SIMILAR SHAPE RETRIEVAL IN MARS

Kaushik Chakrabarti *Michael Ortega-Binderberger* *Kriengkrai Porkaew* *Sharad Mehrotra*

University of Illinois, University of California
kaushikc,miki,porkaew@acm.org, sharad@ics.uci.edu

ABSTRACT

This paper presents a novel approach to representing 2-d shapes that adaptively models different portions of the shape at different resolutions, having higher resolution where it improves the quality of the representation and lower resolution elsewhere. The proposed representation is invariant to scale, translation and rotation. The representation is amenable to indexing using existing multidimensional index structures and can thus support efficient similarity retrieval. Our experiments show that the adaptive resolution technique performs significantly better compared to the fixed resolution approach previously proposed in the literature.

1. INTRODUCTION

Large repositories of digital images are becoming increasingly common in many application areas such as e-commerce, medicine, media/entertainment, education and manufacturing. There is an increasing application need to search these repositories based on their visual content. For example, in e-commerce applications, shoppers would like to find items in the store based on, in addition to other criteria like category and price, visual criteria i.e. items that look like a selected item (e.g., all shirts having the same color/pattern as a chosen one). To address this need, we are building the *Multimedia Analysis and Retrieval System (MARS)*, a system for effective and efficient content-based searching and browsing of large scale multimedia repositories [8]. MARS represents the content of images using visual features like color, texture and shape along with textual descriptions. The similarity between two images is defined as a combination of their similarities based on the individual features [8].

One of the most important features that represent the visual content of an image is the *shape* of the object(s) in an image [7, 5, 6, 4]. In this paper, we address the problem of similar shape retrieval in MARS. We propose a novel adaptive resolution (AR) representation of 2-d shapes. We show that our representation is invariant to scale, translation and rotation. We show how each shape, represented by AR, can be mapped to a point in a high dimensional space and can hence be indexed using a multidimensional index structure [1]. We define a distance measure for shapes and discuss how similarity queries, based on the above distance measure, can be executed efficiently using the index structure. The experimental results demonstrate the effectiveness of our approach and its superiority to the fixed resolution (FR) technique previously proposed in the literature.

2. RELATED WORK

In this section, we describe the fixed resolution approach and other prior work on shape retrieval.

This work was supported in part by the National Science Foundation under Grant No. IIS-9734300, in part by the Army Research Laboratory under Cooperative Agreement No. DAAL01-96-2-0003 and in part by NSF/DARPA/NASA Digital Library Initiative Program under Cooperative Agreement No. 94-11318. Michael Ortega-Binderberger is supported in part by CONACYT grant 89061 and an IBM Fellowship.

Fixed Resolution (FR) Representation In the FR approach proposed by Lu and Sajjanhar [5], a grid, just big enough to cover the entire shape, is overlaid on the shape. Each cell of the grid has the same size (hence the name fixed resolution). For example, in Figure 1(a), the shape is overlaid with a 8×8 grid. If we assume the grid to be 256×256 pixels, each cell is 32×32 pixels in size. Some grid cells are fully or partially covered by the shape and some are not. A bitmap is derived for the shape by assigning 1 to any cell with more than 15% of the pixels covered by the shape, and 0 to each of the other cells. The shape represented by the bitmap is shown in Figure 1(a) (below the 8×8 grid overlay). The quality of the representation (i.e. how closely it approximates the actual shape) improves as we go to higher resolutions. Figure 1(b) shows that a higher resolution bitmap (16×16 grid) represents the better (i.e. closer approximation to the original shape) than the 8×8 representation.

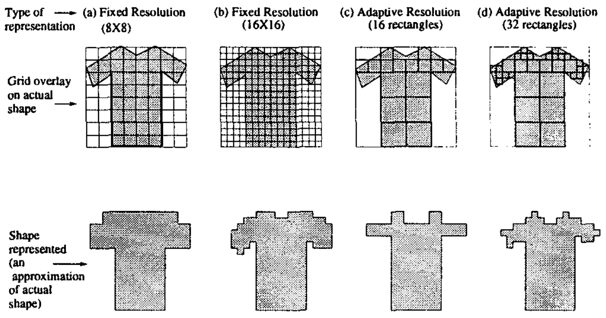
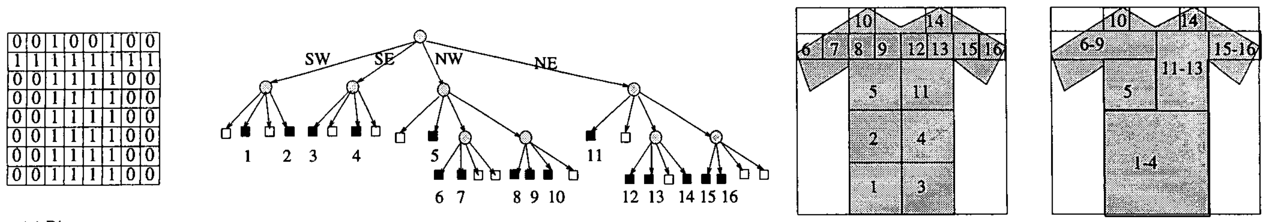

Type of representation
(a) Fixed Resolution (8X8)
(b) Fixed Resolution (16X16)
(c) Adaptive Resolution (16 rectangles)
(d) Adaptive Resolution (32 rectangles)
Grid overlay on actual shape
Shape represented (an approximation of actual shape)

Figure 1: Fixed resolution and adaptive resolution representations

To support similarity queries, [5] defines a distance measure between shapes. The distance between two shapes is defined as the number of bits by which their bitmaps differ from each other. A similarity query computes the distance of the query shape from every shape in the database and returns the k closest matches as answers to the user. [5] shows that the higher the resolution, the more closely it approximates the actual shape, higher the accuracy of the answers (in terms of satisfying the information need of the user). But high resolution also raises the query cost: at higher resolutions, we need more bits to represent each shape which increases both the I/O cost (as we need to scan a larger sized database) as well as the CPU cost (as we need to compute distances between longer bit sequences) of the query. The choice of the resolution thus presents a tradeoff between the query cost and accuracy.

Indexing in FR Approach There is no obvious way to index bitmaps. The only way to answer a similarity query is to sequentially scan the entire database i.e. compute the distance of the query from every item in the database. This technique is not scalable to large databases consisting of millions of shapes as it may take minutes or even hours to answer a query. To circum-



(a) Bitmap

(b) Quadtree decomposition of bitmap (gray indicates internal node, black indicates all bits 1, white indicates all bits 0). Numbers indicate z-order

(c) Adaptive Representation produced by quadtree decompostion (Numbers indicate z-order)

(d) Adaptive representation after merging rectangles based on z-order

Figure 2: Adaptive resolution representations

vent the problem, we propose to index the shapes using a multi-dimensional index structure (e.g., R-tree, SS-tree, Hybrid Tree). The index structure would reduce the cost of the query from linear (of sequential scan) to logarithmic to the size of the database, thus making the similarity queries scale to large sized databases. To use the index, instead of assigning a bit to a grid cell, we assign a count to each cell: the count being the number of pixels in the cell that are covered by the shape. For a 8 $\times$ 8 grid, we will get 64 values for each shape, thus mapping each shape to a point in a 64-dimensional space. The shapes can now be indexed using a 64-d point index structure. For a $n1 \times n2$ grid, the number of dimensions $d = n1.n2$. We can use a suitable L_p metric as the distance measure between the d-dimensional vectors. Given a query which is also a point in the d-dimensional space, we can use the index structure to quickly find the k shapes that are closest to the query using the standard k nearest neighbor algorithm [3]. As mentioned before, higher the resolution, more the dimensionality, higher the accuracy of the answers, but higher the execution cost of the query. We will refer to this count-based (and not the bit-based) representation as the FR representation for the rest of the paper.

Other shape retrieval techniques Other shape representation techniques include Fourier descriptors [9], moment descriptors [7], boundary points [6] and rectangle decomposition [4]. Recent studies have shown that the FR approach performs better than most of these approaches [5].

3. ADAPTIVE RESOLUTION APPROACH

Rotation, Scale and Translation Normalization Before we present the new shape representation, we describe how we normalize the shape to make it invariant to rotation, scale and translation. Our normalization strategy is similar to that of [5] developed for the FR representation. To guarantee rotation invariance, we need to convert an arbitrarily oriented shape into a unique common orientation. We first find the major axis of the shape i.e. the straight line segment joining the two points P_1 and P_2 on the boundary farthest away from each other [5]. Then we rotate the shape so that its major axis is parallel to the x-axis. This orientation is still not unique as there are two possibilities: P_1 can be on the left or on the right. We solve the problem by computing the centroid of the polygon and making sure that the centroid is below the major axis, thus guaranteeing an unique orientation. Let us now consider scale invariance. We define the bounding rectangle (BR) of a shape as the rectangle with sides parallel to the x and y axes just large enough to cover the entire shape (after rotation). Note that the width of the BR is equal to the length of the major axis. To achieve scale invariance, we proportionally scale all shapes so that their BRs have the same fixed width. In this paper, we fix the width of the BR to 256 pixels. We make the BR a square by fixing its height to 256 pixels as well; non-square shapes are handled by placing the shape at the bottom of the BR and padding zeroes in the remaining (upper) part of the BR. The shape is translation invariant as it is represented with respect to its BR (i.e. lower left corner of BR is considered (0,0)).

Adaptive Resolution Representation The problem with the FR representation is that it uses the same resolution to represent the entire shape. There are certain portions of the shape where low resolution is sufficient i.e. increasing the resolution does not improve the quality of the representation for these portions of the shape (e.g., the lower rectangular portion of the shape in Figures 1(a) and (b)). Using high resolution for these regions increases the number of dimensions without improving the query accuracy. On the other hand, there are portions of the shape (e.g., the upper portion of the shape in Figures 1(a) and (b)) where higher resolution improves the quality of the representation significantly. For these regions, the improvement in query accuracy obtained by using high resolution is worth the extra cost of having more dimensions. Also, high resolution is usually not necessary for the interior of a shape but is important near the border of the shape. Having the same resolution for the entire shape is wasteful in terms of the number of dimensions used to represent the shape and hence the query cost.

To overcome the shortcomings of the FR representation, we propose an *adaptive resolution* (AR) representation of shapes i.e. a representation where the resolution of the grid cells varies from one portion of the shape to another, having higher resolution where it improves the quality of representation and lower resolution where it does not. An adaptive representation of the same shape is shown in Figure 1(c). It uses 16 grid cells to represent the entire shape but the cells have different resolutions (i.e. sizes). The cells in the lower portion and interior of the shape have lower resolution (i.e. larger size) while those in the upper portion and near the borders have higher resolution (i.e. smaller size). Figure 1(d) shows another adaptive representation of the same shape with 32 grid cells. Note that as the number of cells increases, more cells are added to the portion of the shape where higher resolution is required while the other portions remain unchanged.

Computing AR representation Using Quadtree Decomposition We compute the AR representation of a shape by applying quadtree decomposition on the bitmap representation of the shape. The bitmap is constructed in the same way as the FR approach discussed in [5] (cf. Section 2). We use the highest resolution bitmap for the decomposition (i.e. each grid cell is 1 $\times$ 1 pixels) but lower resolution bitmaps could be used as well. The decomposition is based on successive subdivision of the bitmap into four equal-size quadrants. If a bitmap-quadrant does not consist entirely of 1s or entirely of 0s (i.e. the shape "partially covers" the quadrant), it is recursively subdivided into smaller and smaller quadrants until we reach bitmap-quadrants, possibly 1 $\times$ 1 pixels in size, that consist entirely of 1s or en-

```
Distance(int *shape1, int *shape2)

i=0, j=0;
while ($i < n$ AND $j < n$)
    if $i$th rect r1 of shape1 overlaps with $j$th rect r2 of shape2
        common_area += Overlap_Area(r1, r2);
        if (r1 is bigger than r2) j++;
        else i++;
    else // r1 and r2 do not overlap
        if (ZValue(r1) > ZValue(r2)) j++;
        else i++;
area_of_shape1 = $\Sigma_{i=0}^{(n-1)}$ Area of $i$th rectangle;
area_of_shape2 = $\Sigma_{i=0}^{(n-1)}$ Area of $i$th rectangle;
union_area = area_of_shape1 + area_of_shape2 - common_area;
distance = 1 - $\frac{common\_area}{union\_area}$;
return distance;
```

Table 1: Computing distance between two shapes.

tirely of 0s (termination condition of the recursion). Figure 2(a) shows a 8×8 bitmap of the shape in Figure 1 and Figure 2(b) shows the quadtree decomposition of the bitmap. Each node in the quadtree covers a rectangular (always square) region of the bitmap. The level of the node in the quad tree determines the size of the rectangle. The internal nodes (shown by gray circles) represent "partially covered" regions, the leaf nodes shown by white boxes represent regions with all 0s while the leaf nodes shown by black boxes represent regions with all 1s. The "all 1" regions are used to represent the shape. Figure 2(b) has 16 such rectangular regions and the shape represented is shown in Figure 2(c). Since we perform the quadtree decomposition on the 256×256 bitmap, the number of black leaf nodes in the quadtree is usually far more than the number n of rectangles we want to choose to represent the shape. In that case, we choose the n largest rectangles i.e. we do not choose any black leaf node at level i unless we have chosen all the black leaf nodes at level $j, j < i$ where the levels are numbered in increasing order from top to bottom. In this way, we can cover the bulk of the shape with a few rectangles (i.e. small values of n) and add details to the shape as we add more rectangles (i.e. larger values of n).

Indexing After the n rectangles are chosen, we sort them based on z-order. The number sequence assigned to the black leaf nodes of the quad tree in Figure 2(b) represent the z-order. The same numbers are shown on the corresponding rectangles in Figure 2(c). Note that the z-ordering is simply a left-to-right ordering of the n selected black leaf nodes in the quad tree. We represent the shape as a *sequence* of the n rectangles. Since the rectangles are always squares, we can describe each rectangles by 3 numbers: its center $C = (C_x, C_y)$ and its size (i.e. side length) S. We represent the shape as a sequence of $3n$ numbers where $3i$, $3i+1$ and $3i+2$ numbers represent the C_x, C_y and S of the ith rectangle ($0 \leq i \leq (n-1)$) in the n-sequence. We have thus mapped each shape to a point in $3n$-dimensional space. We can now index the shapes using a $3n$-dimensional index structure. The choice of n depends on the desired dimensionality d of the index structure i.e. $n = \frac{d}{3}$.

Executing Similarity Queries Using Multidimensional Index Structure To support similarity queries, we must first choose a distance measure between shapes. We choose the popular "area difference" distance measure previously used in [5, 4]. The area difference is the area of the regions where the two shapes do not match when they are overlaid on each other. To normalize the measure, we divide the area difference by the area covered by the two shapes together i.e. area of the union of the two shapes. To be able to answer similarity queries using a multidimensional index structure, we should be able to efficiently compute (1) the distance between two points i.e. between two shapes represented using the AR representation and (2) the minimum distance (MINDIST) between a point and a node of the multidimensional index structure [1]. Once we can compute the above distances, we can answer a similarity query by executing the k-NN algorithm on the multidimensional index structure [3]. The algorithm works as follows. It maintains the nodes and objects of the index structure in the priority queue in increasing order of their distances from the query and uses the queue to traverse the tree in the same order. At each step, it pops the item from the top of the queue: if it is an object, it is added to the result list, if it is a node, it computes, using the above distance functions, the distance of each of its children from the query and pushes it back to the queue. The algorithm stops when the result set contains k objects. We first present the function to compute the distance between two points. Given two shapes $s1$ and $s2$ consisting of n rectangles each (i.e. represented as $3n$-dimensional points), a naive way to compute the distance is to compare all pairs of rectangles and compute the distance between them. This approach is computationally expensive ($O(n^2)$). We exploit the following properties of Z-ordering to compute the distance in $O(n)$ time. Let $r1$ and $r2$ be two rectangles of $s1$ and $s2$ respectively. First, if $r1$ and $r2$ do not overlap with each other and $ZValue(r1) > ZValue(r2)$ and $r1'$ is a rectangle in $s1$ that appears after $r1$ (i.e. $ZValue(r1') > ZValue(r1)$), then $r1'$ and $r2$ do not overlap with each other. Second, if $r1$ and $r2$ overlap with each other and $r1$ is larger than $r2$ (i.e. $r1$ is totally covers $r2$) and $r1'$ is a rectangle in $s1$ that appears after $r1$ (i.e. $ZValue(r1') > ZValue(r1)$), then $r1'$ and $r2$ do not overlap with each other (see [2] for proofs). The procedure to compute the distance is shown in Table 1. We have also designed an algorithm to compute the MINDIST i.e. the minimum distance between a point and node of the index structure. We do not describe the algorithm here due to space limitations but can be found in [2]. We can now answer similarity queries efficiently by executing it as a k-NN query on the multidimensional index structure.

Optimization The quality of the representation (and hence the accuracy of the answers) increases with the number of rectangles, but so does the dimensionality and hence the query cost. We present an optimization that increases the number of rectangles without increasing the number of dimensions. We merge rectangles together if they are (1) "mergeable" i.e. produce a rectangle when merged and (2) appear consecutively in the z-ordered sequence. Figure 2(d) shows the set of rectangles in Figure 2(c) after the merging. Since this representation is more compact (i.e. we can represent the same shape with less number of rectangles), for a given choice of dimensionality d, we can represent the shape more accurately. The merging based on z-order ensures that the distance functions described above can be used with some minor modifications. Unlike in the unmerged representation, the rectangles in the merged representation can be non-squares; hence, we need to store two sizes S_x and S_y for each rectangle instead of one. To represent n rectangles, we need $4n$ numbers instead of $3n$. For a desired dimensionality d, the number of rectangles to choose to represent the shape is $\frac{d}{4}$ instead of $\frac{d}{3}$.

4. EXPERIMENTS

We conducted several experiments to evaluate the effectiveness of the AR representation and compare it with the FR representation. For our experiments, we

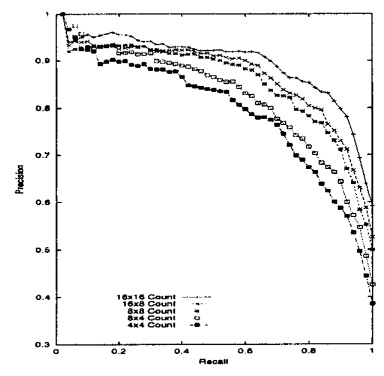


Figure 3: Precision-recall graph for FR technique.

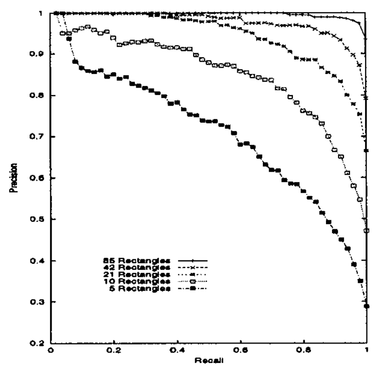


Figure 4: Precision-recall graph for AR technique.

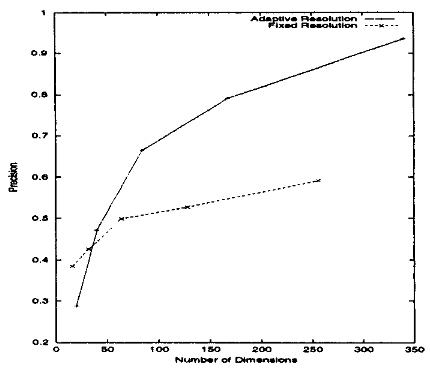


Figure 5: Comparison of FR and AR techniques at 100% recall.

used the "islands" file in the polygon dataset of the Sequoia benchmark (available online at http://s2k-ftp.cs.berkeley.edu:8000/sequoia/benchmark/polygon/). The dataset contains 21,021 shapes. For a given query, we generate the ground truth by executing the query against the highest resolution bitmap representation of the shape (i.e. each grid cell is 1 × 1 pixel) and retrieve the top k answers. We refer to these answers as the *relevant set*. We then execute the query against a given low resolution FR representation using the index structure and retrieve the top k answers. We refer to these answers as the *retrieved set*. We compare the relevant and retrieved sets for various values of *k* (we vary k from 10 to 100) and we plot the precision (defined as $\frac{|relevant \cap retrieved|}{|retrieved|}$) and recall (defined as $\frac{|relevant \cap retrieved|}{|relevant|}$) graphs for various resolutions (4 × 4, 8 × 4, 8 × 8, 16 × 8 and 16 × 16). The result is shown in Figure 3. All the measurements are averaged over 100 queries. The graph shows that the quality of the answers improves as the resolution increases. We repeat the above experiment for the AR representation. The result is shown in Figure 4. The graph shows that the quality of the answers improves with the increase in the number of rectangles. Note that we are doubling the number of dimensions at each step in both cases but the improvement in the quality of answers at each step is more significant in the AR technique compared to the FR technique. The reason is that in the AR case, the additional rectangles concentrate on improving the representation *only* where it is necessary. On the other hand, in the FR case, the resolution is increased equally all over, as much in the unnecessary portions as in the necessary ones, thus diluting the effect and not improving the quality of representation as much as in AR case.

We compare the two techniques in terms of the quality of retrieval when the same number of dimensions are used to represent the shape. For a given recall, we compute the precision of the two techniques at a given dimensionality. Note that for the FR presentation of $n1 \times n2$ resolution, the dimensionality is $n1.n2$ while for the AR approach with n rectangles, the dimensionality is $4n$. In Figure 5, we plot the precision at 100% recall for various dimensionalities for both techniques. The AR technique significantly outperforms the FR technique in terms of precision at almost all dimensionalities. For example, at 100 dimensions, the AR technique has about 70% precision while the FR technique has about 50% precision. We observed similar behaviour at other values of recall. This shows that AR is a more compact representation i.e. with the same number of dimensions, AR approximates the original shape better than FR and hence provides higher precision. Assuming that the query cost is proportional to the number of dimensions used[1], the AR technique provides significantly better quality answers at the same cost and is hence a better approach to shape retrieval.

[1]We have performed experiments using a multidimensional index structure and measured the actual I/O and CPU cost for the two techniques. Our experiments validate the claim that the query cost is indeed proportional (actually super-linearly) to the number of dimensions used. We do not present those results here due to space limitations but can be found in the full version of the paper [2].

5. CONCLUSION

Similar shape retrieval is an important problem with a wide range of applications. In this paper, we have presented a novel adaptive resolution approach to representing 2-d shapes. The representation is invariant to scale, translation and rotation. With the proposed representation, we can index the shapes using a multidimensional index structure and can thus support efficient similarity retrieval. Our experiments show that, for the same query cost, the adaptive technique provides significantly better quality answers compared to the fixed resolution representation and is hence a better approach to shape retrieval.

6. REFERENCES

[1] K. Chakrabarti and S. Mehrotra. The hybrid tree: An index structure for high dimensional feature spaces. *Proceedings of the IEEE International Conference on Data Engineering*, March 1999.

[2] K. Chakrabarti, M. Ortega-Binderberger, K. Porkaew, P. Zuo, and S. Mehrotra. Similar shape retrieval in mars. *Technical Report, TR-MARS-00-03, University of California at Irvine*, 1999.

[3] G. R. Hjaltason and H. Samet. Ranking in spatial databases. *Proceedings of SSD*, 1995.

[4] H. Jagadish. A retrieval technique for similar shapes. *Proceedings of SIGMOD*, 1991.

[5] G. Lu and A. Sajjanhar. Region-based shape representation and similarity measure suitable for content-based image retrieval. *Springer Verlag Multimedia Systems*, 1999.

[6] R. Mehrotra and J. Gary. Similar shape retrieval in shape data management. *IEEE Computer*, 1995.

[7] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, and P. Yanker. The QBIC project: Querying images by content using color, texture and shape. In *Proc. of SPIE Conference on Storage and Retrieval for Image and Video Databases*, February 1993.

[8] M. Ortega-Binderberger, Y. Rui, K.Chakrabarti, S. Mehrotra, and T. Huang. Supporting ranked boolean similarity queries in mars. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, November 1998.

[9] Y. Rui, A. She, and T. Huang. Modified fourier descriptors for shape representation – a practical approach. *Proceedings of 1st Int. Workshop on Image Databases and Multi Media Search*, 1996.