Multiscale Similarity Matching for Subimage Queries of Arbitrary Size

Kai-Sang Leung *and* *Raymond Ng**
Department of Computer Science, The University of British Columbia
2366 Main Mall, Vancouver BC, Canada V6T 1Z4

Abstract

Many image database management systems support whole-image queries. However, in some situations, users may only remember certain portions of the images. In this paper, we develop Padding and Reduction Algorithms to support subimage queries of arbitrary size based on local color information. The idea is to estimate the best-case lower bound to the distance between the query and the image. To improve the efficiency and effectiveness of content-based retrieval, a multiscale representation is proposed. Since image contents are usually pre-extracted and stored, the number of levels used in such a representation needs to be determined. We address this issue analytically by estimating the CPU and I/O costs, and experimentally by comparing the performance and accuracy of the outcomes of various filtering schemes. Our findings suggest that a 3-level hierarchy is preferred.

We also study three strategies for searching multiple scales. Our studies indicate that the hybrid strategy with horizontal filtering on the coarse level and vertical filtering on remaining levels is the best choice. When used with Padding and Reduction Algorithms in the preferred 3-level multiscale representation, desired images can be retrieved efficiently and effectively.

Keywords

Similarity matching, subimage querying, multiscale representation, cost model, search strategy

1 INTRODUCTION

Over the past few decades, database management systems have been recognized as tools with practical value for handling large volumes of data. Exist-

*Person handling correspondence: Raymond Ng. Telephone: +1(604)822-2394. Fax: +1(604)822-5485. E-mail: rng@cs.ubc.ca

Visual Database Systems 4 Y. Ioannidis & W. Klas (Eds.)

ing systems were mainly designed to provide environments for efficient and effective retrieval and storage of only alphanumeric data. In recent years, advances in technologies for scanning, networking, and CD-ROMs have lowered the prices for disk storage. These advances, coupled with the acceptance of common image compression and file formats, enable users to acquire, store, manipulate and transmit large numbers of visual data in the form of images. In addition, images are generated at an increasing rate by an ever-growing number of sources. For example, an estimated 281 gigabytes of image data will be produced daily by the instruments on two NASA's Earth Observing Systems platforms (Castelli *et al.* 1997). As a result, the number of digital image archives has increased tremendously. In the 1992 US National Science Foundation workshop on visual information management systems (Jain 1993), four 'Grand Challenge' application domains were identified; image databases play important roles in these application domains. The demands for image database management systems (IDBMS's) also increase in other application areas like World-Wide Web publications, retail cataloging, art work retrievals, advertisement creation, and imaging clip arts (Barber *et al.* 1994, Gudivada and Raghavan 1995, Petkovic *et al.* 1996). Given this explosive growth in the availability and demand for image data, we can no longer rely solely on traditional database retrieval technology based on manually associating textual descriptions with image contents. The development of efficient and effective content-based retrieval systems based on automated extracted contents (such as color) is necessary.

To achieve this goal, several research projects have been carried out. Over the past few years, multi-dimensional indexing structures (Guttman 1984, Samet 1990, Lin *et al.* 1994) have been designed, multi-level filtering approaches (Sawhney and Hafner 1993, Ng and Tam 1997) have been proposed, and information preserving transformations (Faloutsos *et al.* 1994, Thomasian *et al.* 1997) have been suggested for providing efficient indexing. Expressive query language systems (Flickner *et al.* 1995, Bach *et al.* 1996, Faulus and Ng 1997) have also been developed to accommodate efficient querying from image databases for applications with dense and sparse image spaces. Other research projects for supporting efficient and effective query processing and optimization have also been carried out.

With the popularity of similarity matching on color and spatial information, many IDBMS's store the information in local color histograms. To process user queries, some systems use fixed grid segmentation approaches. For further improvement on the efficiency and the effectiveness of content-based retrieval, multiscale matching approaches (Jacobs *et al.* 1995, Chen *et al.* 1997) have been proposed. However, detailed analytical and experimental results on the determination of the suitable number of levels for these approaches are seldom reported, and comparisons of different strategies for searching multiple scales are rare. Moreover, in many application areas, users are interested in only local image contents; therefore, subimage query processing is needed. Unfortunately, not many IDBMS's can handle arbitrary-size subimage queries based on color and spatial similarity. For the systems that can deal with subimage queries of arbitrary size, multiscale matching is rarely used. Therefore, in this paper, we study the following three questions:

1. How can we process subimage queries of arbitrary size ?

   Many existing IDBMS's (*e.g.*, the system developed by Jacobs *et al.* (1995)) support only whole-image queries (which specify the contents of the whole images to be retrieved), but not subimage queries. Due to the poor human memory capability for retaining a fine granularity of spatial information of color, users typically cannot recall very many details of images they have seen before. To pose a whole-image query, users need to come up with the color information on the remaining portion of the images which they may not know or care about. Hence, methods for processing arbitrary-size subimage queries are needed. To answer subimage queries of this kind, we develop two algorithms, called Padding and Reduction. The idea is to estimate the best-case lower bound to the histogram distance between the subimage query and the image.
2. How many levels of the multiscale representation do we need ?

   To provide better selectivity for image retrievals based on color and spatial information, many IDBMS's (*e.g.*, QBIC (Flickner *et al.* 1995)) segment an image into blocks of a certain size. However, the efficiency and the effectiveness of image retrievals depend on the quality of the segmentation and the relevance of the segmentation to the user's needs. For some queries, the scale at which the images are segmented may be too fine; for other queries, such a scale may be too coarse. The difficulty of picking one best scale in which the image should be segmented can be addressed by the use of multiscale representation. Since image contents are usually pre-extracted and stored, we need to determine a suitable number of levels for the multiscale representation. Analytically, we use a cost model to estimate CPU and I/O costs. Experimentally, we measure the performance and the accuracy of the outcomes of strategies for searching the representation. Our findings suggest that a 3-level hierarchy is preferred.
3. What is a good strategy for searching multiple scales ?

   Some matching strategies have been proposed for searching multiple scales. One of these strategies is branch-and-bound (Chen *et al.* 1997). Due to its effectiveness for handling whole-image queries, the strategy can be adapted to handle subimage queries. However, at each iteration, jumping back and forth in the data file to get the necessary data (histograms) for computation seems unavoidable. Such jumping may impose a high I/O cost. As a result, it can be inefficient for large databases. In this paper, we study three strategies that try to avoid this kind of jumping. To find an efficient and effective multiscale searching strategy among the three strategies, analytical and experimental evaluations are conducted. The results indicate that the hybrid strategy with horizontal filtering on the coarse level and vertical filtering on the remaining levels is the best when operated with Padding and Reduction Algorithms. In addition to avoiding the frequent jumping incurred in branch-and-bound, the strategy also maintains a good balance between efficiency and effectiveness.

In the next section, we present Padding and Reduction Algorithms. Concepts, implementations, and experimental results are discussed. Section 3 describes the multiscale representation and a strategy for searching such a representation (Pure Vertical search). We also show the analytical and experimental results related to the second question mentioned above. In Section 4, two more search strategies, namely Pure Horizontal and Horizontal-and-Vertical, are discussed. We compare both analytical and experimental results of the three search strategies so as to determine the best strategy for searching the multiple scales. Finally, conclusions and suggestions for future work are presented in Section 5.

2 PADDING AND REDUCTION ALGORITHMS

For the IDBMS's that support whole-image queries, color is usually extracted automatically and stored in feature vectors (*e.g.*, n-dimensional color histograms). Once these vectors are created for the images in the database, the similarity between the whole-image query and the image can be computed based on a variety of distance measures. Example of these measures are statistical distribution features (Stricker and Dimai 1996), histogram intersection (Swain and Ballard 1991), and (weighted) Euclidean distance (Flickner *et al.* 1995). Similarity matching methods based on statistical distribution features were proposed on the assumption that query objects are usually located in the center of images. Unfortunately, this assumption may not hold in some applications. For example, the user may only be interested in images where the sun is rising in the upper right portion of the image. Histogram intersection is robust for computing similarity between the whole-image query and the image. However, the feasibility of using such a measure in subimage query matching is unclear. As such, in our discussion, we focus on the Euclidean measure. Results given in this section can be generalized to other measures such as weighted Euclidean.

To answer the subimage query of arbitrary size, some systems (*e.g.*, QBIC) segment an image into several blocks, each of which has an associated color histogram. One problem of this arrangement is that subimage queries may be of arbitrary size that need not be an integral multiple of the chosen block size. Other systems use template-based matching algorithms. A key problem with those algorithms is that a huge amount of computations is needed, because of the large number of positions to be compared. Therefore, other algorithms for efficient and effective processing of arbitrary-size subimage queries are needed.

To exploit the existing advantages of the Euclidean measure for computing similarity between the whole-image query and the image, we can use such a measure in computing similarity between the subimage query and the image. However, given a subimage query Q of arbitrary size, it may not necessarily be of the same size as the precomputed image block I. Without loss of generality, we let the subimage query Q be made up of v pixels and the image block I be made up of w pixels (where $w \geq v$). Due to the quadratic nature of the Euclidean measure, comparing the histograms representing these Q and I may seem unfair. Excessive pixels in one histogram may dramatically influence the resulting histogram distance. Alternatively, we may normalize the histograms for Q and I. Unfortunately, the use of normalized histograms with the above Euclidean measure may not be helpful due to various side-effects (such as scaling and zooming), especially in application areas where the user is confident in the query size. For example, the user may be interested in finding images with 'the Union Jack in upper left portion' (such as the Australian flag). Using the normalized histograms may have an effect of scaling up this query into the query of finding images with 'the Union Jack on the entire region'. As a result, the flag of UK is returned instead of the Australian flag.

To avoid the problem caused by the size differential between Q and I, we propose two approaches — Padding and Reduction — for processing subimage queries of arbitrary size. Given a subimage query Q and an image block I of a size larger than Q, the idea is to estimate the best possible color histograms by either (i) enlarging Q into a new query Q′ that is of the same size as I, or (ii) reducing I to a new image block I′ that is of the same size as Q (Figure 1). More precisely, the two proposed approaches estimate the histogram distance between the subimage query Q and the image block I.

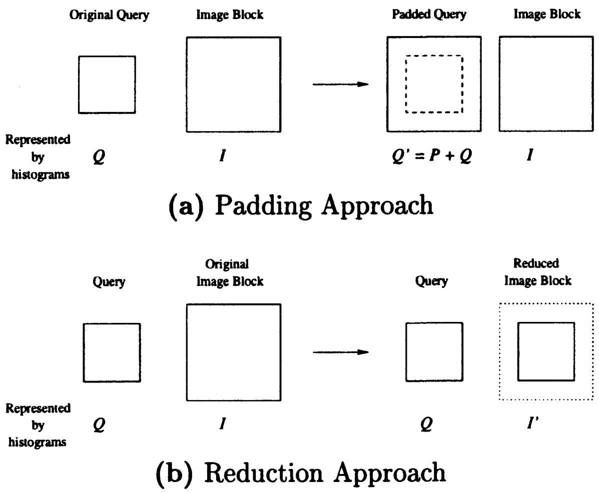


Figure 1 Padding and Reduction Approaches.

2.1 Padding Algorithm

For the Padding Approach, we enlarge the subimage query Q by padding $w-v$ 'desired' pixels to it so that the resulting padded query Q′ is of the same size as the image block I. In order to minimize the estimated histogram distance $\widehat{D_H}$, the 'desired' pixels are chosen from the image block.

Definition 1 Let I, P, Q and $Q'(= P+Q)$ represent the n-dimensional color histograms of the image block, the padded area, the original subimage query, and the resulting padded query respectively. We let $w, w-v, v$ and w be the corresponding number of pixels represented in the histograms I, P, Q and Q'. The goal of the Padding Approach is to find an appropriate assignment to the optimization variable P so that $\widehat{D_H}$ is minimized and the vector inequality $P \leq I$ is met:

$$\text{Given } \sum_{j=1}^{n} Q_j = v \text{ and } \sum_{j=1}^{n} I_j = w \geq v,$$

we want to find the optimal vector P to

objective function $$\min_P (P-\alpha)^T(P-\alpha) \equiv \min_P \sum_{j=1}^{n}(P_j-\alpha_j)^2 \quad (1)$$

inequality constraint subject to $0 \leq P_j \leq I_j$ for $1 \leq j \leq n$

summation constraint and $\sum_{j=1}^{n} P_j = w - v$

domain constraint and $I, P, \alpha(= I - Q)$ integer vectors

The estimated best-case lower bound $\widehat{D_H} = (P+Q-I)^T(P+Q-I) = (P-\alpha)^T(P-\alpha)$. ■

To solve this minimization problem, several methods can be applied. A brute-force method is to exhaustively generate all feasible vectors for P. These generated vectors are then tested for minimality, and the one that gives the minimal Euclidean distance is returned. Solving Problem (1) with this generate-and-test method is unpalatable in the sense that the execution time is expected to be very high.

Alternatively, based on the observation that Problem (1) is an instance of quadratic programming (QP) problems, we can use existing software packages that are built for handling QP. Unfortunately, these software packages may not be helpful, because many of them compute the optimal vectors in the domains of real number (*i.e.* lacking integer programming functionality). This domain problem coupled with the roundoff error may lead to the unreliability of some answers. Therefore, efficient and effective algorithms for computing the estimated best-case lower bound to the histogram distance are needed.

Note that the objective function in Problem (1) is in the form of the sum of squares of the difference terms (d-terms):

$$\sum(P_j - \alpha_j)^2$$

In order to minimize the sum, we need to minimize the d-terms. In which order should the d-terms be minimized ? Due to the quadratic nature of the squares of the d-terms, we notice, on a close examination of the representation, that deducting 1 off a large d-term is more effective in minimizing the sum than deducting 1 off a small d-term:

$$\text{If integers } a > b \geq 1, \text{ then } (a-1)^2 + b^2 < a^2 + (b-1)^2$$

It is also clear that for two d-terms having the same value c, subtracting an integer d from each of these d-terms is more effective in minimizing the sum than subtracting $2d$ from only one of these two equal-valued d-terms:

$$\text{If integers } c > 2d \text{ and } d \geq 1, \text{ then } (c-d)^2 + (c-d)^2 < (c-2d)^2 + c^2$$

Hence, for the Padding Approach, we start with $P_j = 0$ for all j; each d-term $(P_j - \alpha_j)$ becomes $-\alpha_j$. These d-terms are then rearranged in non-ascending order of α_j (in other words, non-descending order of $-\alpha_j$) and result in:

$$\{(P_{(1)} - \alpha_{(1)}), (P_{(2)} - \alpha_{(2)}), \cdots, (P_{(n)} - \alpha_{(n)})\} \text{ where } \alpha_{(1)} \geq \alpha_{(2)} \geq \cdots \geq \alpha_{(n)}$$

After the rearrangement, we try to lower the value of the first d-term $(P_{(1)} - \alpha_{(1)})$ by adding the amount $\alpha_{(1)} - \alpha_{(2)}$ to $P_{(1)}$ so that the first d-term has the same value as the second d-term $(P_{(2)} - \alpha_{(2)})$. We then try to reduce the values of these two d-terms by increasing the values of $P_{(1)}$ and $P_{(2)}$ in round-robin fashion so as to make them have the same value as the third d-term $(P_{(3)} - \alpha_{(3)})$. We keep devaluing the first k d-terms so that they have the same value as the $(k+1)$th d-term through increases of the values of $P_{(1 \leq j \leq k)}$. This process is repeated until the summation constraint $\sum_{j=1}^{n} P_j = w - v$ is satisfied. At any cycle (say, Cycle k), the value of $P_{(1 \leq j \leq k)}$ is constrained by the inequality $P_{(j)} \leq I_{(j)}$, and will not be increased beyond its allowable maximum $I_{(j)}$.

Algorithm 2 (The Padding Algorithm)

```
1  $\forall j,\ \alpha_j \leftarrow I_j - Q_j$
2  $\forall j,\ P_j \leftarrow 0$
3  $\{(P_{(j)} - \alpha_{(j)})\text{'s}\} \leftarrow$ sort the $(P_j - \alpha_j)$ terms in non-ascending order of $\alpha_j$
4  $k \leftarrow 1$
5  while $k < n$ and $\sum_{j=1}^{k} P_{(j)} < w - v$ do
```

```
6        loop for at most $\alpha_{(k)} - \alpha_{(k+1)}$ cycles
7            for each $P_{(1\le t\le k)}$ do
8                if $P_{(t)} < I_{(t)}$ then $P_{(t)} \leftarrow P_{(t)} + 1$
9                if $\sum_{j=1}^{k} P_{(j)} = w - v$ then return $\langle P, \sum(P_j - \alpha_j)^2\rangle$
10       $k \leftarrow k + 1$
11   if $\sum_{j=1}^{n} P_{(j)} < w - v$
12       then loop
13           for each $P_{(1\le t\le n)}$ do
14               if $P_{(t)} < I_{(t)}$ then $P_{(t)} \leftarrow P_{(t)} + 1$
15               if $\sum_{j=1}^{n} P_{(j)} = w - v$ then return $\langle P, \sum(P_j - \alpha_j)^2\rangle$   ■
```

Example 3 Given 3-dimensional integer vectors $I = (\ 12, 3, 10\ )^T$ and $Q = (\ 2, 7, 1\ )^T$; then, $\alpha = I - Q = (\ 10, -4, 9\ )^T$. We want to find an appropriate assignment to integer vector $(\ P_1, P_2, P_3\ )^T$ so that the objective function $(P_1-10)^2+(P_2+4)^2+(P_3-9)^2$ is minimized and the constraints $(\ 0, 0, 0\ )^T \le (\ P_1, P_2, P_3\ )^T \le (\ 12, 3, 10\ )^T$ and $\sum_{j=1}^{3} P_j = 25 - 10 = 15$ are satisfied.

We start with $P_j = 0$ for all j and rearrange all the d-terms. After the rearrangement, we try to lower the value of the first d-term ($P_{(1)} - 10 = -10$) by adding 1 to $P_{(1)}$ so that the first two d-terms have same value ($= -9$). Then, we try to reduce the values of these two d-terms by increasing the values of $P_{(1)}$ and $P_{(2)}$ in round-robin fashion so as to bring them closer to the value of the third d-term ($= 4$); we stop when the constraint $\sum_{j=1}^{3} P_j = 15$ is satisfied.

	$P_{(1)} - \alpha_{(1)}$ $= P_1 - \alpha_1$	$P_{(2)} - \alpha_{(2)}$ $= P_3 - \alpha_3$	$P_{(3)} - \alpha_{(3)}$ $= P_2 - \alpha_2$	$\sum P_j$
Cycle 0	$0 - 10 = -10$	$0 - 9 = -9$	$0 - (-4) = 4$	0
Cycle 1	$1 - 10 = -9$			1
Cycle 2	$2 - 10 = -8$	$1 - 9 = -8$		3
Cycle 3	$3 - 10 = -7$	$2 - 9 = -7$		5
⋮	⋮	⋮		⋮
Cycle 8	$8 - 10 = -2$	$7 - 9 = -2$		15

The Padding Algorithm returns $\langle P, \widehat{D_H}\rangle = \langle\ (\ 8, 0, 7\ )^T, 24\ \rangle$. ■

2.2 Reduction Algorithm

Figure 1 shows the similarities and difficulties of the Padding Approach and the Reduction Approach. As notice from the figure, for the Reduction Approach, we reduce the precomputed image block I by choosing v 'desired' pixels from it so that the resulting reduced image block I′ is of the same size as the subimage query Q. In other words, the $w - v$ 'not so desired' pixels are removed.

Definition 4 Let I, I' and Q represent the n-dimensional color histograms of the original precomputed image block, the resulting reduced image block, and the subimage query respectively. We let w, v and v be the corresponding number of pixels represented in the histograms I, I' and Q. The goal of the Reduction Approach is to find an appropriate assignment to the optimization variable I' so that $\widehat{D_H}$ is minimized and the vector inequality $I' \leq I$ is met:

$$\text{Given } \sum_{j=1}^{n} Q_j = v \text{ and } \sum_{j=1}^{n} I_j = w \geq v,$$

$$\text{we want to find the optimal vector } I' \text{ to}$$

$$\textit{objective function} \qquad \min_{I'} (Q - I')^T (Q - I') \equiv \min_{I'} \sum_{j=1}^{n} (I'_j - Q_j)^2 \qquad (2)$$

$$\textit{inequality constraint} \qquad \text{subject to } 0 \leq I'_j \leq I_j \text{ for } 1 \leq j \leq n$$

$$\textit{summation constraint} \qquad \text{and } \sum_{j=1}^{n} I'_j = v$$

$$\textit{domain constraint} \qquad \text{and } I, I', Q \text{ integer vectors}$$

The estimated best-case lower bound $\widehat{D_H} = (Q - I')^T (Q - I')$. ■

As we can see, Problem (2) is similar to Problem (1). The Algorithm 2 has been adapted for the Reduction Approach by replacing (i) P with I', (ii) α with Q, and (iii) the summation constraint with $\sum_{j=1}^{n} I'_{(j)} = v$. The resulting algorithm is the Reduction Algorithm.

2.3 Evaluation

Having developed the Padding Algorithm and the Reduction Algorithm, we would like to ask the question: In terms of effectiveness, which one produces a better lower bound ? To answer this question, we let:

- $F^P = \{ f^P \mid f^P = (P+Q-I)^T (P+Q-I)$ where P satisfies the constraints: (i) $0 \leq P_j \leq I_j$ for $1 \leq j \leq n$ and (ii) $\sum_{j=1}^{n} P_j = w - v \}$, and
- $F^R = \{ f^R \mid f^R = (Q - I')^T (Q - I')$ where I' satisfies the constraints: (i) $0 \leq I'_j \leq I_j$ for $1 \leq j \leq n$ and (ii) $\sum_{j=1}^{n} I'_j = v \}$,

where $\sum_{j=1}^{n} Q_j = v$ and $\sum_{j=1}^{n} I_j = w \geq v$. The analytical results show that in the domain of integers, there is a 1-to-1 correspondence between F^P and F^R. In other words, for any feasible solution f^P to Problem (1), there is a corresponding solution f^R to Problem (2) such that $f^P = f^R$. Similarly, for any feasible solution f^R to Problem (2), there is a corresponding solution f^P to Problem (1) such that $f^P = f^R$. The $\widehat{D_H}$ values returned by the Padding Algorithm and the Reduction Algorithm are the minimal f^P and the minimal f^R respectively. Therefore, in terms of effectiveness, the two Algorithms give the same best-case lower bound to the histogram distance.

Lemma 5 If a subimage query Q is contained in an image block I, then both the Padding Algorithm and the Reduction Algorithm return the estimated histogram distance $\widehat{D_H} = 0$. ■

The next question is: In terms of efficiency, which one produces the lower bound faster ? The experimental results show that the performance of the two Algorithms differ significantly depending on the size differential between the subimage query and the image block (Figure 2). Given an image block I consisting of w pixels and a subimage query Q consisting of v pixels (where $v \leq w$):

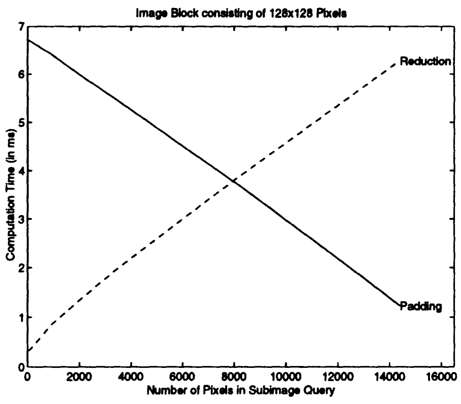


Figure 2 Computation time for Padding and Reduction Algorithms.

- For small size differential (*e.g.*, $w < 2v$): The Reduction Algorithm picks v pixels to form I′, whereas the Padding Algorithm pads $w - v < v$ pixels to Q to form Q′.
- For large size differential (*e.g.*, $w > 2v$): The Padding Algorithm pads $w - v$

pixels to form Q′, whereas the Reduction Algorithm picks $v < w - v$ pixels from I to form I′.

- For medium size differential (*e.g.*, $w \approx 2v$): The Padding Algorithm pads $w - v$ pixels to form Q′, and the Reduction Algorithm picks $v \approx w - v$ pixels to form I′.

Therefore, in terms of efficiency, the Padding Algorithm outperforms the Reduction Algorithm when size differential between Q and I is small, and *vice versa* when the differential is large.

As we can see, the complexity for the Padding Algorithm is $O(w - v)$, and the complexity for the Reduction Algorithm is $O(v)$. Since the two Algorithms give the same best-case lower bound, we can choose the faster Algorithm for the given Q and I. Hence, the average complexity for Padding and Reduction Algorithms is $O(\min(w - v, v))$.

3 MULTISCALE REPRESENTATION

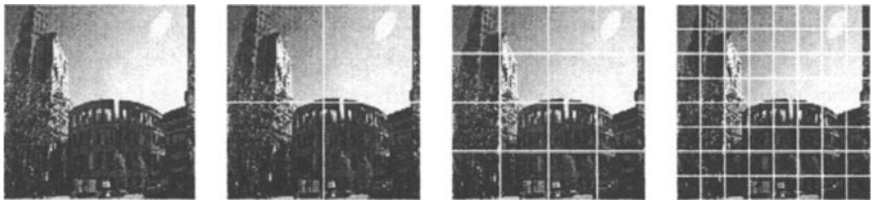
As discussed above, some IDBMS's divide an image into blocks of a certain size. For some queries, the scale at which the images are blocked may be too fine. Applying similarity comparisons to all those fine blocks may be a waste of effort. However, for some other queries, the scale may be too coarse. The desired images in the database may not be discriminated sufficiently. Given that subimage queries can be of arbitrary size, picking one best scale for all queries is hard, if not impossible.

One solution is to have multiple scales/resolutions for matching. The idea is that depending on the scale or need of the given query, a more appropriate scale can be used. The wavelet decomposition method developed by Jacobs *et al.* is a good example of a multiscale scheme, in which the coefficients of the decompositions are distilled through processes of truncation and quantization. During query processing, the algorithm simply compares the number of distilled coefficients that are common to both the query and the image. However, the method is applicable only to whole-image queries; it is not clear how subimage queries can be supported.

Chen *et al.* (1997) proposed a branch-and-bound algorithm for searching multiple scales. While we shall discuss their approach in greater detail later, it suffices to say that in Chen *et al.*'s study and in most of the related studies in the literature, there is little discussion on how to determine the number of levels in the multiscale representation. Analytical or experimental evaluation of this issue is almost non-existent.

3.1 Multiscale Similarity Matching

Here, we study a 4-level multiscale representation in which the entire image is divided into four blocks, and each block is recursively divided into four, and so on (Figure 3):



(a) Level H (b) Level I (c) Level J (d) Level K

Figure 3 The 4-level multiscale representation.

1. At Level H, the entire image is represented by a single color histogram.
2. At Level I, the image is divided into four non-overlapping blocks, and each block is represented by a color histogram covering $\frac{1}{4}$ of the entire image.
3. At Level J, each block at Level I is further divided into four blocks, each of which is represented by a color histogram covering $\frac{1}{16}$ of the entire image.
4. At Level K, each block at Level J is again divided into four, each of which is represented by a color histogram covering $\frac{1}{64}$ of the entire image.

With this multiscale representation, given any subimage query Q of arbitrary size, there exists an image block I whose size is not smaller than Q. So, Padding and Reduction Algorithms can be applied in similarity matching. Depending on the location of image block I, similarity matching can incorporate both color similarity and spatial similarity. For instance, the distance between Q and I can be a weighted sum of the form: $\beta \times$ histogram distance + $(1-\beta) \times$ positional distance, where the histogram distance is computed by either the Padding Algorithm or the Reduction Algorithm. The weighting factor β can be chosen by the user.

Due to the 'minimization' nature of the estimated histogram distance, the distance function has been formulated in a such way that given Q and I, the distance at the coarser scale can serve as a lower bound to the distance at the finer scale. With this property, efficient multiscale search strategies with the use of vertical filtering can be explored. Here, we study one of these strategies, namely PV (Pure Vertical); we shall consider other search strategies in the next section. With PV, all database images are checked one after another. At any point in time, we keep the current u smallest distances (images at their finest scales), where u is the number of images requested by the user. For each image, we keep proceeding to finer scales until (i) the distance value at a particular scale is already so large that the image cannot be qualified as a good match, or (ii) the finest scale is reached and the image is either discarded or selected as a member of the answer set, depending on the distance value.

With the four levels of image blocks in the representation, many multi-level filtering schemes are possible. For example, the choices include a four-level HIJK scheme, a two-level HI scheme, a two-level IK scheme (skipping Level J), and some other schemes. Depending on the efficiency and effectiveness of various schemes and the size of subimage query, appropriate filtering schemes can be chosen.

3.2 Analytical Evaluation

In order to determine the suitable number of levels required for the multiscale representation, we have conducted an analytical evaluation by setting up a cost model. Our model captures:

- CPU cost, which depends mainly on the time required to apply Padding and Reduction Algorithms and the number of times in which the Algorithms are applied.
- I/O cost, which depends mainly on the time required to sequentially and randomly access the pages containing the data (the histograms). This access time can be affected by page size and buffer size.

Some optimization techniques can be applied to reduce the CPU and I/O costs. As an example, it is observed that for a vast majority of subimage queries, if the image blocks I^{sub} and I^{sup} are best matches (which give the smallest distance values) at their corresponding levels (*e.g.*, Levels I and J), then I^{sub} is one of the subblock enclosed in I^{sup}. So, given a subimage query of size smaller than $\frac{1}{16}$ of the entire image (*i.e.* smaller than the precomputed image block at Level J), the most promising image block I^{I} can be found after checking histograms of the four blocks at Level I. Then, rather than considering histograms representing all 16 blocks at Level J, we simply consider the four representing the region covered by I^{I}. By so doing, both CPU and I/O costs are reduced.

For lack of space, we only show the CPU and I/O cost formulas for a three-level HIJ filtering scheme in this section; please refer to Leung's (1997) work for the cost formulas of all other filtering schemes. To estimate the CPU cost, we let T_C be the computational time required to apply Padding and Reduction Algorithms. Then, given a query of size smaller than that of the precomputed image block at Level J, the CPU cost for the HIJ scheme using the PV strategy is $(M + 4\xi_I + 4\xi_J)T_C$, where ξ_I and ξ_J are the number of images checked at Levels I and J respectively (number of database images

$= M \geq \xi_I \geq \xi_J \geq u =$ number of requested images). The two parameters ξ's are determined dynamically during the runtime.

To estimate the I/O cost, we let the space required by each dimension of a color histogram be 4 bytes; a total of $4n$ bytes are needed for one n-dimensional histograms. Hence, the total number of pages occupied by one histogram is $\frac{4n}{P}$ pages where P is the page size. Then, the I/O cost is the sum of total seek times and total data transfer times. Using a minimal buffering (a buffer size of one page) with an appropriate file organization for the PV strategy*, the I/O cost for the HIJ scheme (using a page size of 1 kilobyte and a 64-dimensional histograms) is $T_A + (M - 1 + \frac{3\xi_J}{4})T_M + (M + \xi_I + \xi_J)T_D$, where T_A (average seek) is charged only for getting to the Level-H histogram of the first image, and T_D (data transfer) is charged for loading the page containing the histogram, and T_M (minimum seek) is charged for jumping between records.

In the analyses, we vary parameters like the dimension of the color histogram and the number of database images. The observations are listed below:

- As the dimension of the histogram grows, the time required for CPU operation and I/O operation increases. Hence, the combined CPU and I/O cost increases.
- As the number of database images becomes larger, the combined CPU and I/O cost increases.
- Filtering schemes which start with filter J or K often take more CPU and I/O time.
- Filtering schemes which skip intermediate levels often incur greater CPU and I/O costs.
- The above trends can still be observed when using various level of buffering.

Therefore, filtering schemes that start with the Level H (or start with Level I), and those that do not skip any intermediate levels are favored.

3.3 Experimental Evaluation

In addition to the analytical evaluation, we have also conducted an experimental evaluation. In the experiments, we use subimage queries of different sizes (some even smaller than $\frac{1}{64}$ of the entire image), and measure both the efficiency and effectiveness of the filtering schemes. The efficiency of a scheme can be assessed by its execution time; the effectiveness of a scheme

*Since the database images are checked one after another, an appropriate way to organize the histograms for the PV strategy is to arrange them on an image-by-image basis (*i.e.* Levels H-I-J-K histograms of Image 1 followed by Levels H-I-J-K histograms of Image 2, and so on).

can be assessed by its dissimilarity score. The dissimilarity score is computed by comparing the number of retrieved images actually falling into 'prefect-match', 'best-match', 'good-match', 'fair-match', and 'poor-match' categories to the number of retrieved images ideally falling into these categories. The lower the dissimilarity score, the better is the accuracy. The experimental results suggest that when operated with Padding and Reduction Algorithms:

- The one-level H scheme is best for 'large' subimage queries*.
- The two-level HI scheme is best for 'medium' subimage queries.
- The three-level HIJ scheme is best for both 'small' and 'tiny' subimage queries.

Therefore, desired images can be retrieved efficiently and effectively using only the top three levels (Levels H, I, and J) of the multiscale representation.

3.4 Discussion: Dealing with Queries on Boundaries

Given the above search strategy, one may ask the question: How to handle subimage queries which lie on block boundaries ? For example, the user may specify a subimage query which lies in the center region of an image block, with each of the four subblocks containing $\frac{1}{4}$ of the query. The problem here is that the query is not entirely contained in any subblock.

A heuristic to improve such a situation is to use the overlapping 9-tile partition (Figure 4). Due to the overlapping nature of such a partition, more choices are allowed for better containment. For the above example, we can pick tile #9. To change from the original non-overlapping 4-tile partition to this 9-tile partition, the search time only increases linearly. However, the storage space required for the histograms grows exponentially.

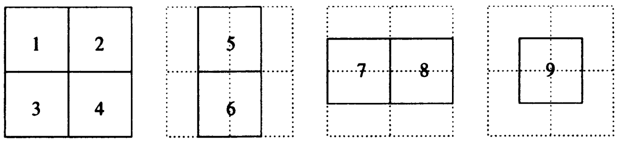


Figure 4 A overlapping 9-tile partition.

To avoid the exponential growth in storage, a simpler heuristic can be used. First, we apply multiscale similarity matching as outlined in Section 3.1 and the next section. Then, on the basis of Lemma 5 that the estimated distance

*A 'large' subimage query is the one that covers more than $\frac{1}{4}$ of the entire image. Query that is not 'large' but covers more than $\frac{1}{16}$ of the entire image is called 'medium'. A 'small' subimage query is the one that is not 'medium' but covers more than $\frac{1}{64}$ of the entire image; a 'tiny' subimage query is the one that does not belong to any of the three categories above.

at the coarsest level between a subimage query Q and its perfectly matched image I^{pm} (*e.g.*, Q is a subpart of I^{pm}) is 0, we create a special candidate set to contain images with estimated distance of 0. This special set is then merged with the answer set returned by the similarity matching; the resulting set of images are ranked thereafter. This idea can be generalized for handling cases where the image has an estimated distance less than some small predetermined threshold ϵ.

4 SEARCH STRATEGIES

As mentioned earlier, in the branch-and-bound algorithm proposed by Chen *et al.*, all the images in the database are first checked at the coarsest scale. Then, the algorithm proceeds to finer scales in non-descending order of distance value. In general, the branch-and-bound search works in such a way that it always keeps track of the distance values of all images contending for further consideration. Images with smallest values are 'extended' to a finer scale. Then, these most recently 'extended' images are considered along with the remaining ones. Again, images with smallest values are 'extended'. The process repeats until the target images are found.

Due to the effectiveness of the above brand-and-bound strategy for handling whole-image queries, this strategy can be adapted to handle subimage queries. However, at each iteration of the search, the color histograms used in the computation of distance values may be at a different level/scale and may be for images different from those in the previous iteration. As a result, it can be inefficient for large databases. In particular, the number of images the strategy must keep track of can be large. Jumping back and forth in the data file to get the necessary histograms for computation may frequently be required. For large databases, such jumping makes it hard to optimize file organization and buffer management, and may impose a high I/O cost.

In this paper, we consider three strategies that try to avoid the kind of jumping mentioned above. The first strategy, called **PV (Pure Vertical)**, checks the images one after another 'vertically' (across levels), instead of jumping back and forth. At any point in time, the strategy keeps the current u smallest distances (images at their finest scale), where u is the number of images requested by the user. When the next image is tested, if the distance value of this image at the coarsest scale is already larger than the current u smallest, then this image can be eliminated. Otherwise, a finer scale is used, until the image is eliminated or is added to become one of the current u smallest.

To investigate the efficiency and the effectiveness of the PV strategy, analytical and experimental evaluations are carried out. A cost model is set up during the analyses for estimating both CPU and I/O costs of various filtering schemes using the PV strategy. In the experiments, subimage queries of arbitrary size are used. Execution time (evaluating efficiency) and dissimilarity

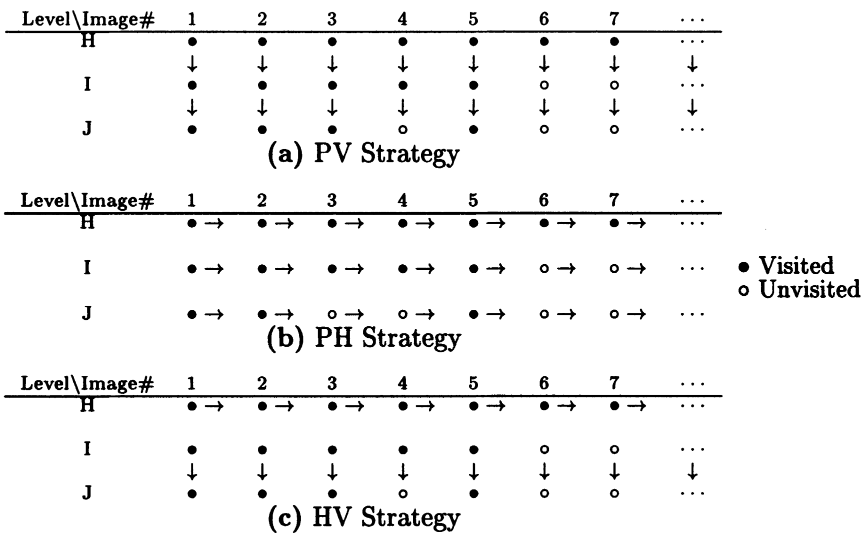


Figure 5 Three proposed search strategies.

score (evaluating effectiveness) are measured for each filtering scheme. Our aim is to maintain a balance between efficiency and effectiveness.

Note that the PV strategy tends to require many comparisons at the finest scale, particularly at the beginning of the search. Thus, we consider the second strategy, called **PH (Pure Horizontal)**, which avoids the kind of numerous comparisons at the finest scale by first checking all images at the coarsest scale. The best μ matches (for some value $\mu \gg u$), are carried over to the next iteration, while all the remaining ones are eliminated. In the next iteration, the next finer scale is used. The process is repeated until the finest scale has been used, and the top u images are returned. Here, images are checked 'horizontally' (across database) level by level to avoid the kind of frequent jumping as in the branch-and-bound.

Results of analytical and experimental evaluations show that the PH strategy gives better performance than the PV strategy. Unlike the PV strategy, the number of images to be carried over from the current level to the next level is controlled by the predefined parameters μ's. However, we need to carefully choose the values of μ's. If this set of numbers (μ's) is not determined carefully (say, the numbers are too small), then for some queries, an image I that gives a good match at a finer scale could have been eliminated before reaching this finer scale. This may happen when there are sufficiently many images which are not as good as I at the finer scale but which are better than I at the coarser scale. Consequently, while delivering efficiency, the PH strategy may suffer from a loss of effectiveness.

The third strategy, called **HV (Horizontal-and-Vertical)**, is a hybrid of PV and PH. At the coarsest scale, the PH strategy is applied to eliminate poor matches. The best μ matches are then carried over to the next stage, in which the PV strategy is used. In other words, horizontal filtering is applied only to the coarsest level, and vertical filtering is then applied to the remaining levels.

4.1 Analytical Evaluation

As can be viewed from Figure 6, the I/O cost for the brand-and-bound strategy (denoted by B&B) are potentially high. This explains why we need to consider three search strategies — namely, PV, PH, and HV — which reduce the I/O costs by avoiding the kind of frequent jumping incurred in the branch-and-bound strategy. To find the best strategy among these three, analyses and experiments have been carried out. Analytically, we set up the cost model to evaluate the filtering schemes and estimate their CPU, I/O, and combined CPU and I/O costs. Cost formulas for each filtering scheme and for each of the three search strategies can be found in Leung's (1997) work.

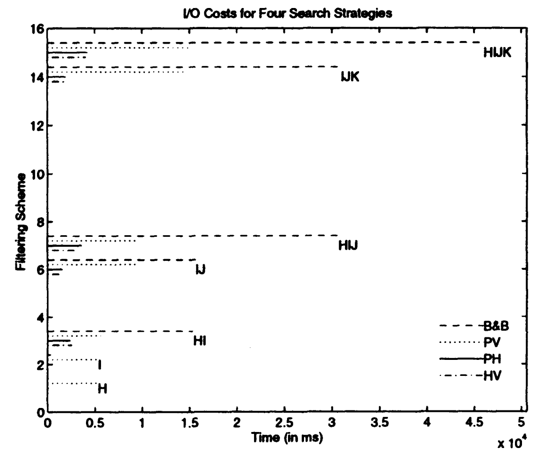


Figure 6 The I/O costs for 4 search strategies.

After we estimated the CPU and I/O costs for each combination of filtering scheme and search strategy, results are analyzed. We found that:

- All three strategies share a common trend: Filtering schemes starting with Level H (or Level I) and those not skipping intermediate levels are considered to be favorable.
- While delivering high degree of accuracy, the PV strategy suffers from a loss of performance/speed (Figure 7).
- The combined CPU and I/O costs for both PH and HV strategies are almost the same (Figure 7). Further experiments are needed to determine the best search strategy.

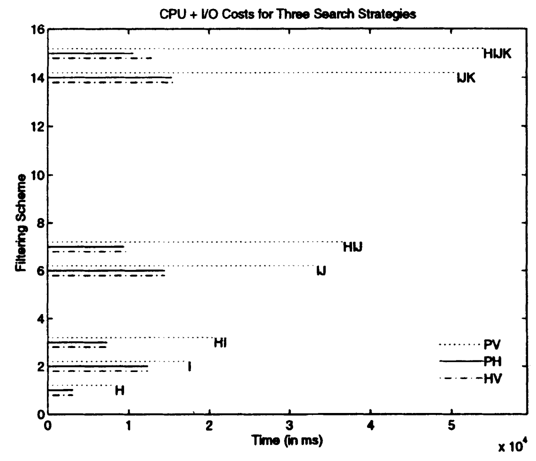


Figure 7 The combined CPU and I/O costs for 3 search strategies.

4.2 Experimental Evaluation

To find an efficient and effective one among the three multiscale searching strategies, we performed several experiments using color histograms of various dimensions (*e.g.*, 8-, 64-, and 512-dimensional color histograms). These histograms are created for a database consisting of a thousand real images collected from various sources and covering wide application domains. Comparing the runtimes and the accuracies of the best configuration (the best filtering scheme for each kind of 'large', 'medium', 'small', and 'tiny' subimage queries) for each strategy, we found that:

- The HV strategy is more efficient than the PV strategy, because the use of vertical filtering in the latter is applied to the whole set of database images. In the HV strategy, the detailed search with the use of vertical filtering is applied not to the set of all the images, but only to its most promising subset.
- The HV strategy is more effective than the PH strategy, because the latter uses horizontal filtering at all levels. If the number of images to be carried over from the current level to the next level is not determined carefully, the PH strategy may suffer from a loss of effectiveness. In the HV strategy, horizontal filtering is applied not to all the levels, but only to the coarsest level. So, we only need to carefully assign a value to one (instead of a maximum of three) predefined parameter μ that controls the number of images to be carried over. As such, the chance of having the desired images being eliminated before reaching the finer scale is reduced.

Therefore, our experimental results confirm that the HV strategy not only avoids the kind of frequent jumping as observed in the branch-and-bound strategy, it also keeps a good balance between efficiency and effectiveness, when compared with the other two strategies. Running the HV strategy on a Sun UltraSPARC-1 workstation, the best 10 desired images can be retrieved from a collection of thousand images in about 3.5 seconds on average.

5 CONCLUSIONS

In this paper, we have addressed the following key issues:

1. TO FIND A METHOD FOR PROCESSING SUBIMAGE QUERIES OF ARBITRARY SIZE : To answer arbitrary-size subimage queries, we developed two algorithms, the Padding Algorithm and the Reduction Algorithm. Knowing the image block I may not necessarily be of the same size as the query Q, we use these two algorithms to estimate the best possible color histograms for Q and I. Here, we either (i) enlarged Q into a new query Q' that is of the same size as I or (ii) reduced I to a new image block I' that is of the same size as Q. For a given Q and I, both algorithms give the same best-case lower bound to the histogram distance. However, their efficiency may differ significantly depending on the size differential between Q and I.
2. TO DETERMINE THE SUITABLE NUMBER OF LEVELS FOR MULTISCALE REPRESENTATION: Given subimage queries of arbitrary size, multiscale representation may improve the efficiency and the effectiveness of content-based retrieval. Since image contents are usually pre-extracted and stored, we need to determine the suitable number of levels for the multiscale representation. Analytically, we estimated the required CPU and I/O costs; experimentally, we compared the performance and the accuracy of the outcomes of strategies for searching the multiscale representation. Our findings suggest that a 3-level hierarchy (up to 4×4 segmentation) is preferred.
3. TO FIND AN EFFICIENT AND EFFECTIVE MULTISCALE SEARCH STRATEGY: In this paper, we studied three search strategies, namely Pure Vertical, Pure Horizontal, and Horizontal-and-Vertical. To find the best strategy among these three, analytical and experimental evaluations were conducted. Our results indicate that the Horizontal-and-Vertical strategy is the best when using Padding and Reduction Algorithms, because the strategy not only avoids high I/O costs incurred by the frequent jumping in the data file, but also keeps a good balance between performance and accuracy.

Although the results of this paper are very promising, there are some aspects to consider for further improvement. One aspect is to investigate methods to extend our Padding and Reduction Algorithms for handling user confidence or uncertainty on color and spatial information. With the extension, users will be able to express the degree of uncertainty on image contents within the subimage.

Throughout the paper, we focus on the case where the user remembers only a single portion of the image. However, it is possible that the user may remember more than one portion of the images he has seen before. Hence, methods for handling subimage queries with multiple 'known' portions are needed. A brute-force approach is to apply Padding and Reduction Algorithms to each portion independently. An intersection is then applied to the candidate sets of images retrieved by the Algorithms, and the resulting images are ranked thereafter. However, the execution time for this approach may be high; more efficient approaches are necessary.

ACKNOWLEDGEMENTS

This research was partially sponsored by NSERC Grant OGP0138055, and IRIS-2 Grants HMI-5 and IC-5.

REFERENCES

Bach, J.R. *et al.* (1996) The Virage Image Search Engine: An Open Framework for Image Management. *Proceedings of SPIE Conference on Storage and Retrieval for Still Image and Video Databases IV* (Vol. 2670): 76–87. San Jose CA, USA.

Barber, R. *et al.* (1994) Ultimedia Manager: Query By Image Content and its Applications. *Digest of Papers of the Spring COMPCON '94*: 424–429. San Francisco CA, USA.

Castelli, V. *et al.* (1997) *Searching Image Databases at Multiple Levels of Abstraction.* Research Report RC 20702, IBM T. J. Watson Research Center, Yorktown Heights NY, USA.

Chen, J.-Y. *et al.* (1997) Multiscale Branch and Bound Image Database Search. *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases V* (Vol. 3022): 133–144. San Jose CA, USA.

Faloutsos, C. *et al.* (1994) Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems* **3**(3–4): 231–262.

Faulus, D.S. and Ng, R.T. (1997) An Expressive Language and Interface for Image Querying. *Machine Vision and Applications* **10**(2): 74–85.

Flickner, M. *et al.* (1995) Query by Image and Video Content: The QBIC System. *IEEE Computer* **28**(9): 23–31.

Gudivada, V.N. and Raghavan, V.V. (1995) Content-based Image Retrieval Systems. *IEEE Computer* **28**(9): 18–22.

Guttman, A. (1984) R-trees: A Dynamic Index Structure for Spatial Searching. *Proceedings of ACM SIGMOD Conference on Management of Data*: 47–57. Boston MA, USA.

Jacobs, C.E. *et al.* (1995) Fast Multiresolution Image Querying. *Proceedings of ACM SIGGRAPH Conference on Computer Graphics & Interactive Techniques*:

277–286. Los Angeles CA, USA.

Jain, R., editor (1993) NSF Workshop on Visual Information Management Systems. *SIGMOD Record* **22**(3): 57–75.

Leung, K.S. (1997) *Efficient and Effective Subimage Similarity Matching for Large Image Databases.* Master's Thesis, The University of British Columbia, Vancouver BC, Canada.

Lin, K.-I. *et al.* (1994) The TV-tree — An Index Structure for High-dimensional Data. *VLDB Journal* **3**(4): 517–549.

Ng, R.T. and Tam, D. (1997) An Analysis of Multi-level Color Histograms. *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases V* (Vol. 3022): 22–34. San Jose CA, USA.

Petkovic, D. *et al.* (1996) *Recent Applications of IBM's Query By Image Content (QBIC).* Research Report RJ 10006, IBM Almaden Research Center, San Jose CA, USA.

Samet, H. (1990) *The Design and Analysis of Spatial Data Structures.* Addison-Wesley.

Sawhney, H.S. and Hafner, J.L. (1993) *Efficient Color Histogram Indexing for Quadratic Form Distance Functions.* Research Report RJ 9572, IBM Almaden Research Center, San Jose CA, USA.

Stricker, M. and Dimai, A. (1996) Color Indexing with Weak Spatial Constraints. *Proceedings of SPIE Conference on Storage and Retrieval for Still Image and Video Databases IV* (Vol. 2670): 29–40. San Jose CA, USA.

Swain, M.J. and Ballard, D.H. (1991) Color Indexing. *International Journal of Computer Vision* **7**(1): 11–32.

Thomasian, A. *et al.* (1997) *RCSVD: Recursive Clustering with Singular Value Decomposition for Dimension Reduction in Content-based Retrieval of Large Image/Video Databases.* Research Report RC 20704, IBM T. J. Watson Research Center, Yorktown Heights NY, USA.

6 BIOGRAPHY

Kai-Sang Leung participates in the image database management system project led by Raymond Ng, focusing on efficient and effective query processing and optimization. Research work done in the participation is largely on subimage query processing. Kai-Sang Leung is currently pursuing his Ph.D. in computer science at UBC.

Raymond Ng, together with his colleagues at UBC, leads an 8-year project in building a generic image database management system that can support automatic feature extraction and analysis, and a variety of application domains. The project is in its fourth year. Raymond Ng has published widely on major database conferences (*e.g.*, VLDB, SIGMOD) and journals (*e.g.*, Journal of ACM, IEEE Computer, ACM TODS, ACM Multimedia). Raymond Ng is an associate professor at UBC.