An Efficient Core-Area Detection Algorithm for Fast Noise-Free Image Query Processing

Khanh Vu
School of EECS
University of Central Florida
Orlando, FL 32816-2362
khanh@cs.ucf.edu

Kien A. Hua
School of EECS
University of Central Florida
Orlando, FL 32816-2362
kienhua@cs.ucf.edu

Duc A. Tran
School of EECS
University of Central Florida
Orlando, FL 32816-2362
dtran@cs.ucf.edu

Keywords

content-based image retrieval, indexing/retrieval, noise-free queries, core-area detection algorithm

ABSTRACT

Recent content-based image retrieval techniques enable users to arbitrarily exclude noise (i.e. irrelevant regions) from image similarity consideration. This capability has resulted in high retrieval effectiveness for a wide range of queries. To support large image collections, subimages of a predetermined *base* shape (e.g., circle, polygon) are collected and indexed into a multidimensional access structure. At the query time, an area of such a shape enclosing part of the queried objects, called the *core area*, will be identified and used in the initial search of potential candidates before an appropriate detailed similarity measure is performed on the original query. Identifying the core area of a query can be challenging as it is allowed to contain certain noise and may not be unique. In this paper, we propose an efficient algorithm, called the *Seed-Growing Detection Algorithm*, to automatically detect the optimal core area. We have implemented the proposed technique in our image retrieval system for a large database. Our experimental results show that our approach is effective and able to minimize time overhead of query preprocessing.

1. INTRODUCTION

The popularity of digital image data has spurred a demand for robust *content-based image retrieval* (CBIR) systems. These systems are needed in various application domains including medical imaging, digital libraries, geographic mapping, to name a few. The growing demand for this technology has attracted a significant research interest addressing CBIR problems ([12, 19, 6, 5, 17, 14, 9, 4], etc.).

In a typical Query-By-Example (QBE) environment, users formulate their query by means of example images selected from a pool of general image categories. Since this example set is typically small, the expectation of finding a perfect example (i.e. the entire content is relevant) is low. Therefore, an example should be treated as a collection of regions some of that form the user's objects of interest and the remaining is noise. A robust CBIR system should enable users to exclude noise in formulating queries. Queries so defined are called *noise-free queries* (NFQs).

The focus of most CBIR systems is to achieve robustness to translation, scaling, and changes at the image level. In a typical CBIR system, some essential properties of the database images are extracted and stored as *feature vectors*. During the retrieval process, the feature vector of the query image is computed and compared against those in the database. These feature vectors typically capture the color distribution. Recent techniques also include other features such as spatial information [9], texture [13], structure [4], etc. to increase the overall effectiveness. In these schemes, referred to as *whole-matching* approaches, the entire image content is uniformly treated and its attributes are tightly integrated. Noise exclusion is not possible in these systems.

To minimize the effect of global features on local matches, similarity computation can be performed at subimage levels. One major direction in supporting subimage-level matching is object-based image retrieval. A number of strategies (segmentation [19, 10], back-projection [16],...) have been proposed relying on region boundaries, edges, color, texture, connectivity, object model, etc. However, automatic recognition of real-world objects, which are complex in content and can be of any shape, is a very hard problem. Besides being computationally expensive, these systems are often inaccurate in identifying objects. The reason is that the definition of objects is largely subjective, thereby, pre-determined objects cannot serve as a universal template for matching in all applications. When users contend that noise and their perceived objects of interest are mingled in predetermined complex objects, it is difficult to support NFQs.

Recent CBIR techniques proposed in [8, 15, 6, 11] address this drawback. A common trait of these approaches is to decompose images into a fixed number of blocks, and their features are captured and saved separately. These techniques do not attempt to identify objects or related regions at the database build time. At the query time, similarity is determined based only on regions specified by the defining user. Experiments have shown that effectiveness is greatly improved for a wide range of NFQs.

To support large image collections, windows of a *base*

SAC 2001, Las Vegas, NV

shape (e.g. polygon, circle) are slid over the processed image to extract its subimages' features for indexing. At the query time, an area of such a shape enclosing part of the queried objects will be identified and used in the initial search of potential candidates before the final similarity measure is performed on the original query.

In general, the base-shaped area of an NFQ, called the *core area*, is allowed to contain certain noise so that its size can be maximized to capture the NFQ characteristics as precisely as possible. A core area is optimal with respect to some preset limit if it is the largest but contains noise no greater than this limit. For average users, identifying such an optimal core area of an NFQ can be a challenging task since it can compromise the effectiveness of the initial search and thereby the retrieval quality. In this paper, using *SamMatch* [8] as the underlying similarity model, we propose an efficient algorithm to search for the optimal core area of the query to automate the entire retrieval process,

The remainder of this paper is organized as follows. In Sect. 2, we present an overview of SamMatch. We preview our search technique in Sect. 3. In Sect. 4, we present an efficient algorithm to detect the core area of the query. In Sect. 5, our analysis and performance of the algorithm are discussed. Finally, we give our concluding remarks in Sect. 6.

2. SAMMATCH ENVIRONMENT

SamMatch is based on a sampling-based matching idea. In this environment, samples of 16 × 16-pixel blocks are taken at various locations of each image. The rationale for this block size is that the correlation between pixels tend to decrease after 15 to 20 pixels [2]. This characteristic allows reduction of the storage overhead by representing each sampled block using its average color. To support general applications, these blocks are collected at uniform locations throughout the image (see Fig. 1). It shows 113 samples evenly spread out in a 256 × 256 image frame.

In SamMatch, images are stored using the *Munsell* color system. In this uniform color system, the dissimilarity of two colors is simply the distance between them. (If the images are in RGB, one can convert them into Munsell using the mathematical transformation presented in [7].) The (H,V,C) data is quantized into 256 possible values. The most dominant Haar wavelet coefficient [18] of each sampled block is used as its average color.

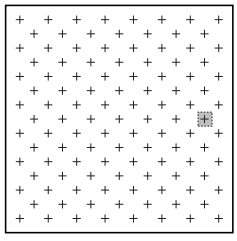

Figure 1: Sampling regions

2.1 Similarity Measure

Consider two arbitrary-shaped subimages I and Q, each represented by n regions. The color descriptor of such n regions of the subimages can be compared one-to-one to determine their dissimilarity. The distance between I and Q is computed by the following equation:

$$\mathcal{D}(Q, I) = \sqrt{\sum_{i=1}^{n} (c_i^D - c_i^Q)^2} \quad (1)$$

where c_i^D and c_i^Q, the color values of the i^{th} region of I and the corresponding region in the query image Q, respectively.

2.2 Handling Scaling of Subimages

One distinct advantage of the sampling-based approach is that the system can apply various sampling rates on the query image to find matches at various scalings. An example given in Fig. 2 illustrates how SamMatch handles scaling: a fixed sampling rate for all database images (Fig. 2 (a)), and three sampling rates for the query image: a higher rate to find larger matching objects (Fig. 2 (b)), the same rate to find matching objects of the same size (Fig. 2 (c)), and a lower rate to find smaller matching objects (Fig. 2 (d)). Clearly, we can apply many more sampling rates for finer scales to be compared at the query time. Since only one sampling rate is necessary for database images, there is no storage penalty for more sampling rates.

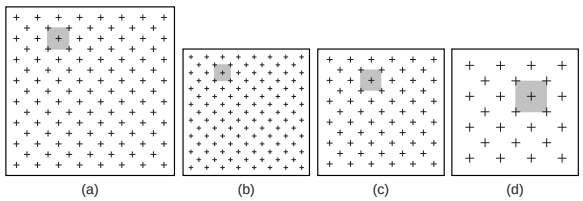

Figure 2: One fixed sampling rate for database images (a) and three sampling rates for the query (b, c, d) in order to match larger objects, same-size objects and smaller objects, respectively

3. INDEXING AND RETRIEVAL

To search for a matching subimage in a database image, we can "slide" the NFQ over the database image. At each sliding location, we compare the sampling regions of the query with the corresponding sampling regions of the subimage. This procedure can be repeated for every image in the database to retrieve all qualified images. To enable indexing database images, we slide windows of a predetermined *base* shape to extract subimages' indexing values from these images. One of the criteria for selecting such a base shape is the maximum coverage of the NFQ within a preset limit (discussed in Sect. 4). We select the square shape for our current implementation. Our approach can be easily modified to handle any base shape.

We build our access structure in three steps as follows:

1. We compute the feature vector for each database image as described earlier.

2. We slide base-shaped windows of various sizes over each database image. At each location, the sliding window encloses and defines a square-shaped subimage for the purpose of indexing. We derive the feature vector for this subimage by extracting the corresponding components of the feature vector of the whole image. A fixed-size *signature* is computed from these components.

3. For each database image, we map the signatures of its subimages into points in the corresponding vector space, and cluster them into a fixed number of *minimum bounding region* (MBRs). These MBRs collectively represent the database image, and are inserted into the R^* tree [3].

Using the access structure constructed above, our NFQ matching procedure can be summarized as follows:

1. *Query Preprocessing*:
   - We apply different sampling rates to the query image, each designed to match against subimages of a predetermined size. For each sampling rate, one feature vector is computed (Fig. 3 (a)).
   - We search the optimal core area using our efficient detection algorithm (presented in Sect. 4). We compute the signature of this area using the components of the image feature vector with the highest sampling rate (Fig. 3 (b)).
2. *Initial Search*: We retrieve relevant MBRs from the R^* tree using the core area's signature as the search key (Fig. 3 (c)).
3. *Detailed Comparison*: The signatures in the returned MBRs identify locations within the database images of potential matches. At each location, the shape of the NFQ identifies the NFQ-shaped subimage in a database image to be compared against the original NFQ. The components of this subimage's feature vector are extracted from the containing image's feature vector, and the detailed similarity measure is based on these components and those of the NFQ (see Fig. 3 (d)). If the comparison is positive, the database image containing the matching subimage is returned as a query result (Fig. 3 (e)).

In the next section, we will address the core area detection of the query in detail.

4. CORE AREA DETECTION

As mentioned in the last section, we need to determine the core area in order to use the R^*-tree in the initial search procedure. We treat this issue in this section by presenting an efficient algorithm for the detection of the optimal core area given the maximum tolerable noise level. Such a core area is optimal in the sense that it is maximized to capture the characteristics of the NFQ to the greatest extent.

4.1 Problem Definition

We first define the problem of finding the core area more formally. Given an image frame, it defines a discrete $Z \times Z$ space (digitized equivalent of Euclidean $R \times R$ space). Let us consider an NFQ $\mathtt{S}$, and a square subset $\mathtt{L}$ of the image, such that $\mathtt{S} \subseteq Z \times Z$ and $\mathtt{L} \subseteq Z \times Z$. The number of relevant pixels in $\mathtt{L}$ can be represented as $\Re(\mathtt{L}) = |\mathtt{L} \cap \mathtt{S}|$ and the number of irrelevant pixels as $\overline{\Re}(\mathtt{L}) = |\mathtt{L} \setminus \mathtt{S}|$. Using these concepts, we can describe the problem of finding the core area as follows.

Core-Area Detection Problem : Given an NFQ $\mathtt{S} \subseteq Z \times Z$ and a maximum tolerable noise level ξ, a square $\mathtt{L} \subseteq Z \times Z$ is the *core area* of the NFQ if it satisfies the following conditions:

1. $\overline{\Re}(\mathtt{L}) \leq \xi$, and
2. for any square $\mathtt{L'} \subseteq Z \times Z$
   (a) $\overline{\Re}(\mathtt{L'}) > \xi$, or
   (b) $\overline{\Re}(\mathtt{L'}) \leq \xi$ and $\Re(\mathtt{L'}) < \Re(\mathtt{L})$, or
   (c) $\Re(\mathtt{L'}) = \Re(\mathtt{L})$ and $\overline{\Re}(\mathtt{L}) \leq \overline{\Re}(\mathtt{L'}) \leq \xi$.

Intuitively, the above definition requires the core area be the largest square subimage with noise no greater than the maximum tolerable level. Furthermore, it contains the least noise among those square subimages with the same dimensions and less noise than the maximum tolerable level.

Although there has been research on related problems (e.g., binary shape decomposition [20, 21]), we are not aware of any solution to the above problem. In [20], for example, 2-D binary shapes are morphologically decomposed into conditionally maximal convex polygons. Each convex polygon component is a subset of the given image, and the union of all such polygons is the original image. Obviously, this is a different problem, in which noise and the sizes of those polygons are not of concern.

4.2 A Straightforward Algorithm

For the sake of clarity, we first consider a straightforward algorithm in this subsection. Clearly, the core area can be found by exhaustively scanning over $\mathtt{S}$. An algorithm based on this strategy is given in Algo. 1. This algorithm is quite expensive. It takes $\mathtt{O}(n^5)$ in the worst case to find the core area. Steps 2(c) is added to reduce the scanning time.

Algorithm 1. *(***Exhaustive Scanning***) Detecting the core area L of an NFQ*

1. *Let MBS be any $n \times n$ minimum bounding square enclosing the NFQ.*
2. *Start with $\mathtt{L}$ of size $\mathtt{d} \times \mathtt{d}$, where $\mathtt{d} = min(\mathtt{n}, \lfloor\sqrt{\xi}\rfloor)$*
3. *While $\mathtt{d} \leq \mathtt{n}$*
   (a) *Let $\mathtt{L'}$ be a square subimage of size $\mathtt{d} \times \mathtt{d}$ centered at some $(x, y) \in MBS$.*
   (b) *For each $(x, y) \in MBS$, let $\mathtt{L}$ be $\mathtt{L'}$ if $\overline{\Re}(\mathtt{L'}) < \xi$.*
   (c) *Enlarge $\mathtt{L}$ as long as $\overline{\Re}(\mathtt{L}) < \xi$. Call the new size of $\mathtt{L}$ to be $\mathtt{d}_1 \times \mathtt{d}_1$.*
   (d) *Let $\mathtt{d}$ be $\mathtt{d}_1 + 1$.*

4.3 Seed-Growing Detection Algorithm (SGDA)

To substantially improve the query processing cost, we present in this subsection a better algorithm to reduce the above complexity by several orders of magnitude. In this algorithm, we first determine a set of good *seed pixels*. For each such pixel, we gradually expand (or grow) an enclosing square subimage, up to the noise limit, to search for the optimal core area. A pixel is a good seed if it has good potential to expand into the optimal core area. In our technique, this potential is computed based on the number of reachable pixels in each of the eight predefined directions as illustrated in Fig. 4(a). For each direction, we count the number of connected relevant pixels until a noise pixel is encountered. These eight counts form an array called the

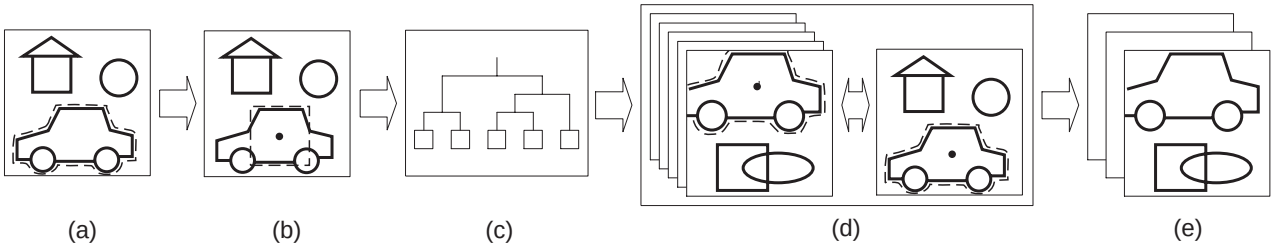

Figure 3: Query Processing

extend of the pixel. We note that we can take advantage of the counts already computed for the adjacent pixels when counting a given pixel as illustrated in Fig. 4(b). Thus, the *extends* of all pixels in the NFQ can be calculated in one pass over the image area using dynamic programming.

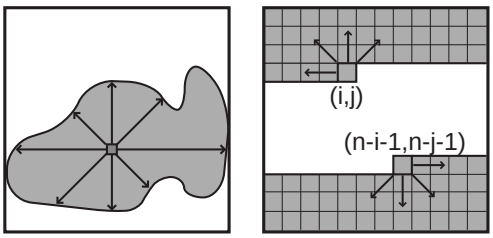

Figure 4: Optimal core area detection - (a) compute the extend by counting the connected pixels in eight predefined directions, (b) using dynamic programming to compute all extends in one pass

Within each `extend` array, the eight counts are recorded in ascending order (i.e., `extend`[0] $<$ `extend`[1] $<\cdots<$ `extend`[7]) to facilitate the following computation:

$$\texttt{potential}_{(x,y)} = \sum_{i=1}^{d} \frac{\texttt{extend}[i] - \texttt{extend}[i-1]}{2^i} + \frac{x + 2 \cdot y}{10^4}, \tag{2}$$

where `extend`[-1] $= 0$.

Using the above equation, we can determine the potential of a given pixel, at location (x, y), as a seed for expanding into the optimal core area. We explain this equation as follows. The first term increases the potential measure by an increment for each round i. The denominator, however, reduces this increment amount by a factor of 2^i due to the following reasons. The eight components of the `extend` array can be seen as the "radii" of eight concentric squares as illustrated in Fig. 5. These squares form eight square bands of pixels. The innermost band is the smallest square with only relevant pixels; and it has full potential to be included in the core area. However, as we move to a band further from the center, the percentage of relevant pixels in this band typically drops. The potential of this band being a part of the core area, therefore, decreases accordingly. This characteristic is captured in our equation by using $1/2^i$ as a weighing factor for round i.

In essence, Eq. (2) roughly estimates the size of the core area seeded at (x, y). We use this estimate as the potential measure of this location as a seed for expanding into the optimal core area. The second term in Eq. (2) is very small. It is included for the convenience of resolving ties when comparing the potential measures of neighboring pixels. Since this term is $\frac{x+2\cdot y}{10^4}$, we favor the pixel with a greater x value and y over x. This choice is arbitrary and has no effect on the outcome.

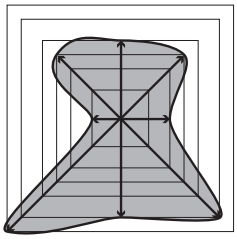

Figure 5: Concentric squares the radii of which are `extend`'s are used to estimate the potential at location (x,y)

Once all the potential measures have been computed, we scan the image area to find candidate pixels with a better potential than all their adjacent pixels. Among these candidates, we select only those pixels with a potential measure greater than a certain threshold to participate in the subsequent enlargement operation. This is done by applying a `SmartExpand` procedure to each of the final candidates. At each step, this procedure expands the square, initially centered at the candidate pixel, toward the area where the least noise is absorbed. This is illustrated in Fig. 6. Although there are four directions for expanding the white square, it is done in the lower-right direction because the least noise is absorbed. This expansion process repeats until the maximum allowable noise is reached or no more relevant pixels could be added. We note that considering only pixels with a better potential than all their immediate neighbors in the final search procedure allows us to substantially reduce the cost of our technique.

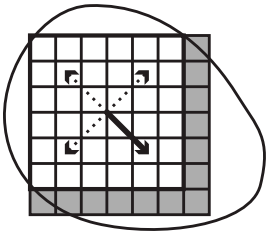

Figure 6: Applying *SmartExpand* to each good seed pixel to search for the core area.

A high-level description of the core-area detection algorithm is given in Algo. 2. A C-code implementation of this algorithm can be found at [1].

Algorithm 2. *(**SGDA**) Detecting the core area L of an NFQ*

1. *Find the minimum bounding square, MBS, that encloses the NFQ.*
2. *For all $(x, y) \in MBS$, compute their* `extend` *in one pass using dynamic programming.*
3. *For all $(x, y) \in MBS$, compute their* `potential` *using Equation (2).*
4. *Scan the pixels in MBS to determine the set of good candidates,* `candidates`*, whose potential is greater than all their adjacent pixels and a predefined threshold.*
5. *Let* `L` *be a minimum square of size* 0×0 .
6. *Process each (x, y), in* `candidates`*, in turn as follows:*
   (a) *Let* `L'` *be a* `extend[0]` $\times$ `extend[0]` *square centered at (x, y).*
   (b) *Apply the* `SmartExpand` *procedure to expand* `L'`*. If the size of* `L'` *is now greater than* `L`*, then let* `L` *be* `L'`*.*

---

5. ANALYSIS AND PERFORMANCE

5.1 Space and Time Complexity

Assuming the size of MBS is $n \times n$, the time complexity of SGDA (Algo. 2) can be broken down as follows:

- The time complexity of Step (2) is $O(n^2)$ using dynamic programming.
- In Step (3), all the potential measures can be computed in $O(n^2)$.
- Step (4) takes $O(n^2)$ since only immediate neighbors are considered.
- Step (5) requires a constant time to execute.
- Step (6) can be completed in $O(n^2 \cdot m)$, where `m` is the number of candidates, each requires no more than $O(n^2)$ time to count the noise pixels.

Thus, the time complexity of SGDA is $O(n^2 \cdot m)$. Since m is generally less than 6 as observed in our experiments, the time complexity of SGDA is $O(n^2)$.

In terms of memory space, it is easy to see that the SGDA algorithm requires $O(n^2)$ space, which is very reasonable. We will provide experimental results in the next section to better illustrate the efficiency of our technique.

5.2 Performance of the Detection Algorithm

Recall that our SGDA algorithm finds the core area of the NFQ by expanding potential seeds. When the number of seeds is large, the time complexity of the detection is high. To determine the number of seeds generated by our algorithm for real data sets, we ran experiments on more than 200 NFQs randomly selected from our database of 16 008 images. The results are plotted in Fig. 7. We can see that for a vast majority of NFQs only a few seeds are generated (one to four seeds). In fact, no NFQ in our experiments yields more than five seeds. This fact demonstrates our proposed seed-searching function is highly effective.

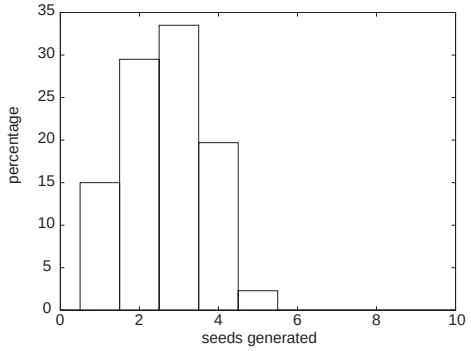

Figure 7: Percentage of Seeds generated

When the number of seeds is small, the reduction in the algorithm's complexity (as opposed to its time constants) has resulted in a tremendous speedup over the exhaustive scanning *at the query time*. As shown in Fig. 8, our detection algorithm can execute from hundreds (at the region-level) to many thousand (at the pixel-level) times faster than the exhaustive scanning algorithm. There is virtually no effect on the overall query processing time, even at the finest level of granularity. At this level, for instance, our detection algorithm takes about 3/10 of a second to detect the core area of an NFQ enclosed in a 256 $\times$ 256 MBS, while the naive algorithm would take more than six hours to process the same NFQ.

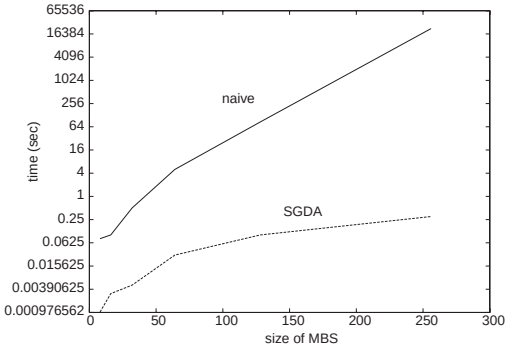

Figure 8: Detection time in log scale

Figure 9 shows an example of detecting the base area of a NFQ by the SGDA algorithm.

6. CONCLUDING REMARKS

Recent content-based image retrieval techniques enable users to eliminate irrelevant regions or noise in formulating queries. By minimizing the effect of noise in similarity measure, the retrieval effectiveness of these approaches is greatly improved for wide range of user-defined queries. To

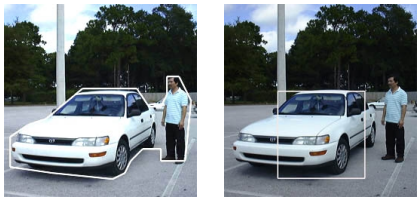

Figure 9: The core area with $\xi = 10\%$ (right) detected by Algo. 2 on an NFQ (left).

support fast initial searches, subimages of a predetermined *base* shape (e.g., circle, polygon) are collected and indexed. At the query time, an area of such a shape enclosing part of the queried objects, called the *core area*, will be identified and used in the initial search of potential candidates before the detailed similarity measure is performed on the original query. To relieve the user from the burden of correctly identifying the core area of a query, we propose an efficient detection algorithm. We formally define this problem and present our seed growing detection algorithm.

We provide a complexity analysis of our algorithm. It shows that we can improve the time complexity by several order of magnitudes while the space requirement is reasonable. To assess the performance of our algorithm, we have implemented the proposed technique on top of SamMatch. Experiments on a large image collection indicate that our algorithm is indeed effective and very efficient.

7. REFERENCES

[1] http://www.cs.ucf.edu/~khanh/code.

[2] P. Andleigh and K. Thakrar. *Multimedia Systems Design*. Prentice Hall, New York, 1996.

[3] N. Beckman, H. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *ACM SIGMOD*, pages 322–331, May 1990.

[4] D. Chetverikov. Detecting regular structures for invariant retrieval. In *Third International Conference on Visual Information Systems*, pages 459–466, 1999.

[5] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3):231-262, 1994.

[6] Y. Gong, H.Chua, and X. Guo. Image indexing and retrieval based on color histogram. In *Proc. of the 2nd Int. Conf. on Multimedia Modeling*, pages 115–126, November 1995.

[7] M. Hiyahara and Y. Yoshida. Mathematical transform of (r, g, b) color data to munsell (h, v, c) color data. In *SPIE Visual Communications and Image Processing*, pages 650–657, 1988.

[8] K. A. Hua, K. Vu, and J. Oh. Sammatch: A flexible and efficient sampling-based image retrieval technique for large image databases. In *Proc. of the 1999 ACM International Multimedia Conference*, pages 225–234, Oct 1999.

[9] J. Huang, S. R. Kumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proc. Computer Vision and Pattern Recognition*, pages 762–768, 1997.

[10] W. Ma and B. Manjunath. Netra: A toolbox for navigating large image databases. *Multimedia System*, 7:184–198, May 1999.

[11] J. Malki, N. Boujemaa, C. Nastar, and A. Winter. Region queries without segmentation for image retrieval by content. In *Third International Conference on Visual Information and Information Systems*, pages 115–122, 1999.

[12] A. Natsev, R. Rastogi, and K. Shim. Walrus: A similarity retrieval algorithm for image databases. In *Proc. of the 1999 ACM SIGMOD Int. Conf. on Management of Data*, pages 395–406, 1999.

[13] N. Nes and M. d'Ornellas. Color image texture idexing. In *Third International Conference on Visual Information Systems*, pages 467–474, 1999.

[14] W. Niblack, R. Barber, W. Equitz, M. Flicker, E. Glasman, D. Petkovic, P. Yanker, and C. Faloutsos. The qbic project: Query images by content using color, texture and shape. In *SPIE V1908*, 1993.

[15] S. Ravela, R. Manmatha, and E. Riseman. Image retrieval using scale-space matching. In *Proc. of the Fourth European Conf. On Computer Vision*, pages 273–282, 1996.

[16] J. R. Smith. *Integrated Spatial and Feature Image Systems: Retrieval, Compression and Analysis*. PhD thesis, Graduate School of Arts and Science, Columbia University, Feb. 1997.

[17] M. Swain. Interactive indexing into image database. In *SPIE V1908*, 1993.

[18] A. Watt and F. Policarpo. *The Computer Image*. ACM Press, New York, 1999.

[19] M. Wood, N. Campbell, and B. Thomas. Iterative refinement by relevance feedback in content-based digital image retrieval. In *The Sixth ACM International Multimedia Conference*, pages 13–20, September 1998.

[20] J. Xu. Morphological decomposition of 2-d binary shapes into conditionally maximal convex polygons. In *Pattern Recognition*, volume 29, pages 1075–1104, 1996.

[21] Y. Zhao and R. Haralick. Binary shape recognition based on an automatic morphological shape decomposition. In *Proc. of Int. Conf. Acoust., Sign. Procesing*, pages 1691–1694, 1989.