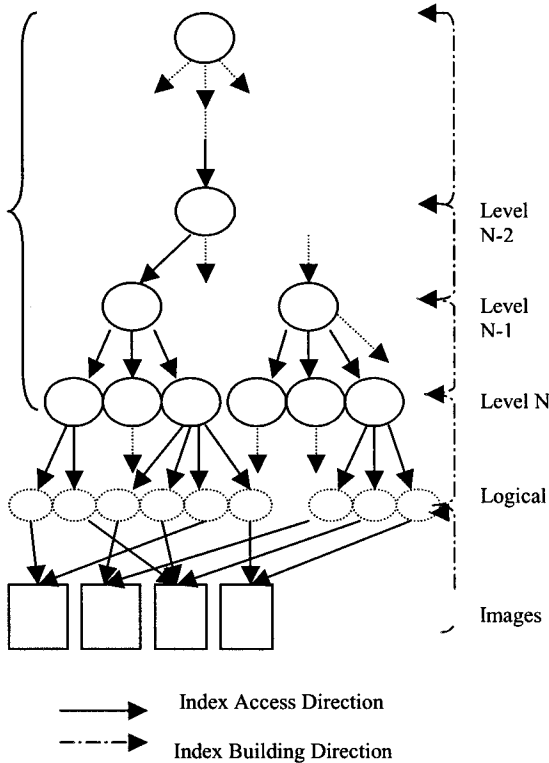


Figure 1: The hierarchical framework

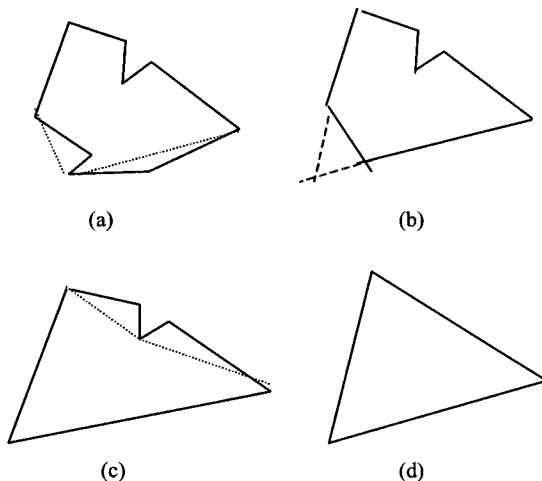


Figure 2: 3-Level Representations for an example shape with polygon. (a) Example logical shape representation (b) level-3 representation (c) level-2 representation (d) level-1 representation

Figure 1 shows the hierarchical framework structure. One example shape mapping process for $N=3$ is shown in above Figure 2. As Figure 2(a) shows, two sets of edges are pruned to get level-3 representation. One of them is because of its short length, the other is because the angle value between the two edges lies in $(90+60, 90+90]$. In the framework, when we try to prune edges which have angle lying in $AI[i]$ at level- i , to keep the original shape's information as much as possible, two ways are considered: either directly connect two ends of two edges to create a new edge or extends the shorter edge's two connected edges to form an angle if can. As in Figure 2 (b) shows, there are two ways to prune one edge. The one removing smaller area of the shape will be chosen. We then calculate the new edges' lengths and update the angle in between. If the updated angles are greater than or lie in $AI [i]$ at level- i , then further pruning should be carried out. For instance, in Figure 2 (c), after two edges are pruned, the newly produced two edges form an angle which is greater than 90 degree again. Then further pruning is made, and finally we get the level-1 representation with dimension of 3 only.

We note that a path along the hierarchy basically leads to a partition where all object shapes have similar shape representations at all levels along the path. As such, during querying, if only relevant paths need to be examined, we can effectively prune away the other paths, i.e., only certain partitions will be searched. This will lead to an efficient image retrieval system.

Definition 3.2: (Dimension Reduction Ratio)

The Dimension Reduction Ratio (DRR) from level- i to level- $(i-1)$ is defined as:

$$DRR(i, i-1) = [\text{Dim}(R(i)) - \text{Dim}(R(i-1))] / \text{Dim}(R(i))$$

where $\text{Dim}(R(i))$ refers to the dimension of representation (number of edges) at level- i .

The framework maps high-dimension shape representations into multiple levels of representations (MLR) with lower-dimensions. As shown in Figure 2, from the logical shape to its level-1 representation, dimension is reduced by $(9-3)/9 = 2/3$. With multiple applications of angle mapping and pruning of relatively short enough edge, the framework can eliminate the differences in objects with similar shapes. Thus similar shapes should have same or similar MLRs. With largest difference allowed in level-1, similar shapes should have the most similar (if not the same) level-1 representation, which is the most common shape, such as triangle, rectangular, etc.

During the mapping process, the value of N can be used to control the AI at each level. Both AI and PLT are used to determine the max difference from current level representation to its mapped higher-level representation.

4. HIERARCHICAL PARTITIONING WITH SHAPE REPRESENTATIONS

4.1 Indexing Building

Based on the framework discussed in Section 3, we propose a hierarchical partitioning scheme with shape representations (HPSR). The scheme essentially uses the shape information (e.g., length of edges, angles with neighboring edges) as the indexing key. In HPSR, the clustering scheme employed is a simple one and is similar to that used in CURE [17].

Basically, two shapes that are similar (see similarity measure shortly) are grouped as a cluster, and a representative is picked to represent the cluster. This process is repeated until the desired number of clusters is obtained. In our work, we have set the desired number of shapes per level as that matching the reduction ratio which is defined as follows.

Definition 4.1.1: (Representation Reduction Ratio)

The Representation Reduction Ratio (RRR) from level- i to level- $(i-1)$ is defined as:

$$RRR(i, i-1) = \frac{\text{NumberOfNodesAtLevel}(i)}{\text{NumberOfNodesAtLevel}(i-1)}$$

RRR can be dynamically controlled by the distance threshold for partitions. Large RRR leads to shallow tree, vice versa. The HPSR scheme is described algorithmically as follows.

Algorithm 4.1.1: HPSR

1. Initialize n to N .
2. For all logical shapes, apply AM to get their level- n representation.
3. Merge identical representations into partitions.
4. If number of partition is too big, cluster similar representations given distance threshold.
5. For each partition, produce partition's representative as a node at level n .
6. Build connection between level- n nodes and their children.
7. If $n > 1$, map level n nodes to level $n-1$ representations and decrease n by 1, then go to step 3.

Compared with existing indexing structure, HPSR produces the tree with nodes which are representatives of partitions with lower and lower dimensions from bottom to top. One example indexing structure with $N=2$ for three images is shown in figure 3. As we can see, the $RRR(\text{logical level}, 2) = (6-4)/6 = 1/3$ and $RRR(2,1) = (4-2)/4 = 1/2$. The DRR effect can be also seen clearly for each high-dimension representation.

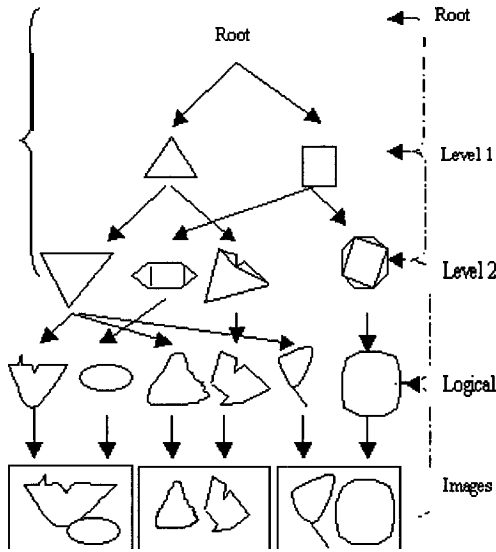


Figure 3: an example 2-level HPSR indexing tree for a database with 3 images.

4.2 Query Processing

Given a query image, all shapes are extracted, whose approximate representations are **first** generated by using the same technique above without partition. Starting at level-1 nodes, query's level-1 representation is compared with them. If no one similar node can be found, then it can be certain that there are no shape underneath the indexing tree that satisfy the query. If it can be found, there is a chance that some logical shapes may satisfy the query. In this case, searching resumes at lower level's nodes that are pointed from the node found. This procedure is repeated until either no lower level to go or the leaf level. The access path is also indicated in figure 2 and 3. Formally, the process goes as follows:

Algorithm 4.2.1: Query Processing in HPSR

1. Generate query shape's N multi-level representations.
2. Initialize level n as 1.
3. Compute the similarity between query shape's level- n representation and level- n nodes.
4. If no similar node is found, return null.
5. If leaf level is reached, get shapes' logical representations pointed by leaf nodes and compute the similarity with query shape. Return those similar shapes.
6. Increase n by 1 and retrieve the similar node's children as level- n nodes. Go back to step 3.

During query processing from root to leaf, more than one similar node may be found at certain level. A parameter can be set to determine how many branches we want to search through. The chance at top levels is low. Another important feature that we can see is that, at higher levels, the comparisons are made between low-dimension representations. This results in much more **efficient** indexing searching since the computation time is reduced greatly.

4.3 HPSR Matching

To successfully find the most similar node at each level given a query image's shape representations, a similarity measure is essential. One popular method for polygon shape representation is the turning function [19], which is invariant to position, scale, and rotation. In this paper, we extended turning function to be applicable to any rectilinear shape.

During the search process, **only** low-dimension representations derived from dominating outlines (definition 3.1) are involved. Dominating outline for a shape is either polygon or one-by-one connected edges. Polygon only matches polygon, and it is same for one-by-one connected edges. To get the similarity between two dominating outlines, we apply the turning function. The turning function represents the tangent of a point on the boundary with respect to a reference axis of an arbitrary orientation. During the traversal of the boundary, the tangent at each point is computed. The starting point on the outline corresponds to the origin point on the turning function. Turning function is supposed to test on all possible reference axes with different degree from 0 to 360, and on all the choice of origin. The formula is

$$D(a, b) = \int_0^1 (a(x) - b(x))^2 dx$$

Where $a(x)$ and $b(x)$ are the turning functions for shape a and b . x is the outline length. As we only store the relative length for each edge; therefore, x ranges from 0 to 1. Given two shapes a and b , a smaller D value means that they are more similar. If it's polygon, we have to compute over all choice of

reference axis, which are angles of polygon, and origins for the second polygon, which are vertices of second polygon, in order to find the minimal difference. So the time is $O((M+N)MN)$ or $(O(M^3))$ where M is the edge number. If the shapes are not polygons, we simply put the origin as either end point. So the time is $O((M+N)*2)$ or $O(M)$ simply. The following figure shows an example for turning function.

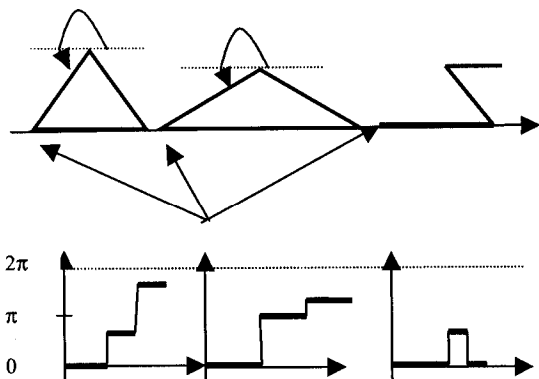


Figure 4: example shapes with their turning functions.

To get more precise matching, the branch outline starting from each vertex at dominating outline may also be computed correspondingly by applying turning function (similar to one-by-one connected edges). The overall similarity between two approximations is:

$$S(a,b) = D(a,b) + \frac{1}{\alpha * \sum_{i=1}^M D(B[a(i)], B[b(i)])}$$

where α indicates that dominating outline is α times important than branch outline. $B[a(i)]$ indicates that it's the i th branch for shape a from origin. i is from 1 to M , where M is the number of edge in the dominating outline of shape a .

5. HIERARCHICAL PARTITIONING WITH ANGLE VECTORS

In HPSR, the computation cost to compare two shapes can still be fairly high. As AM processing mainly concerns with angles, in this section, we introduce a compact representation called Angle Vector to further improve on the efficiency. We shall call the resultant structure a Hierarchical Partitioning with Angle Vector (HPAV).

To map a shape representation to its corresponding AV, only angle information is considered. Section 3 describes how to define Angle Interval (AI) for each level. Given mapping level N , AI has 3 intervals. We add one more interval called AI [0] which range from $(0, 90]$ or $[270, 360)$. Based on the AI, we define AV as a vector, each dimension of which contains the number of angles whose degrees lie in different AI. For example, the AV for each representation in Figure 2 can be expressed as follows in figure 5. For instance, for level-2 representation from Figure 2(c), we can get its 2-dimension AV, where the first dimension indicates the number of angles lying in AI [0] and the second dimension indicates the number of angles lying in AI [1], as mapping from level 3 to level 2 has eliminated those angles lying in AI [3]. Therefore, as mapping from bottom to top, the AV dimension decreases by 1 at each level. Finally at the top,

only one dimension is left which indicates the number of angles lying in AI [0]. In our implementation, we separate shapes with polygon and without polygon, as both have different number of edges when they have the same number of angles.

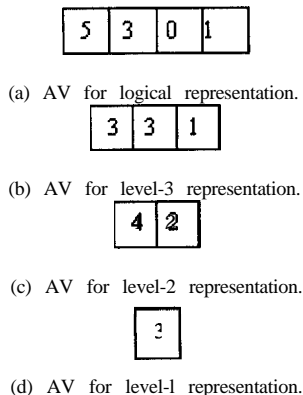


Figure 5: AVs for an example shape's MLRs

To construct the HPAV structure, we need to build the HPSR structure first. We then map each node in the HPSR structure into its corresponding AV to form the HPAV structure. Thus, the only difference between the HPSR and HPAV is the node representation. In HPSR, each node is a low-dimension level-1 shape representation. However, in HPAV, nodes are in the form of AV. Both are indexed with the same quality. But one extra cost for HPAV structure is the mapping process from shapes to AVs. One example HPSV tree obtained from HPSR tree (Figure 3) is shown in Figure 6.

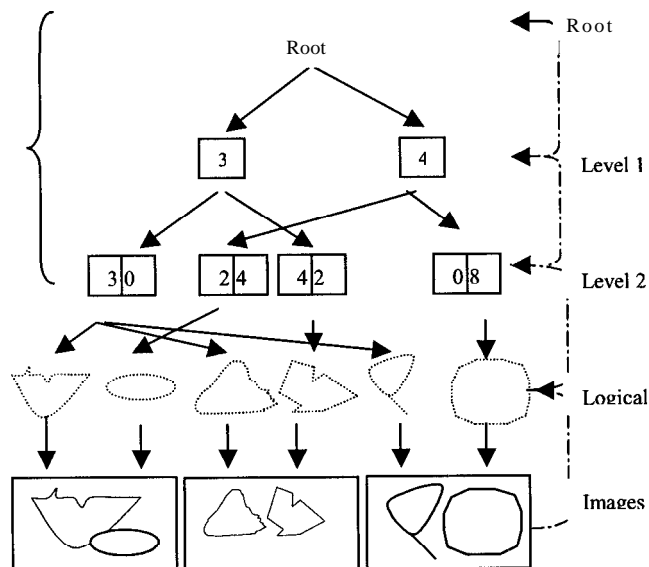


Figure 6: an example 2-level HPAV Indexing tree for a database with 3 images.

In HPAV tree, each level has much lower fixed dimensions, which is decided by the mapping level N . From the leaf level to level-1, the number of dimensions keeps decreasing by one at each level. Finally at level 1, only 1 dimension is left. The

DRR values produced at each level by HPAV may be smaller than HPSR. However, the HPAV structure starts with initially very low dimension, which is N only. Furthermore, the node size is much smaller compared to HPSR at each level. Therefore, the storage space can be reduced dramatically. This also implies faster path traversal.

For query processing, it's same to the algorithm 4.2.1 (for HPSR), except that now we use AV at each level from HPSR and use AV to go through the HPAV tree to find similar leaf nodes. Although the tree is still in the same quality with HPSR tree, the wrong branch may be taken based on AV comparison, as there is a chance that irrelevant representations may be mapped into the same AV. The testing results are reported in the next section.

As for the AV comparison, inspired from the formula for the angle between two *nonzero* vectors in *2d-space*, define the similarity measure as follows:

$$Sim(AV1, AV2) = \frac{AV1 \cdot AV2}{\|AV1\| * \|AV2\|}$$

6. EXPERIMENTS

In this section, we report an experimental study conducted to study the effectiveness of the proposed methods.

6.1 Set Up

All of the experiments were performed on an Intel Pentium III Processor 700MHz with 128 Mbytes of RAM. The testing **datasets** consist of several sets of databases. Each set is used for a different experiment.

We evaluated the two structures HPSR and HPAV on their effectiveness and efficiency, as well as storage requirements. As we use the original shape's representations for the final comparison, the accuracy is always 100%. We use another important factor, Recall, to determine the effectiveness of the scheme. To see the efficiency of HPSR and HPAV schemes, we define one parameter called Time Reduction Ratio (TRR) as:

$$TRR = [Time (sequential) - Time (index)] / Time (index)$$

where index refers to HPSR or HPAV. To see different impact on storage requirement on the structures, we also compare them over a set of databases. In our experiments, we have created images with a number of objects. To determine whether an image is relevant to a query image, we adopt a simple strategy: A database image is considered to be relevant to a query image if the database image contains 80% of the shapes in the database image are similar to those of the query image.

6.2 Tuning Mapping Level and PLT

The AM process will determine whether similar shapes can be mapped into the same partition, which directly affects the quality of the schemes. During the AM process, two parameters may affect the results: mapping level N, which decides Angle Interval, and Prune Length Threshold (PLT). Based on recall and CRR, we want to choose the optimal value for N and PLT.

We run this experiment on the HPSR structure with synthetic 5,000 images, each of which has an average of 15 objects. The query image contains 15 objects. The following 2 figures show the impact on TRR and recall for different N values.

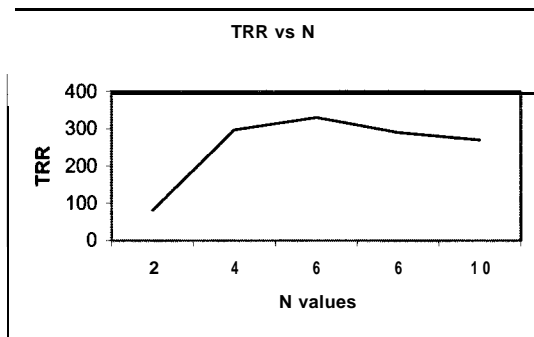


Figure 7: Impact on TRR by Mapping Value

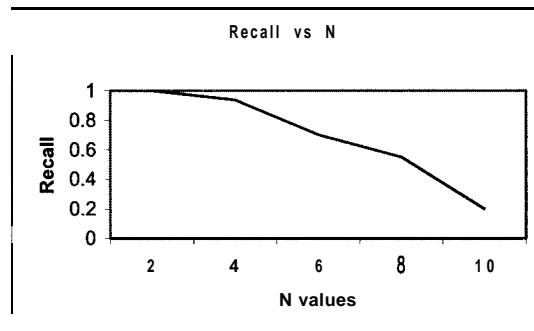


Figure 8: Impact on recall by mapping value.

From Figure 7, we noticed that TRR increases when N increases from 2 to 6. This is expected since a smaller N means a larger partition and hence more shapes have to be compared. However, as N continues to increase from 6, the TRR begins to reduce slowly. One main reason we believe is that the overheads in examining too many levels become significant. In Figure 8, we see the recall results. As shown, recall decreases as the value of N increases. The reason is that higher mapping level causes more Angle Intervals. Due to strict angle mapping, some similar representations may be classified into different leaf nodes. This is more serious as N increases. Therefore, from both figures, we obtain one "optimal" value for N, which is 4.

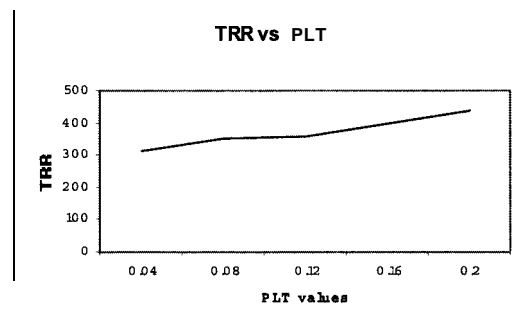


Figure 9: Impact on TRR by PLT

In the same experiment, we also tested the impact caused by PLT with N=4. Figure 9 indicates that with PLT increasing, the TRR grows also, as it produces lower-dimension representation with fewer edges. In Figure 10, recall keeps decreasing as larger PLT means coarser approximation.

Combining the results of the two experiments, we pick the "optimal" range for PLT to be around 0.1.

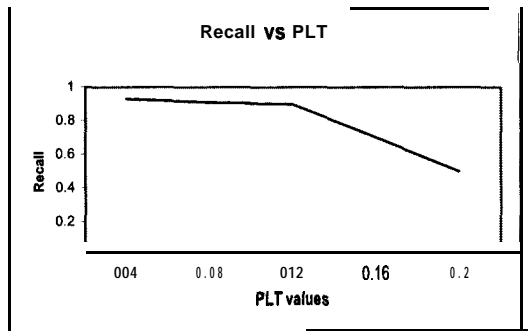


Figure 10: Impact on Recall by PLT

6.3 Effectiveness of HPSR and HPAV

With the "optimal" values for N and PLT, we shall use them to compare both HPSR and HPAV structures. This experiment is done on a set of databases with 1000, 2000, 3000, 4000, and 5000 images respectively. Figure 11 shows the results. The recalls for both methods have *no* obvious up or down. The average recall for HPSR and HPAV is 93% and 77% respectively, given the same database. For HPSR, the high recall confirmed that our framework can produce good shape approximations that can be used to prune the search space effectively. For HPAV, the low recall is due to the fact that irrelevant shape representations may be mapped to the same AV, i.e., AV is essentially a coarser shape representation. Therefore, the query AV at certain level has a chance to choose an invalid branch. And this probability is around 23%. In all the above experiments, we have allowed only one unique path to find a single leaf node. We have also tried with allowing multi-branch searching at the leaf nodes. The results are better. However, efficiency will drop quickly as more comparisons are involved.

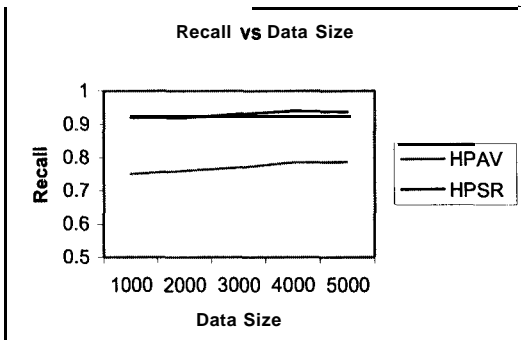


Figure 11. A comparative study on effectiveness.

6.4 Efficiency of HPSR and HPAV

The main goal of our methods is to speed up the retrieval process without sacrificing on their effectiveness. To see the efficiency of the two schemes, the first experiment was done on a set of databases with 1000, 2000, 3000, 4000, and 5000 images. Each set has the same average number of 15 objects ranged from 1 to 30. Query is an image with 15 objects. The purpose is to understand how the TRR varies with database size increasing.

As Figure 12 shows, the TRRs for both schemes increase as database grows. The difference between HPAV and HPSR can be seen clearly. As the number of dimensions at each

level for HPAV tree is fixed, its TRR is close to a linear function of data size. Since its searching tree time is almost constant, the increasing for HPAV is greater than HPSR.

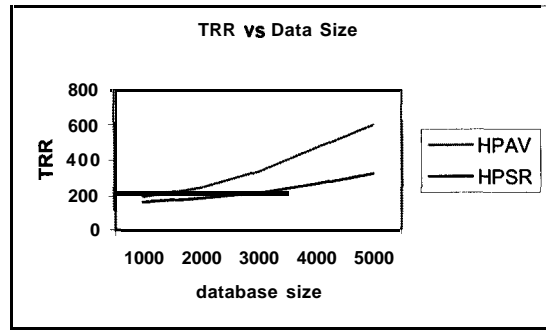


Figure 12: TRR varies with database size

Next we want to study the effect of the number of objects per image on the efficiency of the schemes. We use 5 sets of 5,000 images; each set has different object number per image from 5, 10, 15, 20, and 25. Figure 13 indicates the impact. From Figure 12, we noticed that TRR increases as the number of objects per image grows. Figure 13 also showed that TRR for HPAV is much more affected by the number of objects than that for HPSR.

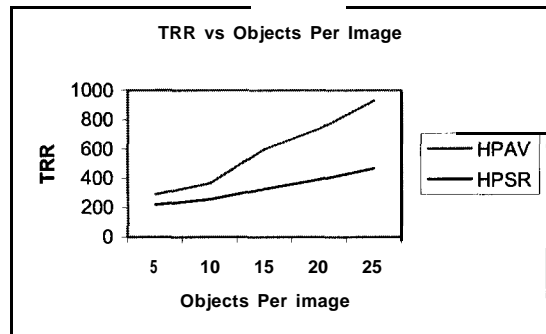


Figure 13: Number of Objects per Image vs. TRR

All the previous experiments were done with average 15 dimensions/edge per object. In this experiment, we vary the average number of dimensions per object from 5 to 25. We choose a set of 5000 images with 15 objects per image and test on the dimension impact on TRR. As we can see from Figure 14, both schemes' TRRs increase much faster than the impact on data size and number of objects per image. When the number of dimensions is fewer than 5, speed up is not so much as not many dimensions are reduced by the AM. However, when the number of dimensions exceeds 15, the gain becomes significant. This is the power of AM. We also believe that a larger number of objects contained in a query image would also provide a higher gain.

6.5 Storage Space for IMLR and IAV Tree

Storage requirement is also important when the indexing structure is needed to be stored somewhere. We did an experiment on sets of databases with 1000, 2000, 3000, 4000, and 5000 images respectively. The average number of objects contained in an image is 15.

From Figure 15, we can see that as the database size increases, the storage of both schemes grows slowly. In fact

for HPAV, the size of the structure is almost constant. Its size increases only when there is a chance to increase the nodes in the structure. Since HPSR maintains shape information, its storage requirement is larger - it needs several times of space more than the HPAV scheme. As database keeps increasing, the gaps between data size, HPSR and HPAV also become larger.

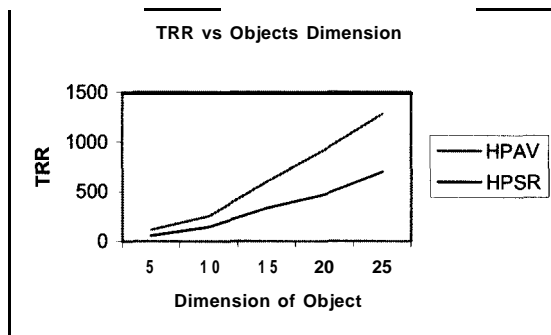


Figure 14: Dimension of Object vs TRR.

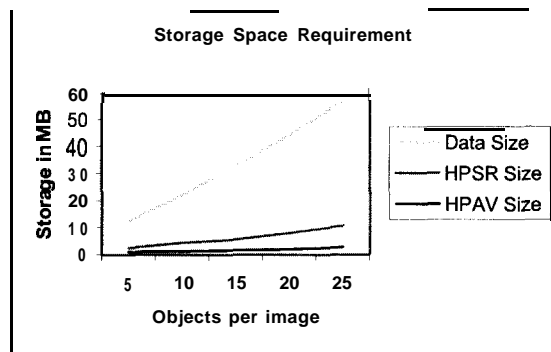


Figure 15. Effect of number of objects per image.

7. CONCLUSION

In this paper, we have proposed a framework for partitioning image databases based on shapes of objects. The framework makes use of approximations of shapes obtained using the angle mapping method that reduces the dimensionality of shapes. We also proposed two methods based on the framework. While the first method, HPSR, uses shape information to be the indexing key, the second method, HPAV, employs a compact but coarser vector representation. We implemented both schemes and evaluated their effectiveness and efficiency as well as storage requirement. The results of our study showed that both methods are effective and can obtain up to 93% recall for HPSR and up to 77% recall for HPAV. On the other hand, HPAV is much more efficient than HPSR and requires only around 1/3 the storage space of HPSR tree.

REFERENCES

[1] A. Natsev, R. Rastogi, and K. Shim, "WALRUS: a similarity retrieval algorithm for image database". SIGMOD '99, page 395-406.
 [2] A. GUTTMAN. 1984. "R-Trees: A dynamic index structure for spatial searching". In *Proceedings of the ACM SZGMOD Conference on Management of Data*, ACM Press, New York, NY, 47--57.

[3] T. SELLIS, N. ROUSSOPOULOS, AND C. FALOUTSOS. 1987. "The R+-Tree: A dynamic index for multidimensional objects". In *Proceedings of the 13th Conference on Very Large Data Bases*, Berkeley, CA, 507-518.
 [4] BECKMANN, N., KRIEGEL, H., SCHNEIDER, R., AND SEEGER, B. 1990. "The R*-Tree: An efficient and robust access method for points and rectangles". *SZGMOD '90*, Atlantic City, NJ, May 23--25, 1990.
 [5] LIN, K-I, JAGADISH, H., AND FALOUTSOS, C. 1994. "The TV-Tree---An index structure for high dimensional data". *VLDB J. 3 (Oct.)*, 5 17--542.
 [6] LEUTENEGGER, S.T., EDGINGTON, J.M., AND LOPEZ, M. A. 1997. "STR: A simple and efficient algorithm for R-Tree packing". *ZCDE '1997*.
 [7] GUDIVADA, V.N. AND RAGHAVAN, V. V. 1995. "Design and evaluation of algorithms for image retrieval by spatial similarity". *ACM Trans. Inf. Syst. 13, 2 (Apr. 1995)*, 115--144.
 [8] HOU, T. , LUI, P., AND CHUI, M. Y. 1992. "A content-based indexing technique using relative geometry features". In *Proceedings on Image Storage and Retrieval Systems, SPIE---The International Society for Optical Engineering*, vol. 1662.
 [9] PETRAKIS, E. 1993. "Image representation, indexing and retrieval based on spatial relationships and properties of objects". Ph.D. Dissertation. Dept. of Computer Science, University of Crete.
 [10] ESSAM A. EI-KWAE and MANSUR R. KABUKA, "Efficient Content-based Indexing of Large Image Databases". *ACM Transactions on Information Systems*, Vol. 18, No. 2, April 2000, Pages 171--210.
 [11] LEE, S.Y. AND SHAN, M. K. 1990. "Access methods of image databases". *Int. J. Pattern Recogn. Artif. Intell. 4, 1, 27--44*.
 [12] CHANG, S.K., SHI, Q.Y., AND YAN, C. W. 1987. "Iconic indexing by 2D-strings". *IEEE Trans. Pattern Anal. Mach. Zntell. PAMZ-9 (3)*, 413--428.
 [13] GUDIVADA, V.N. AND JUNG, G. S. 1995. "An algorithm for content-based retrieval in multimedia databases". In *Proceedings of the International Conference on Multimedia Computing and Systems*, 90--97.
 [14] Lei zhu, Aidong Zhang, Aibing Rao and R. Srihari, "Keyblock: an Approach for Content-based image retrieval", *ACMMM2000*, page 157-167
 [15] Ben Bradshaw, "Semantic based image retrieval: a Probabilistic Approach", *ACM MM 2000*, page 167-177.
 [16] Jia Li, James Z. Wang, and Gio Wiederhold. "IRM: Integrated Region Matching for Image Retrieval", *ACM MM 2000*, page 147-157.
 [17] Tian Zhang, Raghu Ramakrishnan and Miron Livny; "BIRCH: an efficient data clustering method for very large databases", *SZDMOD '96*, Pages 103 - 114
 [18] Sudipto Guha, Rajeev Rastogi and Kyuseok Shim; "CURE: an efficient clustering algorithm for large databases", *SZGMOD*, 1998, Pages 73 - 84
 [19] Christopher R. Palmer and Christos Faloutsos; "Density biased sampling: an improved method for data mining and clustering"; *Proceedings of the 2000 ACM SZGMOD on Management of data*, 2000, Pages 82 - 92
 [20] E.Arkin, et al., "an efficiently computable metric for comparing polygonal shapes." *IEEE transaction on Pattern Analysis and Machine Intelligence*, Vol.13, no.3, 209-216
 [21] K. L. Tan, B.C. Ooi and L.F. Thiang, Retrieving Similar Shapes Effectively and Efficiently"; *Multimedia Tools and Applications*, Kluwer Academic Publishers, accepted for publication, 2001.