Speeding up the Similarity Search in High-Dimensional Image Database
by Multiscale Filtering and Dynamic Programming

Shyi-Chyi Cheng (鄭錫齊), Tian-Luu Wu (伍添祿)*
Institute of Computer and Communication Engineering
National Kaohsiung First University of Science and Technology

*Department of Electronic Engineering, Yung Ta Institute of Technology and Commerce

Abstract
This paper presents a scalable content-based image indexing and retrieval system based on a new multiscale filter. Image databases often represent the image objects as high-dimensional feature vectors and access them via the feature vectors and similarity measure. A similarity measure based on the proposed multiscale filtering technique is defined to reduce the computational complexity of the similarity search in high-dimensional image database. Moreover, a special attention is paid to solve the problem of feature value correlation by dynamic programming. This problem arises from changes of images due to database updating or considering spatial layout in constructing feature vectors. The computational complexity of similarity measure in high-dimensional image database is very huge and this limits the applicability of image retrieval to many research areas. To demonstrate the effectiveness of the proposed algorithm, we conducted extensive experiments and compared the performance with IBM’s query by image content (QBIC) and Jain and Vailay’s methods. The experimental results demonstrate that the proposed method outperforms both of the methods in retrieval accuracy and noise immunity. The execution speed of the proposed method is much faster than that of QBIC method and it can achieve good results in terms of retrieval accuracy compared with Jain’s method and QBIC method.

Keyword: Multiscale filtering, high-dimensional image database, spatial layout.

1. Introduction
Image databases often represent the image objects as vectors of $d$ numeric features and access them via the feature vectors and similarity measure. The feature vector dimensions of typical vector-based descriptors are quite large. The high dimensionality of the feature vectors lead to high computational complexity in distance calculation for similarity retrieval, and inefficiency in indexing and search. To make the content-based image retrieval truly scalable to large size image databases, efficient multidimensional indexing techniques need to be explored.

Several methods have been proposed to overcome these problems [1]. The techniques can be roughly categorized into the following classes [2,3]: (1) the dimensionality reduction (DR) approach, (2) the multidimensional indexing approach, and (3) the filter-based approach. An indexing algorithm may be a combination of two or more of the mentioned classes. For example, one promising approach is to first perform dimension reduction and then use appropriate multidimensional indexing techniques.

Even though the dimension of the feature vectors in image retrieval is normally very high, the embedded dimension is much lower [4]. Principle component analysis (PCA) [4] is a good way to condense most of information in a data set to a few dimensions. One thing worth pointing out is that blind dimension reduction is dangerous, since information can be lost if the reduction is below the embedded dimensions [5].

The existing popular multidimensional indexing techniques include the bucketing algorithm, k-d tree, priority k-d tree, quad-tree, K-D-B tree, hB-tree, R-tree and its variants R*-tree. In
addition to the above approaches, clustering and neural nets, widely used in pattern recognition, are also promising indexing techniques. Very good reviews and comparisons of various indexing techniques in image retrieval can be found in [6]. The multidimensional indexing approach receives a challenge to access image databases: the performance of existing multidimensional indexing schemes degrades dramatically as the dimensionality increases [3,7].

The filter-based approach searches the nearest \( k \) neighbors of a query by filtering the vectors so that only a small portion of them must be visited. The percentage of vectors visited during a search depends on the strategy used to design the filter. As an example, the LPC-file [2] partitions the vector space into rectangular cells and these cells are used to generate bit-encoded approximates for each vector. The drawbacks of the filter-based approach are: (1) the design of the approximations is not a trivial work and the precision of the approximation is not good while applying to image data of good locality; (2) additional information should be added to the approximate in order to enhance the filtering rate when the database is getting larger and larger.

This paper presents a scalable content-based image indexing and retrieval system based on a new multiscale filter. The image objects are represented as high-dimensional feature vectors and users can access them via the feature vectors and similarity measure. A similarity measure based on the proposed multiscale filtering technique is defined to reduce the computational complexity of the similarity search in high-dimensional image database. Moreover, a special attention is paid to solve the problem of feature value correlation by dynamic programming [8]. This problem arises from changes of images due to database updating or considering spatial layout in constructing feature vectors. The computational complexity of similarity measure in high-dimensional image database is very huge and this limits the applicability of image retrieval to many research areas. To demonstrate the effectiveness of the proposed algorithm, we conducted extensive experiments and compared the performance with the IBM’s query by image content (QBIC) [9] and Jain and Vailay’s method [10]. The experimental results demonstrate that the proposed method outperforms both of the methods in retrieval accuracy and noise immunity. The execution speed of the proposed method is much faster than that of QBIC method and it can achieve good results in terms of retrieval accuracy compared with Jain’s method and QBIC method.

### 2 Building the multiscale feature vectors

Consider a database DB consisting of a large number of image objects, each of them is represented as a \( d \)-dimensional feature vector \( F = \{ \{ f_i, p_j \}, i = 1, \ldots, d \} \), where \( f_i, p_i \) are the \( i \)th feature and the corresponding percentage of pixels of the vector, respectively. Depending on the types of features used in the system, each \( f_i \) could be a scale, a vector, or a set of vectors. The Euclidean distance between a pair of image objects is commonly used as a dissimilarity measure. The typical way to compute the dissimilarity between two objects is using an \( L_2 \) distance metric. Let \( s = \{ \{ s_i, p_j \}, i = 1, \ldots, d \} \) and \( t = \{ \{ t_i, q_j \}, i = 1, \ldots, d \} \) be two feature vectors. Then the distance between \( s \) and \( t \) is computed as

\[
D^2(s,t) = \sum_{i=1}^{d} p_i^2 + \sum_{j=1}^{d} q_j^2 - \sum_{i=1}^{d} \sum_{j=1}^{d} 2a_{i,j}p_iq_j
\]

where \( a_{i,j} \) is the similarity coefficient between features \( s_i \) and \( t_j \),

\[
a_{i,j} = 1 - \frac{dist(s_i,t_j)}{dist_{\text{max}}}
\]

where \( dist_{i,j} \) is the Euclidean distance between feature values \( s_i \) and \( t_j \) and \( dist_{\text{max}} \) is the maximum distance between any two feature values. This metric gives the weighted length of the difference vector between \( s \) and \( t \).
and \( t \), weighted by \( a_{i,j}, i = 1,\ldots,d_i; j = 1,\ldots,d_j \), which accounts for the perceptual distance between different pairs of features.

\[
da_{i,j} = \frac{1}{d_i} \sum_{j=1}^{d_j} \frac{1}{d_j} \]

Fig. 1. An example of representing an image by a three-level feature vector set: (a) separate the image into a set of disjoint regions by a segmentation algorithm [13]; (b) scan the regions in the left-to-right and top-to-down fashion to construct a region sequence; (c) represent each region by a different scale feature value and form a different scale feature vector for the image.

It is necessary to clarify the difference between “multiresolution” and “multiscale” in the context of this work. The resolution of an image signifies the area in physical units a pixel corresponds to. When the same physical object is seen at a difference, the resolution of the image changes. At the same time, the number of pixels the image of the object occupied in the retina reduces. Many multiresolution transformation algorithms [11,12] were reported in the literature to create a pyramid of image resolutions by subsampling. We chose not to perform this subsampling, hence we create a tower of feature vector instead of a pyramid for image retrieval. That is a feature vector is represented in \( N \) different scales.

Fig. 1 shows an example to represent an image by a three-level feature vector set by the following process: (1) separate the image into a set of disjoint regions by a segmentation algorithm [13] and scan the regions in the left-to-right and top-to-down fashion to construct a region sequence; (2) construct the first level feature vector by representing each region of the region sequence as a color histogram; (3) construct the second level feature vector by representing each region as its mean; (4) construct the third level feature vector by representing each region as the average of the tri-stimulus values (\( R, G, B \)) of the mean color.

The content of an image in this paper is represented by a set of multiscale feature vectors, which reflect the spatial layout of visual features of the image. The similarity of two images is defined as the difference between their corresponding highest resolution feature vectors. However, the computational complexity of similarity measurement using the feature vectors of higher resolutions is high. To avoid the problem, a multiscale filtering technique is proposed to reduce the number of high-resolution feature vectors to be visited during a search. In practice, lower-resolution vectors are first used to conduct a search and the results provide good lower bounds to filter irrelevant matches based on high-resolution vectors. The detail of the multiscale filter will be discussed later.

3. Similarity measure by dynamic programming

Considering two similar images, the regions segmented based on the same segmentation algorithm would be close in representative visual features. However, due to varying viewpoints of the image, occlusion, and varying lighting conditions, the environment for identifying an image is nonstationary. To find the distance between two sequences of feature
In these situations, the dynamic matching method, which is commonly known to be optimal in aligning two strings, is proposed. As shown in Fig. 2, where input sequences $A$ and $B$ are developed along the horizontal $i$-axis and vertical $j$-axis, respectively, we try to find a path that represents the best match between two input sequences. This best match path is shown as the sequence $C$ in Fig. 2. After this match path is obtained, we then calculate the distance of each matched pair of characteristic values, which are feature vectors from $A$ and $B$. Last, the distance between two sequences of feature vectors is defined as the sum of these feature distances. The matching path is denoted as

$$M(A, B) = (c(1), c(2), \ldots, c(k), \ldots, c(r))$$ (3)

where $c(k) = (i(k), j(k))$ and $r$ is the length of the alignment. Note that $max(m, n) \leq r \leq m + n$. When there is no difference between $A$ and $B$, the match path coincides with the diagonal line $i = j$. It deviates further from the diagonal line as the difference grows.

The recurrent equation used to align $A$ and $B$ with the shortest distance in a bottom-up fashion by dynamic programming is

$$d[i, j] = \min \begin{cases} 
  d[i - 1, j - 1] + dist(a_i, b_j) \\
  d[i, j - 1] + dist(a_i, b_j) \\
  d[i - 1, j] + dist(a_i, b_j)
\end{cases}$$ (4)

where $d[i, j]$ denotes the alignment cost between $A_j = (a_1, a_2, \ldots, a_i)$ and $B_j = (b_1, b_2, \ldots, b_j)$. The goal of the recurrent equation is to find out the value of $d[m][n]$, where $m$ and $n$ are the length of $A$ and $B$, respectively. Note that the less the value of $d[m][n]$, the similar the two sequences.

As mentioned in the previous section, an image is represented as a tower of feature vectors of different resolutions. Let $d^{(k)}[i, j]$ denote the shortest distance between two sequences of feature vectors $A^{(k)} = (a_1^{(k)}, a_2^{(k)}, \ldots, a_{i(k)}^{(k)})$ and $B^{(k)} = (b_1^{(k)}, b_2^{(k)}, \ldots, b_{j(k)}^{(k)})$ with the $k$th level resolution. In this paper, the distance between the two images $A$ and $B$ is defined as

$$D^2(A, B) = d^{(3)}[m, n].$$ (5)

That is the value of similarity between two images is computed by dynamic programming using the feature vectors of the highest resolution. Unfortunately, the computation of Equation (5) is time consuming. A fast algorithm based on a multiscale filtering technique is proposed to solve the problem.

4. The proposed multiscale filter

For illustration convenience, we assume the type of visual feature used to design the proposed image retrieval system is color. In this case, the three-level feature vectors used to characterize the content of an image can be defined as

$$F^{(1)} = (H_1, H_2, \ldots, H_N)$$
$$F^{(2)} = (\overline{c}_i, \overline{c}_k, \ldots, \overline{c}_N)$$
$$F^{(3)} = (\overline{m}_i, \overline{m}_k, \ldots, \overline{m}_N)$$ (6)

where $H_i = ([c_i^1, p_i^1], [c_i^2, p_i^2], \ldots, [c_i^r, p_i^r])$ is the color histogram with $T$ bins to characterize region $i$; $\overline{c}_i$ is the mean color of the region $i$; $\overline{m}_i$ is the average of the tri-stimulus values $(\overline{R}_i, \overline{G}_i, \overline{B}_i)$ of $\overline{c}_i$; $N$ is the number of region segmented. That is

$$\overline{c}_i = \frac{1}{p_i} \sum_{j=1}^{p_i} c_i, i = 1, \ldots, N$$ (7)

and

$$\overline{m}_i = \frac{\overline{R}_i + \overline{G}_i + \overline{B}_i}{3}, i = 1, \ldots, N.$$ (8)

For the construction of the multiscale filter, three distance functions are defined to compute the distances between feature values from different
resolution feature vectors. The third-level distance function used to calculate the distance of the average values of representative colors of regions \(a\) and \(b\) is defined as

\[
\text{dist}^{(3)}(a, b) = 3(m_ar - m_br)^2. \tag{9}
\]

The Euclidean distance is used to compute the distance of the two regions \(a\) and \(b\) in terms of mean color difference

\[
\text{dist}^{(2)}(a, b) = (\overline{r}_a - \overline{r}_b)^2 + (\overline{g}_a - \overline{g}_b)^2 + (\overline{b}_a - \overline{b}_b)^2. \tag{10}
\]

The feature values of the first-level feature vectors are color histograms. The quadratic distance function defined in Equation (1) is used to compute the distance between two histograms. These functions satisfy the following formula [3]:

\[
\text{dist}^{(i)}(a, b) \leq \text{dist}^{(2)}(a, b) \leq \text{dist}^{(3)}(a, b) \tag{11}
\]

for every pair of regions \(a, b\).

Given two sequences of feature vectors \(A_m = (a_1, a_2, ..., a_m)\) and \(B_n = (b_1, b_2, ..., b_n)\), the number of possible ways to align \(A_m\) and \(B_n\) by dynamic programming using the recurrent equation defined in Equation (4) is \(3(m - 1)(n - 1) + 2\). Among them, the proposed multiscale filter conducts a branch-and-bound search with dynamic programming based on the third-level feature vectors to generate the match paths between \(A_m\) and \(B_n\) ascending in terms of alignment cost. The match paths are then used to calculate the first-level distance \(D^{(1)}\), the second-level distance \(D^{(2)}\), and the third-level distance \(D^{(3)}\) of \(A_m\) and \(B_n\) by summing up the distance of each match pair of the match paths using Equations (1), (11), and (10), respectively. Note that

\[
D^{(1)} \geq D^{(2)} \geq D^{(3)} \tag{12}
\]

according to Equation (11). The goal of the proposed filter is to find the match path of minimal alignment cost based on the first-level feature vectors. Once it is generated, the process to generate the match paths stops. The proposed multiscale filtering algorithm is described below to complete the discussion.

Algorithm 1. Proposed multiscale filtering (PMF).

Input: Two sequences of feature vectors 
\(A = (a_1, a_2, ..., a_m)\) and \(B = (b_1, b_2, ..., b_n)\) with multiscale representation.

Output: The minimal distance between \(A\) and \(B\).

Method:

1. Generate \(d^{(i)}[i, j]\) \(i = 1, ..., m; j = 1, ..., n\) by dynamic programming using Equations (4) and (10).

2. Let \(i, j, \text{ and } k\) denote the indexes of the match path order sets \(S^{(1)}, S^{(2)}, \text{ and } S^{(3)},\) respectively. Initially, \(i = j = k = 0;\)

3. Let \(D^{(1)}, D^{(2)}, \text{ and } D^{(3)}\) denote the cost of the match path under study based on the level-1, the level-2, and the level-3 feature vectors, respectively.

4. Let \(\delta^{(1)}\) and \(\delta^{(2)}\) denote the minimal cost of match path found so far based on the level-1, and the level-2 feature vectors, respectively. Initially, \(\delta^{(1)} = \delta^{(2)} = \infty.\)

5. While \(i < 0\) do the following steps:

5.1 generate a new level-3 match path \(M^{(3)}_k\) with its cost larger than that of previously generated match paths;

5.2 put \(M^{(3)}_k\) on the \(k\)th location of \(S^{(3)};\)

5.3 calculate \(D^{(3)}\) by summing up the distance of each match pair of \(M^{(3)}_k\) using Equation (10);

5.4 if \((\delta^{(2)} \geq D^{(3)})\) then

5.4.1 add \(M^{(3)}_k\) to a temporal Queue \(Q_{\delta^{(1)} \rightarrow S^{(3)}};\)

5.4.2 calculate \(D^{(2)}\) by summing up the distance of each match pair of \(M^{(3)}_k\) using Equation (11);

5.4.3 if \((\delta^{(2)} > D^{(2)})\) then \(\delta^{(2)} = D^{(2)}\) and \(M^{(2)}_j = M^{(3)}_k;\) else do nothing;

5.4.4 increase \(k\) by 1;

5.5 else do the following steps:

5.5.1 put \(M^{(2)}_j\) on the \(j\)th location of
$S^{(2)}$ and increase $k$ by 1;

5.5.2 if ($\delta^{(2)} \geq \delta^{(1)}$) then

5.5.2.1 add $M^0_j$ to a temporal Queue $Q_{s-n\rightarrow s}$;

5.5.2.2 calculate $D^{(3)}$ by summing up the distance of each match pair of $M^{(2)}_j$ using Equation (1);

5.5.2.3 if ($\delta^{(1)} > D^{(3)}$) then

\[ \delta^{(1)} = D^{(1)} \text{ and } M^{(1)}_j = M^{(2)}_j; \]

else do nothing;

5.5.2.4 remove $M^{(2)}_j$ from $Q_{s-n\rightarrow s}$;

5.5.2.5 set the $M^{(2)}_{j+1}$ to the match path of minimal cost in $Q_{s-n\rightarrow s}$ and set the new $\delta^{(2)}$ to the $D^{(2)}$ of $M^{(2)}_{j+1}$;

5.5.2.6 increase $j$ by 1;

5.5.3 else

5.5.3.1 put $M^{(1)}_i$ on the $i$th location of $S^{(1)}$ and increase $i$ by 1.

6 Return ($\delta^{(1)}, M^{(1)}_0$).

Step 5.4 of the above algorithm tries to search the $j$th member of $S^{(2)}$ from the members of $S^{(0)}$ sequentially. Note that the $j$th member of $S^{(2)}$ found if the value of $D^{(3)}$ of the $j$th match path in $S^{(0)}$ is larger than $\delta^{(2)}$ which records the cost of the candidate $j$th member of $S^{(2)}$. We need not check the remaining math paths of $S^{(3)}$ to determine the $j$th member of $S^{(2)}$ because their $D^{(3)}$s will be larger than $\delta^{(2)}$ according the inequality defined in Equation (12). Following the same principle, Step 5.5.2 tries to find the first member of $S^{(1)}$ from those of $S^{(2)}$.

5. Experimental results

In order to evaluate the proposed approach, a series of experiments were conducted on an Intel PENTIUM-III 500MHz and four databases of 627, 1866, 8380, and 10456 natural images are used. Each image is first tailored into 256x256 for testing the proposed retrieval approach. The performance of the proposed image retrieval method is evaluated in terms of execution speed and retrieval accuracy.

![Fig. 3. Average precision versus number of retrieved images.](image1)

![Fig. 4. Average recall versus number of retrieved images.](image2)

The retrieval technique based on edge and color histogram proposed by Jain and Vailaya [10] was also implemented for performance comparison. Before the evaluation, human assessment was done to determine the relevant matches in the database to the query images. The top 100 retrievals from the database and the proposed approaches were marked to decide whether they were indeed visually similar. The retrieval accuracy was measured by precision and recall

\[ \text{Precision (k)} = \frac{C_k}{k} \text{ and Recall (k)} = \frac{C_k}{M} \]
where \( k \) is the number of retrievals, \( C_k \) is the number of relevant matches among all the \( K \) retrievals, and \( M \) is the total number of relevant matches in the database obtained through human assessment. The average precision and recall curves are plotted in Figs. 3 and 4. It can be seen that the proposed method achieves good results in terms of retrieval accuracy compared with Jain’s method and QBIC method.

![Average retrieval time vs size of database](image)

**Fig. 5.** Average retrieval time (in seconds) versus the size of database.

Fig. 5 shows the performance comparison in terms of the average retrieval time (in seconds) in response to a query using the proposed method, the histogram-based (QBIC) method, and Jain’s method with different sizes of databases. Each image will be segmented into 32 regions and each region is characterized by 4 dominant colors. Comparing with the color-histogram method (128 bins, 256 bins, 512 bins, and 1024 bins) and Jain’s method, the proposed method is very fast. Although the proposed method is not faster than Jain’s method if the case of small test database, the proposed method sustains good performance even when the test database grows up to be large.

In order to test the robustness of the proposed system, both normal and noise images are considered to access databases. Fig. 6 is one retrieval example of the proposed method using a noise query image. Experimental results show that the proposed method is superior to other methods for comparison.

**5. Conclusion**

In this paper we have presented a scalable content-based image indexing and retrieval system based on a new multiscale filter. The similarity measurement based on the proposed multiscale filtering technique has been defined to reduce the computational complexity of the similarity search in high-dimensional image database. Moreover, a special attention is paid to solve the problem of feature value correction by dynamic programming. Comparing with QBIC and Jain’s methods, the experimental results demonstrate that the proposed method outperforms both of methods in retrieval accuracy, noise immunity, and the performance.

The proposed method suffers a drawback: the searching time linearly depends on the number of images in a database. Future work should be done to solve this problem.

**References**


Figure 6. Example of retrieval results of the proposed method using a noise query image. The top image is the query image, the retrieval results are ranked from left to right and top to bottom according to their similarity measurements. The score of similar measurement is at the bottom of each image.