The Acoi Algebra: A Query Algebra for Image Retrieval Systems

Niels Nes and Martin Kersten

University of Amsterdam
{niels,mk}@wins.uva.nl

Abstract. Content-based image retrieval systems rely on a query-by-example technique often using a limited set of global image features. This leads to a rather coarse-grain approach to locate images. The next step is to concentrate on queries over spatial relations amongst objects within the images. This calls for a small collection of image retrieval primitives to form the basis of an image retrieval system. The Acoi algebra is such an extensible framework built on the relational algebra. New primitives can be added readily, including user-defined metric functions for searching. We illustrate the expressive power of the query scheme using a concise functional benchmark for querying image databases.

keywords: Image Retrieval, Query Algebra, Features, and IDB.

1 Introduction

With the advent of large image databases becoming readily available for inspection and browsing, it becomes mandatory to improve image database query support beyond the classical textual annotation and domain specific solutions, e.g. [21]. An ideal image DBMS would provide a data model to describe the image domain features, a general technique to segment images into meaningful units and provide a query language to study domain specific algorithms with respect to their precision and recall capabilities. However, it is still largely unknown how to construct such a generic image database system.

Prototype image database systems, such as QBIC [6], WebSeek[18], and PictoSeek [8] have demonstrated some success in supporting domain-independent queries using global image properties. Their approach to query formulation (after restricting the search using textual categories) is based on presenting a small sample of random images taken from the target set and to enable the user to express the query (Q_1) "find me images similar to this one" by clicking one of the images provided. Subsequently the DBMS locates all images using its built-in metrics over the global color distribution, texture, or shape sets maintained.

However, this evaluation technique is bound to fail in the long run for several reasons. First, random sample sets to steer the query process works under the assumption that there is a clear relationship between color, texture and shape and the semantic meaning. This pre-supposes rather small topical image databases and fails when the database becomes large or filled from many sources, such as envisioned for the Acoi image database[1].

Second, the global image properties alone are not sufficient to prune false hits. Image databases are rarely used to answer the query Q_1. Instead, the intended user query (Q_2) is :"find me an image that contains (part of) the one selected" where the containment relationship is expressed as a (predefined) metric over selected spatial and image features or directly (Q_3) :" find me an image that contains specific features or objects using my own metric". In addition to color distribution and texture, spatial information about object locality embedded in the image is needed. A prototype system that addresses these issues is VisualSEEK[18] graphical user interface for the WebSeek image retrieval system.

What are the necessary primitives to express the metric? For example, in a large image database one could be interested to locate all images that contain part of the Coca-Cola logo. This query could be formulated by clipping part of a sample Coca-Cola logo to derive its reddish (R) and white (W) color and to formulate a query of the form:

```
  select display(i)
    from image_region r1,r2, image i
   where distance(r1.avghue, R) < 0.2
     and distance(r2.avghue, W) < 0.2
     and r1 overlaps r2
     and r1,r2 in i
sort by distance(r1.avghue, R), distance(r2.avghue, W)
```

This query uses two primitive parameterized metric functions. The function *distance* calculates a distance in the color space and *overlaps* determines region containment. The former is defined as part of the **color** data type and the latter for the **region** data type.

A challenge for image database designers is to identify the minimal set of features, topological operators, and indexing structures to accommodate such image retrieval queries. In particular, those (indexed) features where their derivation from the source image is time consuming, but still can be pre-calculated and kept at reasonable storage cost. This problem becomes even more acute when the envisioned database is to contain over a million images.

In [13] we introduced an extensible image indexing algorithm based on rectangular segmentation of regions. Regions are formed using similarity measures. In this paper we extended this approach with a query algebra to express queries over the image database.

For such an image retrieval algebra we see three global requirements.

1. The algebra should be based on an extensional relational framework.
2. The algebra should support proximity queries and the computational approach should be configurable by the user.
3. The algebra should be computationally complete to satisfy the wide community of (none-database) image users.

[1] Acoi is the experimental base for the national project on multi-media indexing and search (SION-AMIS project), http://www.cwi.nl/~acoi/Amis

The remainder of this report is organized as follows. In Section 2 we explain our database model, review available region representations and query primitives for image retrieval systems. Also we provide a short introduction to our underlying database system, called Monet. Section 3 explains the query primitives. In Section 4 we define the Acoi Image Retrieval Benchmark, which is the basis for the experimentation reported in section 5. We conclude with an indication of future research topics.

2 Image Databases

This Section introduces the data model for query formulation. The data model is based on regions as the abstraction of the image segmentation process. In section 2.2 we review several region representation methods. In section 2.3 query language extensions for image retrieval are reviewed to provide the background information and to identify the requirements imposed on the image DBMS. Since our image retrieval system is built using the Monet database system we also give a short introduction to Monet.

2.1 Image Database Model

The Acoi database is described by the ODL specification shown in Figure 1.

```
interface Img {
        relationship set < Pix > data inverse Pix::image;
};
interface Pix {
        relationship Img image inverse Img::data;
        relationship Reg region inverse Reg::pixels;
};
interface Reg {
        relationship set < Pix > pixels inverse Pix::region;
        relationship set < Seg > segments inverse Seg::regions;
};
interface Seg {
        relationship set < Reg > regions inverse Reg::segment;
        relationship set < Obj > object inverse Obj::segments;
};
interface Obj {
        relationship set < Seg > segments inverse Seg::object;
};
```

Fig. 1. Data Model

The *data* relationship relates raw pixel information with an image. This is a virtual class, because each pixel is accessed from the image representation upon need. The *region* relationship expresses that each pixel is part of one region only. For the time being we use rectangular regions to simplify implementation and to improve performance. The architecture has been set up to accommandate other (ir-) regular regions, like hexagons and triangulation, as well.

The *segments* mapping combines regions into meaningful units. The segments are typically the result of the image segmentation process, which determines when regions from a semantic view should be considered together. The model does not prescribe that regions are uniquely allocated to segments. A region could be part of several segments and applying different segmentation algorithms may result in identical region sets. The segmentation algorithm used for the current study is based on glueing together regions based on their average color similarity and physical adjacency, details of which can be found in [13].

The relationship object of the segments interface, expresses that segments can form a semantically meaningful object. An example is a set of segments together representing a car.

2.2 Segment Representation

The bulk of the storage deals with region representation, for which many different approaches exist. All have proven to be useful in a specific context, but none is globally perfect. The chain code as described by Freeman [7] encodes the contour of a region using the 8-connected neighborhood directions. Chain codes are used in edge, curve and corner finding algorithms [11]. It is not useful for region feature extraction, since it only represents part of the boundary of an area, no interior. The complexity is $O(p)$ for both storage and performance, where p is the perimeter of the region.

Many boundary representations exist [10], e.g. polygons and functional shape descriptors. Functional shape descriptors use a function to approximate the region boundary. Fourier, fractal and wavelet analysis have been proposed for this [3, 12, 17]. Although these representations have very low storage requirements, i.e. each boundary is represented using a few parameters, they are of limited use aside from shape representation. Recalculation of the regions interior from polygons is very hard and from functional descriptions generally impossible.

Another representation to describe the interior of the region is run length encoding using (position, length) pairs in the scan direction [9]. Diagonal shaped regions are handled poorly by this coding schema.

The pyramid structures [20, 19] represent an region using multiple levels of detail. They are used in image segmentation and object recognition [20, 15]. These structures are very similar to the quad tree [16]. The quad tree is a hierarchical representation, which divides regions recursively into four equal sized elements. The complexity of this structure per region is $O(p + n)$, where the region is located in a $2^n * 2^n$ image and p is again the perimeter of the region. Quad trees can be stored efficiently using a pointerless representation. The quad tree has been used to represent binary images efficiently. The tree needs only to store those regions which have a different color than its parent nodes.

Since none of the structures above solve the regions representation problem, there is a strong need for an extensible framework. It would permit domain specific representations to be integrated into a database kernel, such that scalable image databases and their querying becomes feasible.

To explore this route we use a minimalistic approach, i.e. regions are described by rectangular grids. The underlying DBMS can deal with them in an efficient manner. The domain specific rules and heuristics are initially handled by the query language and its optimizer.

2.3 Image Retrieval Algebra

The image retrieval problem is a special case of the general problem of object recognition. When objects can be automatically recognized and condensed into semantic object descriptors, the image retrieval problem becomes trivial. Unfortunately, object recognition is only solved for limited domains. This calls for an image feature database and a query algebra in which a user can express domain specific knowledge to recognize the objects of interest.

Research on image retrieval algebras has so far been rather limited. The running image retrieval systems support query by example[6] or by sketch [18], only. For example, the interface of the QBIC system lets the user choose for retrieval based on keywords or image features. These systems have a canned query for which only a few parameters can be adjusted. It does not provide a functional or algebraic abstraction to enable the user to formulate a specific request. In the WebSeek Internet demo the user can adjust a color histogram of a sample image to specify the more important colors. However, this interface allows no user defined metric on colors.

Only Photobook [14] allows for user defined similarity metric functions through dynamically loadable C-libraries. Although this approach is a step forward, it is still far from a concise algebraic framework that has boosted database systems in the administrative domain. In section 3 we introduce the components of such an algebra.

2.4 Extensible Database Systems

Our implementation efforts to realize an image database system are focussed on Monet. Monet has been designed as a next generation system, anticipating market trends in database server technology. It relies on a network of workstations with affordable large main memories ($>$ 128 MB) per processor and high-performance processors ($>$ 50 MIPS). These hardware trends pose new rules to computer software – and to database systems – as to what algorithms are efficient. Another trend has been the evolution of operating system functionality towards micro-kernels, i.e. those that make part of the Operating System functionality accessible to customized applications.

Given this background, Monet was designed along the following ideas:

– *Binary relation storage model.* Monet vertically partitions all multi-attribute relationships in Binary Association Tables (BATs), consisting of `[OID,attribute]` pairs. This Decomposed Storage Model (DSM) [5] facilitates table evolution. And it provides a canonical representation for a variety of data models, including an object-oriented model [1]. Moreover, it leads to a simplified database kernel implementation, which enables readily inclusion of additional data types, storage representations, and search accelerators.
– *Main memory algorithms.* Monet makes aggressive use of main memory by assuming that the database hot-set fits into its main memory. For large databases, Monet relies on virtual memory management by mapping files into it. This way Monet avoids introducing code to 'improve' or 'replace' the operating system facilities for memory/buffer management. Instead, it gives advice to the lower level OS-primitives on the intended behavior[2] and lets the MMU do the job in hardware. Experiments in the area of Geographical Information Systems[2] and large object-oriented applications [1] have confirmed that this approach is performance-wise justified.
– *Monet's extensible algebra.* Monet's Interface Language (MIL) is an interpreted algebraic language to manipulate the BATs. In line with extensible database systems, such as Postgres, Jasmine and Starburst, Monet provides a Monet Extension Language (MEL). MEL allows you to specify extension modules to contain specifications of new atomic types, new instance- or set-primitives and new search accelerators. Implementations have to be supplied in C/C++ compliant object code.

3 Algebraic Primitives

Analysis of the requirements encountered in image retrieval and the techniques applied in prototype image systems, such as [6, 18, 8], indicate the need for algebraic operators listed in Table 1. The parameter i denotes an image, p a pixel, r a region, s a segment and o an object. Most functions are overloaded as indicated by a combination of $iprso$.

The first group provides access to the basic features of images, pixels, regions, segments and objects. Their value is either stored or calculated upon need. The Point, Color, Vector and Histogram datatypes are sufficient extensions to the base types supported by the database management system to accommodate the features encountered in practice so far.

The second group defines topological relationships. This set is taken from [4], because there is no fundamental difference between spatial information derived from images and spatial information derived from geographic information systems.

The third group addresses the prime algorithmic steps encountered in algorithms developed in the Image processing community. They have been generalized from the instance-at-a-time behavior to the more convenient set-at-a-time

[2] This functionality is achieved with e.g. `mmap()`, `madvise()`,and `mlock()` Unix system calls.

Properties	
$area(iprso)$	$\rightarrow float$
$perimeter(iprso)$	$\rightarrow float$
$center(iprso)$	$\rightarrow point$
$avg\_color(iprso)$	$\rightarrow color$
$color\_hist(iprso)$	$\rightarrow Histogram$
$texture(iprso)$	$\rightarrow vector$
$moment(iprso)$	$\rightarrow float$
Topological operations	
$touch(prso, prso)$	$\rightarrow boolean$
$inside(prso, prso)$	$\rightarrow boolean$
$cross(prso, prso)$	$\rightarrow boolean$
$overlap(prso, prso)$	$\rightarrow boolean$
$disjoint(prso, prso)$	$\rightarrow boolean$
Join operations	
$F\_join_{f(prso,prso)}(\{prso\}, \{prso\})$	$\rightarrow \{prso\}$
$M\_join_{d(prso,prso),m}(\{prso\}, \{prso\})$	$\rightarrow \{prso\}$
$P\_join_{p(prso,prso)}(\{prso\}, \{prso\})$	$\rightarrow \{prso\}$
Selection operations	
$F\_find_{f(iprso,iprso)}(\{iprso\}, iprso)$	$\rightarrow iprso$
$M\_select_{d(iprso,iprso),m}(\{iprso\}, iprso)$	$\rightarrow \{iprso\}$
$P\_select_{p(iprso,iprso)}(\{iprso\}, iprso)$	$\rightarrow \{iprso\}$
Ranking and Sample operations	
$P\_sort(\{iprso\})$	$\rightarrow \{iprso\}$
$M\_sort_{d(iprso,iprso)}(\{iprso\}, iprso)$	$\rightarrow \{iprso\}$
$N\_sort(\{iprso\})$	$\rightarrow \{iprso\}$
$Top(\{iprso\}, int)$	$\rightarrow \{iprso\}$
$Slice(\{iprso\}, int, int)$	$\rightarrow \{iprso\}$
$Sample(\{iprso\}, int)$	$\rightarrow \{iprso\}$

Table 1. The Image Retrieval Algebra

behavior in the database context. This group differs from traditional relational algebra in stressing the need for θ-like joins and predicates described by complex mathematical formulae.

A *fitness* join (F_join) combines region pairs maximizing a fitness function, $f(rs, rs) \to float$. The pairs found merge into a single segment. The *metric* join (M_join) finds all pairs for which the distance is less than the given maximum m. The distance is calculated using a given metric function, $d(rs, rs) \to float$. The last function in this group, called *predicate* join (P_join), is a normal join which merges regions for which the predicate p holds. An example of such an expression is the predicate "similar", which holds if regions r_1 and r_2 touch and the average colors are no more than 0.1 apart in the domain of the color space. A functional description is:

similar($r\ r_1, r\ r_2$) :=
    $touch(r_1, r_2)$ and
    $distance(r_1.avg\_color, r_2.avg\_color) < 0.1$

The next group of primitives is needed for selection. The fitness find (F_find) returns the region which fits best to the given region, according to fitness function $f(rs, rs)$. The metric select (M_select) returns a set of regions at most at distance m, using the given metric $d(rs, rs)$ function. The predicate select (P_select) selects all regions from the input set for which the predicate is valid.

Join operations	result
$F\_join_f(L, R) \to \{prso\}$	$\{lr \| lr \in LR, \nexists l'r' \in LR \wedge f(l', r') > f(l, r)\}$
$M\_join_{d,m}(L, R) \to \{prso\}$	$\{lr \| lr \in LR \wedge d(l, r) < m\}$
$P\_join_p(L, R) \to \{prso\}$	$\{lr \| lr \in LR \wedge p(l, r)\}$
Selection operations	**result**
$F\_find_f(L, r) \to iprso$	$l \in L, \nexists l' \in L \wedge f(l', r) > f(l, r)$
$M\_select_{d,m}(L, r) \to \{iprso\}$	$\{l \| l \in L \wedge d(l, r) < m\}$
$P\_select_p(L, r) \to \{iprso\}$	$\{l \| l \in L \wedge p(l, r)\}$

Table 2. Signatures of the Join and Selection operations

The last group can be used to sort region sets. We have encountered many algorithms with a need for a partial order. *P_sort* derives a partial order amongst objects. Each entry may come with a weight which can be used by the *metric* sort (*M_sort*). This sort operation is based on a distance metric between all regions in the set and a given region. The N_sort uses a function to map regions onto the domain $\mathcal{N}$.

After the partial order the *Top* returns the top n objects of the ordered table. The *Slice* primitive will slice a part out of such an ordered table. The *Sample* primitive returns a random sample from the input set.

4 Acoi Image Retrieval Benchmark

The next step taken was to formulate a functional benchmark of image retrieval problems. Many such performance benchmarks exist for DBMS for a variety of application areas. Examples in transaction processing are the TP series (TPC-C and TPC-D) and in geographic information systems the SEQUOIA 2000 storage benchmark. We are not aware of similar benchmarks for image retrieval. The construction of such a benchmark is one of the goals of Amis. Both the database and image processing community would benefit from such a public accessible benchmark.[3]

Its function is to demonstrate and support research in image processing and analysis in a database context. Therefore, we derived the following characteristics from the algorithms used in the image processing domain.

- *Large Data Objects* The algorithms use large data objects. Both in terms of base storage (pixels), but also the derived data incurs large space overhead.
- *Complex Data Types* The algorithms use specialized complex data types. Derived data is often stored in special data structures.
- *Fuzzy data* The computational model used is based on heuristics and fuzzy data. This fuzzy data should be accompanied by some form of fuzzy logic.

The Acoi Benchmark Data The data for the benchmark consists of two Image sets, one of 1K images and one of 1M images. The images are retrieved randomly from the Internet using a Web robot. The set contains all kinds of images, i.e. binary and gray scale, small and large but mostly color images.

The Acoi Benchmark Queries Based on the characteristics encountered in the image processing community a set of 6 distinctive queries for the benchmark was identified, which are shown in Table 3.

Query 1 loads the database DB from external storage. This means storing images in database format and calculation of derived data. Since the benchmark involves both global and local image features this query may also segment the images and pre-calculate local image features.

Query 2 is an example of global feature extraction as used in QBIC. This query extracts a normalized color histogram. We only use the Hue component of the Hue, Saturation, Intensity color model. The histogram has a fixed number of 64 bins. In query 3 these histograms are used to retrieve histograms within a given distance and the related images. The histogram h should have 16 none-zero bins and 48 zero. The none-zero bins should be distributed homogeneous over the histogram. The query Q3a sorts the resulting set for inspection.

Query 4 finds the nearest neighboring regions in an image. Near is defined here using a user-defined function, f. This function should be chosen so that neighbors touch and that the colors are as close as possible.

Query 5 segments an input image. Segmentation can also be done with specialized image processing functions, but to show the expressive power of the

[3] Readers can contact authors for a copy of the Acoi Benchmark.

algebra we also include it here in its bare form. Finally Q6 searches for all images in the database which have similar segments as the example image. The resulting list of images is sorted in query 6a.

nr	**query**
Q1	DB-load
Q2	$\{h \mid i \in Ims \land h = normalized\_color\_histogram(i)\}$
Q3	$\{i \mid i \in Ims \land L^2distance(normalized\_color\_histogram(i), h) < 0.1\}$
Q3a	sort Q3
Q4	$\{n_1n_2 \mid n_1n_2 \in Regs(im) \land \nexists n3 \in Regsf(n_1, n_3) > f(n_1, n_2)\}$
Q5	$\{rs \mid rs \subset Regs(i) \land \forall r_1r_2 \in RS :$ $L^2distance(avg\_color(r_1), avg\_color(r_2)) < 0.1$ $\land \exists s_0 \ldots s_n \in rs :$ r touch $s_0 \land$ s_i touch $s_{i+1} \land$ s_n touch $s\}$
Q6	$\{i \mid \forall s_i \in Q6(i) \exists s_e \in Segs(e)$ $d(s_i, s_e) < min\_dist\}$
Q6a	sort Q6

Table 3. Benchmark Queries

The Benchmark Evaluation To compare the results of various implementations of the benchmark we used the following simple overall evaluation scheme. The performance of the Acoi Benchmark against different implementation strategies can be compared using the sum of all query execution times. This way moving a lot of pre-calculation to the DB-load query will not improve performance unless the information stored has low storage overhead and is expensive to recalculate on the fly.

5 Performance Assessment

The benchmark has been implemented in Monet using its extensible features. Details about Monet can be found at {http://www.cwi.nl/~monet}. The DB-load query loads the images using the image import statement into the Acoi_Images set. We only load the images in the system. No pre-calculation has been performed.

The color histogram query (Q2) can be expressed in the Acoi algebra as follows:

```
var Q2 := [normalized_color_histogram](Acoi_Images);
```

The brackets will perform the operation normalized_color_histogram on all images int the Acoi_Images set. It returns a set of a histograms. Q3 uses a

M_select with the L^2 metric. The sorting of Q3a can be done using the M_sort primitive. Query Q4 is implemented in the Acoi algebra using a F_join with the function $f(r_1, r_2)$ defined as follows:

```
$f(r_1, R_2) :=$
        dist( $r_1.color()$, $r_2.color()$) if $r_1$.touch($r_2$)
        max_dist
```

The queries 5 and 6 are implemented by longer pieces of Monet code. The segmentation of query Q5 use an iterative process. This process can make use of the F_join primitive to find the best touching regions based on the color distance, see [13] for full details.

Query Q6 can be solved using a series of M_select calls. For each segment in the example image we should select all Images with similar segments, where similar is defined using the metric given. The intersection of the selected images is the result of query 6. This can be sorted using the M_sort primitive.

The Benchmark Results We run these queries using the small Acoi database of 1K images. The small benchmark fits in main memory of a large workstation. The database size is approximately 1G. We used a sparc ultra II with 128 MB of main memory running the Solaris operating system, to perform the benchmark on. Using the Acoi algebra we could implement the benchmark with very little effort.

The initial results can be found in Table 5. Overall the benchmark took 3468.8 seconds.

In the result we can see that the DB-load query takes more than 80 percent of the overall benchmark result. This unexpected result stems from heavy swapping of the virtual memory management system. Main memory runs out quickly, so swapping will influence the performance. Based on our early experimentation with multi-giga-byte databases this problem can be resolved with some careful loading scripts.

We found that the results of queries Q4 and Q5 were low. The none-optimized current implementation of F_join was responsible for the low performance. To improve it we moved the spatial constrains out of the F_join. This allows us to find candidate pairs based on the spatial relation between regions quickly. This way we improved the performance of the queries Q4 from 5 to 1 second and Q5 from 21 to 1.2 seconds using a few minutes of programming. A similar step in a traditional image programming environment would have meant partly re-coding several pages of c/c++ code.

Query	Time(s)	Query	Time(s)
Q1	2865	Q4	1.0
Q2	598	Q5	1.2
Q3	1.5	Q6	1.5
Q3a	0.3	Q6a	0.3

Table 4. The Acoi Benchmark Results

6 Conclusions

In this paper we introduced an algebraic framework to express queries on images, pixels, regions, segments and objects. We showed the expressive power of the Acoi algebra using a representative set of queries in the image retrieval domain. The algebra allows for user defined metric functions and similarity functions, which can be used to join, select and sort regions. The algebra is extensible with new region properties to accommodate end user driven image analysis in a database context.

We have implemented the algebra within an extensible DBMS and developed a functional benchmark to assess its performance. In the near future we expect further improvement using extensibility in search methods and index structures to improve the performance of the algebra. As soon as the full Acoi database is ready we will perform the benchmark on the set of 1M images.

References

1. P.A. Boncz and M.L. Kwakkel, F. Kersten. High Performance support for OO traversals in Monet. In *BNCOD proceedings*, 1996.
2. Peter A. Boncz, Wilko Quak, and Martin L. Kersten. Monet and its Geographic Extensions: a novel Approach to High Performance GIS Processing. In *EDBT proceedings*, 1996.
3. R. Chellappa and R. Bagdazian. Fourier Coding of Image Boundaries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:102–105, 1984.
4. E. Clementini, P. Felice Di, and P. Oosterom van. A Small Set of Formal Topological Relationships Suitable for End-user Interaction. In *SSD: Advances in Spatial Databases.* LNCS, Springer-Verlag, 1993.
5. G. Copeland and S. Khoshafian. A Decomposed Storage Model. In *Proc. ACM SIGMOD Conf.*, page 268, Austin, TX, May 1985.
6. C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and Effective Querying by Image Content. *Intelligent Information Systems 3*, pages 231–262, 1994.
7. H. Freeman. On the encoding of arbitrary geometric configurations. *Transactions on electronic computers*, 10:260–268, jun 1961.
8. T. Gevers and A. W. M. Smeulders. Evaluating Color and Shape Invariant Image Indexing for Consumer Photography. In *Proc. of the First International Conference on Visual Information Systems*, pages 293–302, 1996.
9. S W Golomb. Run-Length Encodings. *IEEE Transactions on Information Theory 12(3)*, pages 399–401, july 1966.
10. Anil K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1989.
11. Hong-Chih Liu and M. D Srinath. Corner Detection from Chain-Code. *Pattern Recognition(1-2), 1990*, 23:51–68, 1990.
12. B. B. Mandelbrot. *The Fractal Geometry of Nature.* W.H. Freeman and Co., New York, rev 1983.
13. N.J. Nes and M.L. Kersten. Region-based indexing in an image database. *In proceedings of The International Conference on Imaging Science, Systems, and Technology, Las Vegas*, pages 207–215, June 1997.

14. A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. In *SPIE Storage and Retrieval for Image and Video Databases II, No. 2185*, pages 34–47, 1994.
15. E. M. Riseman and M. A. Arbib. Computational Techniques in the Visual Segmentation of Static Scenes. *Computer Graphics and Image Processing*, 6(3):221–276, June 1977.
16. H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison Wesley, 1990.
17. J. Segman and Y. Y. Zeevi. Spherical wavelets and their applications to image representation. *Journal of Visual Communication and Image Representation*, 4(3):263–70, 1993.
18. John R. Smith and Shih-Fu Chang. Tools and Techniques for Color Image Retrieval. In *SPIE Storage and Retrieval for Image and Video Databases IV, No 2670*, 1996.
19. S. L. Tanimoto and T. Pavlidis. A Hierarchical Data Structure for Picture Processing. *Computer Graphics and Image Processing*, 4(2):104–119, June 1975.
20. L. Uhr. Layered recognition cone networks that preprocess, classify, and describe. *IEEE Transactions on Computers*, 21:758–768, 1972.
21. Aref. W.G., Barbara D., and D. Lopresti. Ink as a First-Class Datatype in Multimedia Databases. Multimedia Database Systems, pages 113–160, 1996.