Discovering Association Rules based on Image Content

Carlos Ordonez and Edward Omiecinski *
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

Our focus for data mining in this paper is concerned with knowledge discovery in image databases. We present a data mining algorithm to find association rules in 2-dimensional color images. The algorithm has four major steps: feature extraction, object identification, auxiliary image creation and object mining. Our emphasis is on data mining of image content without the use of auxiliary domain knowledge. The purpose of our experiments is to explore the feasibility of this approach. A synthetic image set containing geometric shapes was generated to test our initial algorithm implementation. Our experimental results show that there is promise in image mining based on content. We compare these results against the rules obtained from manually identifying the shapes. We analyze the reasons for discrepancies. We also suggest directions for future work.

1. Introduction

Discovering knowledge from data stored in typical alphanumeric databases, such as relational databases, has been the focal point of most of the work in database mining. However, with advances in secondary and tertiary storage capacity, coupled with a relatively low storage cost, more and more non standard data (e.g., in the form of images) is being accumulated. This vast collection of image data can also be mined to discover new and valuable knowledge. The problem of image mining [14] combines the areas of content-based image retrieval, image understanding, data mining and databases. This is a first attempt to combine association rules and images, although there has been significant research in image understanding in the Computer Vision community. An initial step towards tapping into the undiscovered wealth of knowledge from mining imagebases is the focus of this paper and more so, whether or not this is feasible.

*This work was supported in part by grant LM 06726-02 from the National Library of Medicine

There is a trend towards mining nonstandard and multimedia data [7]. Digitized images can be considered as a type of multimedia data. Current knowledge discovery technology is far from being able to extract all the knowledge contained in such diverse data types.

As related work to mining image content we can mention the following. There in an interesting prototype from Simon Fraser University called Multimedia Miner [17]. One of its modules is called MM-Associator. This module obtains association rules which are more restricted and simpler than the ones we obtain; these rules relate information about the size, the color and the description of the image, but they do not involve specific objects identified automatically. Another important system used for discovering knowledge in a set of images is the Sky Image Cataloging and Analysis Tool (SKICAT) [6]. This program is used to study astronomical images. SKICAT uses trees and statistical optimization to classify objects obtained from an image segmentation process.

Image mining has two main themes. The first is mining large collections of images and the second is the combined data mining of large collections of image and associated alphanumeric data. As of now we have concentrated on mining only images; but our algorithm can be extended in a straightforward manner to handle images and associated alphanumeric data. An example of the first case might involve a collection of weather satellite imagery of various cities in the United States that has been recorded over an extended period of time. The data mining objective might be to find if there is some pattern that exists for an individual city (over time) or if there is some pattern that exists between different cities. An example of the second case might involve medical imagery and patient (alphanumeric data) records. To develop an accurate diagnosis or prognosis both image data (such as Xrays, SPECT, etc.) and patient data (such as weight, prior health conditions, family history, etc.) can be examined together to find interesting associations.

Our data mining system is built on top of a content-based image retrieval system (CBIR), the "Blobworld" system from the University of California at Berkeley (UCB).

This CBIR system supports object-based queries and thus eliminates the need for the manual indexing of image content. This is a major advantage since manually indexing massive collections of images is impractical and prone to errors. Although CBIR systems are prone to retrieving non-related images.

2. Image mining background

2.1. Motivation

One of the typical data mining problems is to find association rules among data items in a database. In a retail environment such as a grocery store, an association rule might state that *customers who purchase spaghetti and Italian sausage also purchase red wine*. This can be determined by examining all the customer transactions (i.e., purchases). In this case, the data is explicit, there is a specific data item for each of the three grocery items and an individual customer transaction would include a subset of those items and in general a subset of all the items sold by the store. In the case of image-bases, assumming that all the images have been manually indexed (or their contents classified) may not be feasible. This presents one major deviation (problem) from the typical data mining approach for numerical data. If images can efficiently be labelled by a semantic descriptor, then the mining can be done on those high level concepts. However, with hundred's of thousands of images, this will become impossible. An alternative is to rely on automatic/semi-automatic analysis of the image content and to do the mining on the generated descriptors. For example, color, texture, shape and size can be determined automatically. Objects in an image can be determined by the similarity of those attributes. This is the approach we take in this first implementation.

2.2. Content-based Image Retrieval

Content-based image retrieval (CBIR) [10] systems will be needed to effectively and efficiently use large image databases. With a CBIR system, users will be able to retrieve relevant images based on their contents. With CBIR systems, querying is facilitated through generic query classes. Examples of some query classes include color, texture, shape, attributes, text and domain concepts. Color and texture queries allow users to formulate the description of the images to be retrieved in terms of like color and texture. Queries can also be posed with regard to the text associated with the images. For instance, in a medical setting, image retrieval is not only based on image content but also on the physician's diagnosis, treatment, etc. (i.e., additional textual data). We should also point out that CBIR differs from traditional database systems in that images are retrieved based on a degree of similarity and that records are usually retrieved from databases because of exactly matching specified attribute values.

Various content-based retrieval systems (QBIC, Chabot and Photobook) have focused on material-oriented queries and have used low-level image properties (e.g., color and texture) to implement these queries. On the other hand, a content-based retrieval system developed at the University of California at Berkeley [3, 4] focuses on object-oriented queries. That is, queries that search for images that contain particular objects. The approach to object recognition at Berkeley is structured around a sequence of increasingly specialized grouping activities that produces a "blobworld" representation of an image, which is a transformation from the raw pixel data to a small set of localized coherent regions in color and textual space. The "blobworld" representation is based on image segmentation using the Expectation-Maximization algorithm on combined color and texture features.

The salient feature of the Berkeley work is their approach to object recognition. Their approach [3] is based on the construction of a sequence of successively abstract descriptors through a hierarchy of grouping and learning processes. The image descriptors at a low level are color, texture, contrast, polarity, etc. and the grouping is based on spatiotemporal coherence of the local descriptors. The central notion in grouping is coherence and four major issues have been identified in [8] which are segmenting images into coherent regions based on integrated region and contour descriptors; fusing color, texture and shape information to describe primitives; using learning techniques for developing the relationship between object classes and color, texture and shape descriptors; and classifying objects based on primitive descriptors and relationships between primitives.

2.3. Data Mining

Database mining, an important part of knowledge discovery, is defined as the automated discovery of previously unknown, nontrivial, and potentially useful information from databases. The information is a statement that describes the relationship among a set of objects contained in the database with a certain confidence such that the statement is in some sense simpler than enumerating all the relationships between the individual instances of objects [9]. Database mining is the process of generating high-level patterns that have acceptable certainty and are also interesting from a database of facts.

Knowledge discovery derives much of its success from reasoning techniques in artificial intelligence, expert systems, machine learning and statistics. Many paradigms such as inductive learning [15], Bayesian statistics [11], mathematical taxonomy [5], etc, have also been applied to knowledge discovery. In general, knowledge discovery is an amalgamation of concepts from diverse fields.

Efficiency is, in general, important for any computational problem. However, for database mining it also determines whether a particular technique can be applied or not. For example, the number of possible ways to cluster N objects into m clusters in unsupervised learning is exponential in N [12]. Hence, an algorithm which uses exhaustive search for clustering is impractical for real-world databases. In general any algorithm which grows faster than $O(n^2)$ is unlikely to be useful for large databases. Over the years, database systems, mainly relational, have made great strides in improving efficiency. The success of relational database systems in the business community can be attributed to these improvements. Many techniques such as, efficient access methods, buffer management, disk management, etc, are well understood. However, most of these techniques have been developed for on-line transaction processing (OLTP) applications. The access patterns for OLTP applications, which typically access a few hundred records, are considerably different from database mining applications, where entire tables may need to be scanned. One of the challenges in database mining is developing more efficient algorithms, better access structures, optimizing disk I/O, and so on.

3. Image Mining

There are two major issues that will affect the image data mining process. One is the notion of similarity matching and the other is the generality of the application area, that is, the breadth of usefulness of data mining from a practical point of view. For a specific application area, associated domain knowledge can be used to improve the data mining task. Since data mining relies on the underlying querying capability of the CBIR system, which is based on similarity matching, user interaction will be necessary to refine the data mining process.

The essential component in image mining is identifying similar objects in different images. With typical basket-market analysis, the data is usually constrained to a fixed set of items that are explicitly labelled. It is also quite efficient to see if a transaction contains a particular item, i.e., requires an examination of the item labels associated with a transaction. In some cases the data might be preprocessed into a fixed record format where a field exists for each item in the domain and a Boolean value is associated with it, indicating the presence or absence of that item in the transaction. This preprocessing can be done automatically. In a general image mining setting, having a human label every possible object in a vast collection of images is a daunting task. However, we intend to capitalize on the recent work in CBIR, in particular, Blobworld [8].

We built our data mining system on top of a content-based image retrieval system. One premise behind supporting object-based queries in a CBIR system is to eliminate the need for manual indexing of image content. The CBIR system we use is from Berkeley [8]. We will refer to it as the "Blobworld" system. This system produces a "blobworld" representation of each image. A "blob" is just a 2-D ellipse which possesses a number of attributes. An image is made up of a collection of blobs, usually less than ten. Each blob represents a region of the image which is relatively homogeneous with respect to color and texture. A blob is described by its color, texture and spatial descriptors. The descriptors are represented by multidimensional vectors. Most of our limitations stem from the quality of the representation of image content by Blobworld.

3.1. Data mining based on association rules

At this point, we will consider in detail, the problem of finding associations. The problem of generating association rules was first introduced in [1] and an algorithm called *AIS* was proposed for mining all association rules. In [13], an algorithm called *SETM* was proposed to solve this problem using relational operations. In [2], two algorithms called *Apriori* and *AprioriTid* were proposed. These algorithms achieved significant improvements over the previous algorithms. The rule generation process was also extended to include multiple items in the consequent and an efficient algorithm for generating the rules was also presented. In [16], we presented an efficient algorithm for mining association rules that was fundamentally different from prior algorithms. Compared to previous algorithms, our algorithm not only reduced the I/O overhead significantly but also had lower CPU overhead for most cases.

Definitions

- *Association Rule*. An *association rule* is an implication of the form $X \Longrightarrow Y$, where $X, Y \subset \mathcal{I}$, and $X \cap Y = \emptyset$. $\mathcal{I}$ is the set of objects, also referred to as items. X is called the antecedent and Y is called the consequent of the rule. In general, a set of items, such as the antecedent or the consequent of a rule, is called an *itemset*.

- *Support*. Each itemset has an associated measure of statistical significance called *support*. For an itemset $X \subset \mathcal{I}$, *support(*X*)* = s, if the fraction of records in the database containing X equals s.

- *Confidence*. A rule has a measure of its strength called *confidence* defined as the ratio *support(*$X \cup Y$*)* / *support(*X*)*.

Algorithm to mine association rules

The problem of mining association rules is to generate all rules that have support and confidence greater than some user specified minimum support and minimum confidence thresholds, respectively. This problem can be decomposed into the following subproblems:

1. All itemsets that have support above the user specified minimum support are generated. These itemset are called the *large* itemsets. All others are said to be *small*.

2. For each large itemset, all the rules that have minimum confidence are generated as follows: for a large itemset X and any $Y \subset X$, if *support*(X)/*support*($X - Y$) $\geq$ *minimum_confidence*, then the rule $X - Y \implies Y$ is a valid rule.

3.2. Image Mining Algorithm

In this section, we present the algorithms needed to perform the mining of associations within the context of images. The four major image mining steps are as follows:

1. Feature extraction. Segment images into regions identifiable by region descriptors (blobs). Ideally one blob represents one object. This step is also called segmentation.

2. Object identification and record creation. Compare objects in one image to objects in every other image. Label each object with an id. We call this step the preprocessing algorithm.

3. Create auxiliary images. Generate images with identified objects to interpret the association rules obtained from the following step (html page creation).

4. Apply data mining algorithm to produce object association rules.

Here we explain the Image Mining processing in more detail. We keep I/O at a minimum. Images are kept on disk. For feature extraction each image is accessed once. These features are stored in two files, one is an image with all the blobs and the other with the blob descriptors. These blob descriptors are used to build an array with all the features from all the images. Once features are extracted from images we perform object identification using only their blob descriptors; this process is performed entirely in memory. Auxiliary images are kept on disk; these images show each identified object.

Images are not indexed because it is not necessary to search their contents once thay are segmented. Arrays of records are all that are needed to mine images once we have their features. Processing each image is performed independently of each other for feature extraction and this is done sequentially.

Identified objects, object associations and association rules are stored in sequential text files for interpreting results but not for processing. Our program can work with a large number of transactions. It is only limited by the amount of memory occupied by discovered associations.

Segmentation Step

It is not our intention to describe in detail the feature extraction process from the blobworld system. We will rather outline the main steps involved in identifying coherent image regions.

1. Estimate scale color selection σ.

2. Produce 6-dimensional feature vectors. These vectors contain summary information about color and texture only.

3. Produce several clusterings of feature vectors using the Expectation Maximization (EM) method. The 2 dominant colors are determined here. The number of groups in each clustering is called K.

4. Use the Minimum Description Length principle to decide which is the best K.

5. Segment the image into K regions using the spatial grouping of pixels. Each region is connected.

6. Apply a 3x3 max-vote filter to determine dominant colors.

7. Generate blobs with summary information about each region when such region has an area greater than 2% of the image area.

Each blob has the most important information about each region. This information includes color, texture, shape, area and position, but only the first three are considered relevant.

Preprocessing Algorithm

The basic algorithm for finding associations between images/blobs is similar to our association finding algorithm *Partition* [16], as long as we preprocess the image data. By preprocessing the image data, we will identify and label objects contained in the images using the image query processing algorithm [4]. The output of the preprocessing step will be a set of records, $R_1, R_2, \ldots R_k$, one for each image, containing the object identifiers for the objects contained in the image. This step is quite intensive since it is a similarity search between images, actually image descriptors.

INPUT: n segmented images, $\{I_1, I_2, \ldots, I_n\}$,
where I_i is a record containing:
an id number and a blob descriptor vector bd
OUTPUT: n records, $\{R_1, R_2, \ldots, R_n\}$
containing the object identifiers for the blobs

```
FOR i_1 = 1 to n DO
  R_{i_1} = ∅
ENDFOR
object_id = 0
FOR i_1 = 1 TO n − 1 DO
  FOR j_1 = 1 TO size(I_i.bd)
    first_time = true
    FOR i_2 = i_1 + 1 TO n
      IF I_{i_2}.bd_{j_2} is not matched yet THEN
        IF similar(I_{i_1}.bd_{j_1}, I_{i_2}.bd_{j_2}) THEN
          IF first_time THEN
            object_id = object_id + 1
            first_time = false
          ENDIF
          R_{i_1} = R_{i_1} ∪ { object_id }
          R_{i_2} = R_{i_2} ∪ { object_id }
          Mark I_{i_2}.bd_{j_2} as matched
        ENDIF
      ENDIF
    ENDFOR
    Mark I_{i_1}.bd_{j_1} if matched
  ENDFOR
ENDFOR
Filter out unwanted matched objects
```

Figure 1. Preprocessing Algorithm

However, once this is accomplished, the actual data mining step will not require the expensive similarity searching. Our preprocessing algorithm is shown in Figure 1.

First of all we initialize the n records, which store the object id's for each of the n images. This algorithm has four nested loops. These loops are required to compare every blob against the rest in the image database. In this manner we do not miss any potential match. Note that comparisons are always made between blobs in different images. Assuming the blob descriptor vector dimension is bounded, then this algorithm is $O(n^2)$. This is a reasonable asumption since the segmentation step cannot produce a high number of blobs. Nevertheless, if the number of objects in each image can also grow without bound this algorithm is $O(m^2n^2)$ for m the number of possible different objects and n the number of images. This can render the algorithm slow if n and m are similar; in general this algorithm is fairly fast if $m << n$, which is the usual case in practice.

The variable $first\_time$ is used to generate new object id's when one blob is matched for the first time. This is necessary because a single object in one image may be similar to many other objects in the following images and all these objects should have the same id. When one blob turns out to be similar to another one we add the object id to their corresponding records. The first is the record that is being compared against the rest and the second one is the record for which a match was found. The second object in the comparison will be discarded avoiding a future unnecessary comparison. The similarity function to compare blobs is expensive to compute as we will see. So this has an impact on the overall performance of the algorithm.

Each segmented image is treated as a record and its transformation to a set of identified objects will also produce a record. This representation will also give us a direct way to incorporate alphanumeric information associated with the image into the image mining process. The algorithm can handle such information without modification, provided those additional attributes are treated as boolean values.

Similarity Function

The similarity function [4] between two blobs is essential for our image mining program. This function takes four parameters, the two blobs to be compared, a similarity threshold and a vector of standard deviations. The similarity function is mathematically defined as:

$$similarity = e^{-\frac{distance(blob_1, blob_2)}{2}},$$

where

$$distance(blob_1, blob_2) =$$

$$[(blob_1 - blob_2)^T \Sigma^{-1} (blob_1 - blob_2)]^{1/2}.$$

In these formulas $blob_1$ and $blob_2$ are vectors containing summary features and Σ^{-1} represents the vector containing the standard deviations allowed for matching on each desired feature. The -1 power means we divide each distance by the corresponding entry of this vector; this is clarified in Figure 2. This similarity measure is 1 if there is a perfect match on all desired features and approaches zero as the match becomes worse. A low similarity can mean every object is similar to any other object.

It is important to note that the distance for color is computed in a special manner. Colors are stored as three coordinates for a point located in a color-cone, referred as the hue saturation value (hsv). The distance is computed as the minimum pairwise distance of the two dominant colors of the object. Each color is a point in 3-dimensional space affecting its third coordinate by a weight of 0.5 and leaving the first two unchanged. This is computed as a matrix product between the color vector and the weights. This distance constitutes the first entry of the difference vector. For all the

$$INPUT : bd_1, bd_2,\ similarity\_threshold, std\_dev$$
$$OUTPUT : 1\ for\ a\ match,\ 0\ otherwise$$

$$d11 =\ (blob_1.cone_1 - blob_2.cone_1) * [\ 1\ 1\ 0.5]$$
$$d22 =\ (blob_1.cone_2 - blob_2.cone_2) * [\ 1\ 1\ 0.5]$$
$$d1 = |d11|\ +\ abs|d22|$$
$$d12 = (blob_1.cone_1 - blob_2.cone_2) * [\ 1\ 1\ 0.5]$$
$$d21 = (blob_1.cone_2 - blob_2.cone_1) * [\ 1\ 1\ 0.5]$$
$$d2 = abs|d12|\ +\ abs|d21|$$
$$dist_1 = min(d1, d2)$$
$$dist_{2:11}\ =\ blob_1.features\ -\ blob_2.features$$
$$score\ =\ e^{(\ -0.5\ *\ sqrt(\Sigma_{i=1}^{11}((dist_i/std\_dev_i)^2))\ )}$$

$$return\ score\ >=\ similarity\_threshold$$

Figure 2. Similarity function

remaining 10 features the distance is just computed as the difference between each blob entry.

The standard deviation vector permits adjusting parameters for the image mining process to use or discard features in an easy way. If we want to pay close attention to one specific feature we set the standard deviation to a value close to zero, but never zero. If we do not consider some feature to be relevant to the mining process we set the standard deviation to a high value. For most of the features the blobworld system requires standard deviations to be in specific ranges. More specifically standard deviations for color, anisotropy, and contrast require standard deviations at a maximum value of roughly 0.5. For area at most 0.1 is permitted. For all the remaining features any positive standard deviation is legal. If some feature is considered as completely irrelevant an infinity value is assigned to the corresponding entry.

A more detailed description of the similarity function between two blobs is given in Figure 2.

Auxiliary Image Creation

It is important to mention that the auxiliary image creation step is necessary in order for the user to make sense out of the image mining results. We use a web browser as the tool to have an integrated view of images, image features (blobs), object ids, associations, and association rules.

The association rules shown to the user are of the form:

$$\{id_1, id_2 \ldots id_k\} \Longrightarrow \{id_{k+1}, \ldots id_n\}, s = X\%, c = Y\%$$

For each image we show the original image with all the geometric shapes and then one blob image per matched blob. Ideally, each blob should correspond to one shape but this does not always happen as we will discuss. Each of of the blobs is labeled with the *object id* generated by the preprocessing algorithm. These are the id's that appear in the rule. Right now this step is somewhat slow because it involves generating one image per matched blob, but this process is done only once if the image mining program is run several times over the same image set. And also, this step is alleviated by the fact that unmatched blobs are not displayed and thus no image for them is generated.

After showing all the images there are two links to view the association text file and the association rule text file generated by the previous step. These files contain all the associations and rules as well as statistical information that help the user interpret and verify the correctness of the experimental results.

3.3. Example

At this point we show a simple example illustrating how our image mining algorithms work with $n = 10$. The original images and their corresponding blobs are shown on Figure 3. Association rules corresponding to the identified objects are shown on Figure 4. We chose 10 representative images from the image set we created for our experiments. This set of synthetic images is explained in detail in the next section.

In Figure 3 we show the original image at the left with several geometric shapes and white background. These images are labeled with an image id. These images are the *only* input data for our program; no domain knowledge is used. Then we show a series of blob images, each containing one blob. These images are labeled with the id obtained in the preprocessing algorithm. Each blob has a close (most times equal) position to its corresponding geometric shape. There are some cases in which one blob corresponds to several geometric shapes.

For instance in image 004 the object 2 corresponds to the triangle. Object 1 is the background and is eliminated from consideration by filtering out the blobs corresponding to it. Note that in the image 033 the circle has two blobs corresponding to it (6 and 11). Some of the blobs do not correspond to one shape, but to a set of shapes. This the case for blob 8 in image 103 and image 131, or the first blob 9 in image 119. It is important to note that these object identifiers are undesirable but the association rule algorithm eliminates them because they only appear in a couple of cases because their support is low. Also, it is important to note that object 9 is really the ellipse but there are no rules involving 9. Also object 2 is always the triangle except in image 108 in which it also corresponds to an ellipse. It is interesting that the ellipse gets such identifier because this only happened in this case; that is, there is no image in which the ellipse gets 2 and there is no triangle. This is again, a problem arising from the feature extraction step.

As parameters for object identification we set color standard deviation to 0.5, contrast standard deviation to 0.5 and

IMAGE MINING RESULTS

Number of mined images: 10

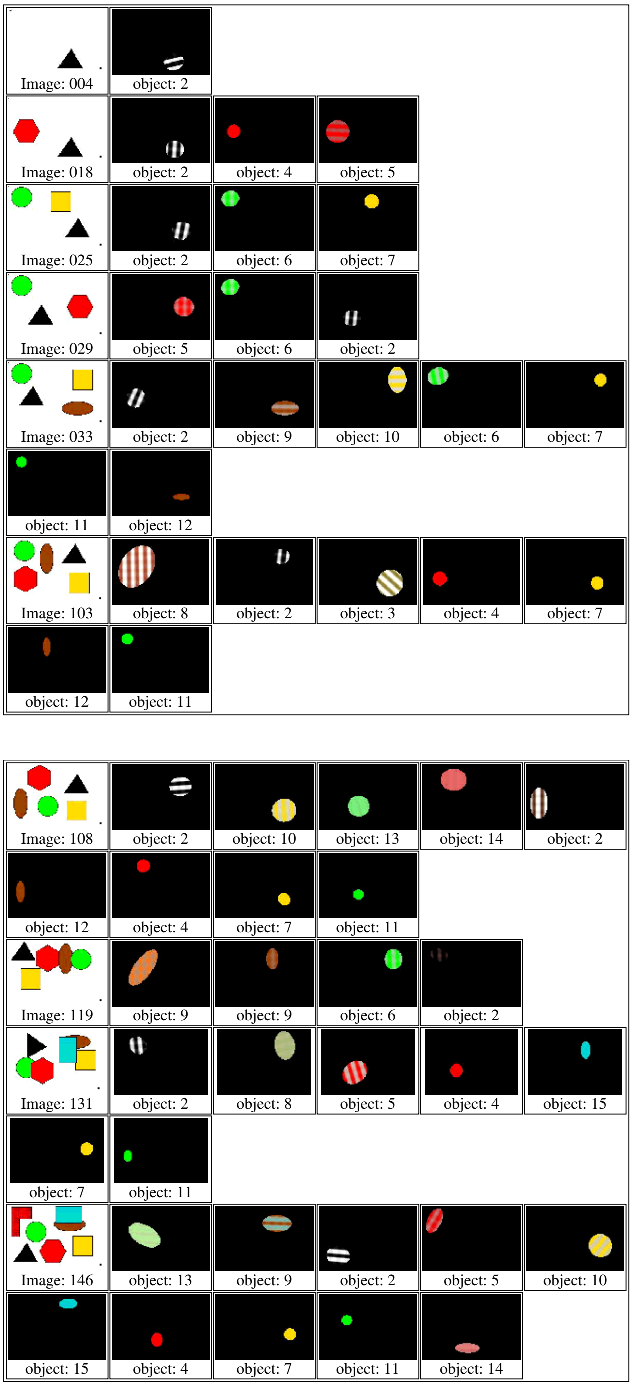

Figure 3. Images and blobs

```
RULES GENERATED

Parameters:
Support:                  30%
Confidence:               70%
Number of records:        10
Number of associations:   41
Support frequency:        3

{ 4 } => { 2 }   s= 0.50  c=1.00   ***    { 4 } => { 7 }   s= 0.40  c=0.80
{ 4 } => { 11 }   s= 0.40  c=0.80         { 4 } => { 2 7 }   s= 0.40  c=0.80
{ 4 } => { 2 11 }   s= 0.40  c=0.80       { 4 } => { 7 11 }   s= 0.40  c=0.80
{ 4 } => { 2 7 11 }   s= 0.40  c=0.80     { 5 } => { 2 }   s= 0.40  c=1.00  ***
{ 5 } => { 4 }   s= 0.30  c=0.75   ***    { 5 } => { 2 4 }   s= 0.30  c=0.75
{ 6 } => { 2 }   s= 0.40  c=1.00          { 7 } => { 2 }   s= 0.60  c=1.00
{ 7 } => { 11 }   s= 0.50  c=0.83         { 7 } => { 2 11 }   s= 0.50  c=0.83
{ 8 } => { 2 }   s= 0.30  c=1.00          { 9 } => { 2 }   s= 0.30  c=1.00
{ 10 } => { 2 }   s= 0.30  c=1.00  ***    { 10 } => { 7 }   s= 0.30  c=1.00  ***
{ 10 } => { 11 }   s= 0.30  c=1.00        { 10 } => { 2 7 }   s= 0.30  c=1.00
{ 10 } => { 2 11 }   s= 0.30  c=1.00      { 10 } => { 7 11 }   s= 0.30  c=1.00
{ 10 } => { 2 7 11 }   s= 0.30  c=1.00    { 11 } => { 2 }   s= 0.50  c=1.00
{ 11 } => { 4 }   s= 0.40  c=0.80         { 11 } => { 7 }   s= 0.50  c=1.00
{ 11 } => { 2 4 }   s= 0.40  c=0.80       { 11 } => { 2 7 }   s= 0.50  c=1.00
{ 11 } => { 4 7 }   s= 0.40  c=0.80       { 11 } => { 2 4 7 }   s= 0.40  c=0.80
{ 12 } => { 2 }   s= 0.30  c=1.00         { 12 } => { 7 }   s= 0.30  c=1.00
{ 12 } => { 11 }   s= 0.30  c=1.00        { 12 } => { 2 7 }   s= 0.30  c=1.00
{ 12 } => { 2 11 }   s= 0.30  c=1.00      { 12 } => { 7 11 }   s= 0.30  c=1.00
{ 12 } => { 2 7 11 }   s= 0.30  c=1.00    { 2 4 } => { 7 }   s= 0.40  c=0.80
{ 2 4 } => { 11 }   s= 0.40  c=0.80       { 2 4 } => { 7 11 }   s= 0.40  c=0.80
{ 2 5 } => { 4 }   s= 0.30  c=0.75        { 2 7 } => { 11 }   s= 0.50  c=0.83
{ 2 10 } => { 7 }   s= 0.30  c=1.00       { 2 10 } => { 11 }   s= 0.30  c=1.00
{ 2 10 } => { 7 11 }   s= 0.30  c=1.00    { 2 11 } => { 4 }   s= 0.40  c=0.80
{ 2 11 } => { 7 }   s= 0.50  c=1.00 ***   { 2 11 } => { 4 7 }   s= 0.40  c=0.80
{ 2 12 } => { 7 }   s= 0.30  c=1.00       { 2 12 } => { 11 }   s= 0.30  c=1.00
{ 2 12 } => { 7 11 }   s= 0.30  c=1.00    { 4 5 } => { 2 }   s= 0.30  c=1.00
{ 4 7 } => { 2 }   s= 0.40  c=1.00        { 4 7 } => { 11 }   s= 0.40  c=1.00
{ 4 7 } => { 2 11 }   s= 0.40  c=1.00     { 4 11 } => { 2 }   s= 0.40  c=1.00
{ 4 11 } => { 7 }   s= 0.40  c=1.00       { 4 11 } => { 2 7 }   s= 0.40  c=1.00
{ 7 10 } => { 2 }   s= 0.30  c=1.00       { 7 10 } => { 11 }   s= 0.30  c=1.00
{ 7 10 } => { 2 11 }   s= 0.30  c=1.00    { 7 11 } => { 2 }   s= 0.50  c=1.00
{ 7 11 } => { 4 }   s= 0.40  c=0.80       { 7 11 } => { 2 4 }   s= 0.40  c=0.80
{ 7 12 } => { 2 }   s= 0.30  c=1.00       { 7 12 } => { 11 }   s= 0.30  c=1.00
{ 7 12 } => { 2 11 }   s= 0.30  c=1.00    { 10 11 } => { 2 }   s= 0.30  c=1.00
{ 10 11 } => { 7 }   s= 0.30  c=1.00      { 10 11 } => { 2 7 }   s= 0.30  c=1.00
{ 11 12 } => { 2 }   s= 0.30  c=1.00      { 11 12 } => { 7 }   s= 0.30  c=1.00
{ 11 12 } => { 2 7 }   s= 0.30  c=1.00    { 2 4 7 } => { 11 }   s= 0.40  c=1.00
{ 2 4 11 } => { 7 }   s= 0.40  c=1.00     { 2 7 10 } => { 11 }   s= 0.30  c=1.00
{ 2 7 11 } => { 4 }   s= 0.40  c=0.80     { 2 7 12 } => { 11 }   s= 0.30  c=1.00
{ 2 10 11 } => { 7 }   s= 0.30  c=1.00    { 2 11 12 } => { 7 }   s= 0.30  c=1.00
{ 4 7 11 } => { 2 }   s= 0.40  c=1.00 *** { 7 10 11 } => { 2 }   s= 0.30  c=1.00
{ 7 11 12 } => { 2 }   s= 0.30  c=1.00

Number of rules generated:   83
```

Figure 4. Association rules for identified objects

anisotropy also to 0.5. The similarity threshold as needed by the similarity function was set to 0.6. We tuned these parameters after several experiments. These parameters maximized the number of associations and decreased the number of undesirable matches. The remaining parameters did not improve object identification for this set of synthetic images so their values were set to infinity.

The data mining algorithm was run with a 30% support and 70% confidence. The output is a set of association rules whose support and confidence are above these thresholds. The 83 rules obtained by the program are shown on Figure 4. The rules we are going to explain here are marked with *** on the Figure 4. We preferred to show the entire output of our program in order to give the reader a truthful assesment of the results.

The first rule $\{4 \Rightarrow 2\}$ tells us that if there is an hexagon there is a triangle; the rule has confidence 1.0 and it is indeed correct. A similar rule is $\{10 \Rightarrow 2\}$ that says that if there is a square then there is a triangle. Note that these two rules have a correct confidence but their support is actually higher. This happens because the hexagon got two identifiers (4 and 5) and the square also got two identifiers (7 and 10). This also originates the problem of having repeated rules since $\{5 \Rightarrow 2\}$ is the same as $\{4 \Rightarrow 2\}$.

Now, looking at larger rules we see that the rule $\{4\ 7\ 11\} \Rightarrow \{2\}$, which says that an hexagon, a square and a circle imply a triangle is right. In fact, there is no image in this example in which these 3 shapes happen together and there is no triangle. A rule which has a lower confidence than 1 is $\{2\ 11\} \Rightarrow \{7\}$. The confidence for this rule is actually lower because image 029 has a triangle and a circle but it does not have a square, as implied by the rule; this happened because the circle in this image was identified as object 6 and was not also identified as object 11.

Some of the rules are redundant as is the case for the rule $\{5 \Rightarrow 4\}$. This rule says an hexagon implies an hexagon. An analog case is the rule for the square $\{10 \Rightarrow 7\}$. This problem is originated from the Blobworld system tht assigns two blobs to the same shape. However, in larger collections of images it may be the case that some rule can be discovered with one blob and not with the other one and therefore this might be helpful.

For this set of images none of the rules shown are false as can be verified. In some cases Their support or confidence are higher or a bit lower because the objects were incorrectly matched; but that difference is not significant. It is important to note that running the program with the lowest possible support (10%) does produce several incorrect rules since any rule valid for one image becomes valid for the entire set.

4. Experimental Results

4.1. Synthetic Image Generation

To test our image mining algorithm we created synthetic images. We used synthetic images as a starting point in showing the feasibility of mining images. Also, with our constrained image set, we can more readily determine the weaknesses and strengths of our approach. These images are 192x128 color JPEG format because the Blobworld software from UCB needed this specific size. It is important to note that this format uses a lossy compression scheme and thus image quality is deteriorated. This was not a problem for our synthetic images, but it may be a problem for images with rich information content, such as photographs.

Our images contain a combination of plain geometric shapes. The seven shapes for our experiments were: triangle, circle, square, rectangle, hexagon, ellipse and an irregular shape similar to the letter L. Each of the shapes had a different uniform color and a black border. The background was always white. The texture for each shape was uniform with one exception, the irregular shape. For technical reasons two additional little objects were added to two opposing corners of each image to delimit its size. These little objects are ignored by the feature extraction step because it discards objects whose area does not represent more than 2% of the total image area.

Each geometric shape has the same size and color in each image where it appears. All shapes have a similar size with respect to each other. However, their position and orientation can differ between images. To make the mining process more interestingand realistic, in some cases we overlapped shapes or placed them very close to each other so they would seem to be part of the same object by the segmentation algorithm.

With the guidelines mentioned above we manually generated 100 basic images and we replicated some or all of these images to obtain larger image sets for our experiments. In Table 1 we show summary information for the images we created. We partitioned images into classes according to their content complexity. Image id's below 100 indicate easy to mine images and image id's greater than 100 mean difficult images, that is, images with complex content. In the easy images we have no more than 4 shapes plus the background. Shapes are in different positions but they are not close to each other and they are not overlapping either. For difficult images we have up to 7 shapes plus the background. In this case shapes overlap and also may be close to each other. This certainly makes the image mining process more difficult. We want to stress that our synthetic images are not as complex as images from the real world. This is a first attempt to mine association rules and thus we created simple images to carefully study the experimental results.

Category	Manual	Automatic
No of associations	63	30
No of rules	330	44
Max association size	6	4
Avg support	0.45	0.43
Avg rule confidence	0.80	0.82

Table 1. Manual versus automatic image content mining

4.2. Quality of results

We should mention that there were no false association rules. It did not happen that an object was incorrectly identified and then a rule was generated with the incorrect identifier. In general when we found a match between two objects they were the same shape. All the incorrect matches are filtered out by the support parameter and then the association rules are generated for objects correctly ideintified. Also, some redundant matches happened because of the blobs that represented several shapes but these matches are filtered out by the rule support.

In Table 1 we present a summary of our experimental results with 100 hundred images. We compare the results obtained by manually identifying objects in each image and then generating association rules from such identifiers (Manual Column) against the results obtained by our current implementation (Automatic Column). Ideally, our image mining algorithm should produce the same results as the manual process. So, the table gives a standpoint to assess the quality of our experimental results. For these 100 images unwanted matches, either incorrect or involving many objects, happened in at most 4 images, and therefore their support was well below the minimum support frequency which was at 30.

These experiments were run using the same parameters for object identification as in our small example with 10 images. The parameters for object identification had the following values. We set color standard deviation to 0.5, contrast standard deviation to 0.5 and anisotropy also to 0.5. The similarity threshold as needed by the similarity function was set to 0.6. We tuned these parameters after several experiments. These parameters maximized the number of associations and decreased the errors in unwanted matches. The association rule program was set to look for rules with a 30% support and 70% confidence.

The background represents an object itself. Since association rules with the background were not interesting for our purposes it was eliminated from consideration by the object identification step. It is important to note that this is done after objects have been identified.

We tuned the object identification step to find similar objects changing values for several parameters in the following manner. The most important features used from each object were color and contrast. We allowed some variance for color (0.5) and the maximum allowed variance for contrast (0.5). The anisotropy helped eliminate matches involving several geometric shapes. We ignored shape, because objects could be partially hidden and rotated. Position was considered unimportant because objects could be anywhere in each image. Anisotropy and polarity were ignored because almost all our shapes had uniform texture. Area was given no weight because objects could be overlapping, and thus their area diminished; this can be useful to make perfect matches when objects are apart from each other.

A few rules had high support. One problem that arose during our experiments was that the same shape could have two different blob descriptors, and these blob descriptors could not be matched with two other descriptors for the same shape in another image. This caused two problems. First, a rule could be repeated because it related the same shapes. Second, a rule did not have enough support and/or confidence and therefore was discarded. So, the rules found were correct and in many cases had an actual higher support and also higher confidence.

To our surprise in some cases there were no object matches because an object was very close to another one or was located in a corner of the image. When two or more objects were overlapping or very close they were identified as a single object. This changed the features stored in the blob. The problem was due to the ellipsoidal shape of the blobs and the fact that when a geometric shape was located in a corner thta changed its anysotropy and polarity descriptors. Given a blob for an object very close to one corner means determining an adequate radius for the blob (i.e., ellipse).

Regular shapes such as the triangle, square and hexagon were easily matched across images. This is a direct consequence of the circular blob representation produced when the image is segmented. In this case neither position nor rotation affect the mining process at all. It was surprising that in some cases there were no matches for the circle; in these cases it was in a corner or some other shape was very close or overlapping. Another important aspect about shape is that we do not use it as a parameter to mine images, but shape plays an important role during the segmentation step. So, shape does affect the image mining results quality.

The rectangle and the ellipse are the next shapes that are easily matched even though we did not use the shape feature. The most complicated shape was the L. In this case a number of factors affected matches. When this shape was overlapped with other shapes a few matches were found because a big blob was generated. Also, orientation changed dominant

# of images	50	100	150	200
1. feature	50292	80777	127038	185080
2. obj identif	210	338	547	856
3. aux image	3847	6911	10756	13732
4. assoc rules	6	3	6	4

Table 2. Measured times in seconds for each Image Mining step with different image set sizes

colors and contrast. When the L was close to another shape its colors were merged making it dissimilar to other L shaped objects. This suggests that irregular shapes in general make image mining difficult.

We worked with color images but it is also possible to use black and white images. Color and texture were important in mining the geometric shapes we created. However, we ignored shape as mentioned above. Shape may be more important for black and white images but more accurate shape descriptors are needed than those provided by the blobs.

4.3. Performance evaluation

We ran our experiments on a Sun Multiprocessor (forge.cc.gatech.edu) computer with 4 processors (each running at 100 MHz) and 128 MB of RAM. The image mining program was written in Matlab and C. The first three steps are performed in Matlab.

The feature extraction process is done in Matlab by the software we obtained from UCB. Object identification and record creation were also done in Matlab by a program developed by us. An html page is created in Matlab to interpret results. The association rules were obtained by a program written in C.

In this section we examine the performance of the various components of the image mining process, as shown in Table 2 for several image set sizes. These times were obtained by averaging the ellapsed times of executing the image mining program five times.

4.4. Running time analysis

Feature extraction, although linear in the number of images, is slow and there are several reasons for this. If image size increases performance should degrade considerably since feature extraction is quadratic in image size. Nevertheless, this step is done only once and does not have to be repeated to run the image mining algorithm several times. Object identification is fast. This is because the algorithm only compares unmatched objects and the number of objects per image is bounded. For our experimental results time for this step scales up well. Auxiliary image creation is relatively slow but its time grows linearly since it is done on a per image basis. The time it takes to find rules is the lowest among all steps. If the image mining program is run several times over the same image set only the times for the *second* and the *fourth* step should be considered since image features already exist and auxiliary images have already been created.

5. Application

Image mining could have an application with real images. The current implementation could be used with a set of images having the following characteristics:

- Homogeneous. The images should have the same type of image content. For instance, the program can give useless results if some images are landscapes, other images contain only people and the remaining images have only cars.
- Simple image content. If the images are complex they will produce blobs difficult to match. Also, the association rules obtained will be harder to interpret. A high number of colors, blurred boundaries between objects, large number of objects, significant difference in object size make the image mining process more prone to errors.
- A few objects per image. If the number of objects per image is greater than 10 then our current implementation would not give accurate results since Blobworld in most cases generates at most 12 blobs per image.
- New information. The image itself should should give information not already known. If all the information about the image is contained in associated alphanumeric data, then that data could be mined directly.

6. Future Work

Results obtained so far look promising but we need to improve several aspects in our research effort. We are currently working on the following tasks.

We also need to analyze images with repeated geometric shapes. If we want to obtain simple association rules this can make our program more general. This can be done without further modification to what is working. However, if we want to mine for more specific rules then we would need to modify our algorithm. For instance, we could try to produce rules like the following: if there are two rectangles and one square then we are likely to find three triangles. The issues are the combinatorial growth of all the possibilities to mine and also a more complex type of condition. We will also study more deeply the problem of mining images with more complex shapes such as the irregular one similar to the letter L.

We need a systematic approach to determine an optimal similarity threshold or at least a close one. A very high threshold means only perfect matches are accepted. On the other hand, a very low similarity threshold may mean any object is similar to any other object. Finding the right similarity threshold for each image type looks like an interesting problem. Right now it is provided by the user but it can be changed to be tuned by the algorithm itself. Also, there are many ways to tune the eleven parameters to match blobs and the optimal tuning may be specific to image type.

There also exists the possibility of using other segmentation algorithms that could perform faster or better feature extraction. It is important to note that these algorithms should give a means to compare segmented regions and provide suitable parameters to perform object matching in order to be useful for image mining. From our experimental results it is clear that this step is a bottleneck for the overall performance of image mining.

We can change the object identification algorithms to generate overlapping object associations using more features. Our algorithm currently generates partititons of objects, that is, if one object is considered similar To another one, the latter one will not be compared again. By generating overlapping associations we can find even more rules. For instance a red rectangular object may be considered similar to another rectangular object and at the same time be similar to another red object. Mining by position is also possible; for instance two objects in a certain position may imply another object to be in some other position. Since the software we are using for feature extraction produces eleven parameters to describe blobs we have 2^{11} possibilites to match objects.

7. Conclusions

We presented a new algorithm to perform data mining on images and an initial experimental and performance study. The positive points about our algorithm to find association rules in images and its implementation include the following. It does not use domain knowledge, it is reasonably fast, it does not produce meaningless or false rules, it is automated for the most part. The negative points include: some valid rules are discarded because of low support, there are repeated rules because of different object id's, unwanted matches because of blobs representing several objects, slow feature extraction step, a careful tuning of several parameters is needed, it does not work well with complex images.

We studied this problem in the context of data mining for databases. Our image mining algorithm has 4 major steps: feature extraction, object identification, auxiliary image creation and identified object mining. The slowest part of image mining is the feature extraction step, which is really a part of the process of storing images in a CBIR system; and is done only once. The next slowest operation is creating the auxiliary blob images which is also done once. Object identification and association rule finding are fairly fast and scale up well with image set size. We also presented several improvements to our initial approach of image mining.

Our experimental results are promising and show some potential for future study. Rules referring to specific objects are obtained regardless of object position, object orientation, and even object shape when one object is partially hidden. Image mining is feasible to obtain simple rules from not complex images with a few simple objects. Nevertheless, it requires human intervention and some domain knowledge to obtain better results.

Images contain a great deal of information, and thus the amount of knowledge that we can extract from them is enormous. This work is an attempt to combine association rules with automatically identified objects obtained from a matching process on segmented images. Although our experimental results are far from perfect we show that it is better to discover some reliable knowledge automatically than not discovering any new knowledge at all.

Acknowledgments

We thank Chad Carson from the University of California at Berkeley for helping us setup the Blobworld system. We also thank Sham Navathe and Norberto Ezquerra for their comments to improve the presentation of this paper.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, DC, May 26-28 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, August 29-September 1 1994.

[3] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Recognition of images in large databases using a learning framework. Technical Report TR 97-939, U.C. Berkeley, CS Division, 1997.

[4] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Region-based image querying. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, 1997.

[5] G. Dunn and B. S. Everitt. *An Introduction to Mathematical Taxonomy*. Cambridge University Press, New York, 1982.

[6] U. Fayyad, , D. Haussler, and P. Storoltz. Mining scientific data. *Communications of the ACM*, 39(11):51–57, November 1996.

[7] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, November 1996.

[8] D. Forsyth, J. Malik, M. Fleck, H. Greenspan, T. Leung, S. Belongie, C. Carson, and C. Bregler. Finding pictures of objects in large collections of images. Technical report, U.C. Berkeley, CS Division, 1997.

[9] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. *Knowledge Discovery in Databases*, chapter Knowledge Discovery in Databases: An Overview, pages 1 – 27. MIT Press, 1991.

[10] V. Gudivada and V. Raghavan. Content-based image retrieval systems. *IEEE Computer*, 28(9):18–22, September 1995.

[11] R. Hanson, J. Stutz, and P. Cheeseman. Bayesian classification theory. Technical Report FIA-90-12-7-01, Artificial Intelligence Research Branch, NASA Ames Research Center, Moffet Field, CA 94035, 1990.

[12] M. Holsheimer and A. Siebes. Data mining: The search for knowledge in databases. Technical Report CS-R9406, CWI, Amsterdam, The Netherlands, 1993.

[13] M. Houtsma and A. Swami. Set-oriented mining of association rules. Technical Report RJ 9567, IBM, October 1993.

[14] C. Ordonez and E. Omiecinski. Image mining: A new approach for data mining. Technical Report GIT-CC-98-12, Georgia Institute of Technology, College of Computing, 1998.

[15] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[16] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules. In *Proceedings of the VLDB Conference*, pages 432 – 444, Zurich, Switzerland, September 1995.

[17] O. R. Zaiane, J. Han, Z. N. Li, J. Y. Chiang, and S. Chee. Multimedia-miner: A system prototype for multimedia data mining. In *Proc. 1998 ACM-SIGMOD Conf. on Management of Data*, June 1998.