

MULTIMEDIA DATA MINING USING P-TREES^{1, 2}

WILLIAM PERRIZO, WILLIAM JOCKHECK, AMAL PERERA, DONGMEI REN, WEIHUA WU, YI ZHANG

North Dakota State University
Fargo, North Dakota 58105
william.perrizo@ndsu.nodak.edu

ABSTRACT

The DataSURG group at NDSU has a long-standing interest in data mining remotely sensed imagery (RSI) for agricultural, forestry and other prediction and analysis applications. A spatial data structure, the Peano count tree, was developed that provided an efficient, lossless, data mining ready representation of the many types of data involved in these applications. This data structure has made possible the mining of multiple very large data sets, including time-sequence of RSI and multimedia land data. The Peano count tree (P-tree) technology provides an efficient way to store and mine images of any format, together with pertinent land data of still other formats.

With the invention of Gene chips and gene expression microarrays (MA data) for use in medicine, plant science and many other application areas, new multimedia data mining challenges appeared. MA data presents a one-time, gene expression level map of thousands of genes subjected to hundreds of conditions. An important multimedia plant science application of the near future is to integrate macro-scale analysis of RSI with the micro-scale analysis of MA and to do the latter across multiple organisms. Most of the MA research has been done for a particular organism and the results have been archived as text abstracts (e.g., Medline abstracts). It will therefore be necessary to combine text mining with most multimedia RSI and MA mining. This is truly a multimedia data mining setting. The way text is almost always mined today is to extract pertinent features into tables and to then mine the tables (i.e., extract structured records from the unstructured text first). P-trees are a convenient technology to mine all media involved in this research.

In fact, in almost all multimedia data mining applications, feature extraction converts the pertinent data to relational or tabular form, and then the tuples or rows are

data mined. If multi-medias are going to be mined by first converting to a common format or media, a good candidate common data structure for that purpose is the P-tree. The P-tree data structure is designed for just such a data mining setting.

Keywords

Spatial - Temporal Data Mining, Multimedia, P-tree

1 INTRODUCTION

Data mining often involves handling large volumes of data. However, over the years the concept of what was a large volume of data has evolved. Problems that simply were considered intractable are now taken on with optimism. Spatial-temporal data and other multimedia data are examples where data mining is beginning to be effectively applied.

The DataSURG group at NDSU came to data mining from the context of evaluation of remotely sensed images for use in agricultural applications. These projects involved evaluation of remote imagery of agricultural fields combined with other data sets to produce yield projections. A typical data set might be composed of 1.7 million grid points in a field, each with up to 6 values associated with it.

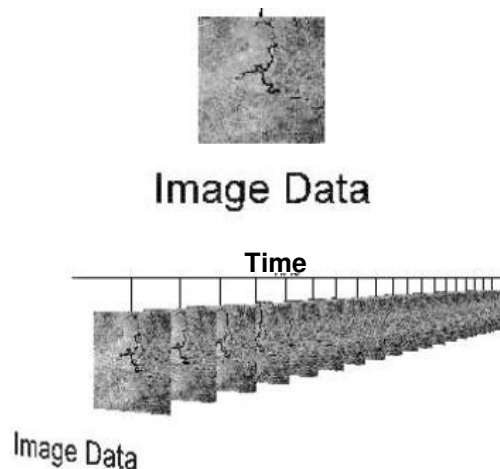


Figure 1: Image data sequenced in the time dimension

¹ Patents are pending on the bSQ and P-tree technology.

² This work is partially supported by GSA Grant ACT# K96130308, NSF Grant OSR-9553368 and DARPA Grant DAAH04-96-1-0329.

Initially these sets were considered large but advances in computer technology and the development of P-tree technology made the sets easily manageable. As more and more data was incorporated the concept of mining sequences of these images developed.

These tools that had been applied to layers of data from different sources are now being viewed as a way to handle sequences of large data sets as they arrive. These data sets do not need to be images but can be stored using the same structures to expedite access.

The purpose of this paper then is to establish that the techniques originally developed for RSI data can provide a major contribution to multimedia data mining. To this end the paper first examines several multimedia data mining approaches to determine their common elements. This element is the production of high dimensional, sparse feature space. This common factor provides the opportunity to use the P-tree technology that is then presented. The use of this technology provides a method to apply multiple data mining techniques to the feature space.

1.1 Multimedia Data Mining

Multimedia data mining is the mining of high-level multimedia information and knowledge from large multimedia databases [10]. It includes the construction of multimedia data cubes which facilitate multiple dimensional analysis of multimedia data and the mining of multiple kinds of knowledge, including summarization, classification and association.

The common characteristic in many data mining applications, including many multimedia data mining applications is that, first, specific features of the data are captured as feature vectors or tuples in a table or relation and then tuple-mined.

There are some examples of multimedia data mining systems. IBM's Query by image content [10] and MIT's Photo book extract image features such as color histograms hues, intensities, shape descriptors, as well as quantities measuring texture. Once these features have been extracted, each image in the database may now be thought of as a point in this multidimensional feature space (one of the coordinates might, for the sake of a simplistic example, correspond to the overall intensity of red pixels, and so on).

Another example is MultiMediaMiner [10]. MultiMediaMiner is a system prototype for multimedia data mining which applies multi-dimension database structures, attribute-oriented induction, multi-level association analysis, statistical data analysis, and machine learning approaches for mining different kinds of rules in relational databases and data warehouses. The system contains 4 major components: image excavator for the extraction of images and videos from multimedia

repositories, a processor for the extraction of image features and storing precomputed data in database, a user interface, and a search kernel for matching queries with image and video feature in the database.

1.1.1 Video-Audio Data Mining

The high dimensionality of the feature spaces and the size of the multimedia datasets make meaningful multimedia data summarization a challenging problem. Video-Audio data mining and other multimedia data mining often involves a preliminary feature extraction step in which the pertinent data is formed into a relation of tuples or possibly time series of tuples, each tuple describing specific selected features of a "frame". P-tree provides a common structure for multi-media data set, which facilitates multimedia data mining.

The process of audio-video multimedia data mining goes as follows:

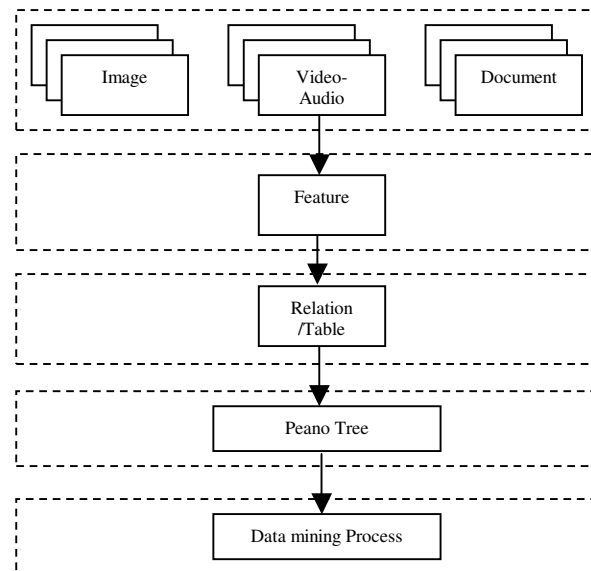


Figure 2 process of video-audio multimedia data mining

For example, performing face recognition from video sequences, involves first extracting specific face geometry attributes (e.g., relative position of nose, eyes, chinbones, chin, etc.) and then forming a tuple of those geometric attributes. Faces are identified by comparing face-geometric features with those stored in a database for known individuals. Partial matches allow recognition even if there are glasses, beards, weight changes, etc. There are many applications of face recognition technology including surveillance, digital library indexing, secure computer logon, and airport and banking security [15].

Another multimedia data mining example is voice biometrics [15]. It relies on human speech, one of the primary modality in human-to-human communication, and provides a non-intrusive method for authentication. By extracting appropriate features from a person's voice and

forming a vector or tuple of these features to represent the voiceprint, the uniqueness of the physiology of the vocal tract and articulator properties can be captured to high degree and used very effectively for recognizing the identity of the person.

1.1.2 Text mining

Text mining can find useful information from unstructured textual information like letters, emails and technical documents. But these kinds of unstructural textural information are not ready for data mining. [8]

Text mining generally involves the following two phases:

1. Preparation phase: document representation
2. Processing phase: clustering or classification

In order to apply data mining algorithms to text data, a weighted feature vector is typically used to describe a document. These feature vectors contain a list of the main themes or keywords or wordstems, along with a numeric weight indicating the relative importance of the theme or term to the document as a whole [9]. The feature vectors are usually highly dimensional, but sparsely populated [8]. P-trees are well suited for representing such feature vector sets. After the mapping of documents to feature vector tables or relations, we can perform document classification in either of two ways: tuple clustering or tuple classification.

1.2 Multimedia Summary

In summary, the key point of this discussion is that a large volume of multimedia data is typically preprocessed into some sort of representation in a high dimension feature space. These feature spaces usually take the form of a table or relation. The data mining of multimedia data then becomes a matter of row or tuple mining (clustering or classification) of the feature tables or relations. While this paper does not propose new techniques for the process of feature extraction, but does propose a new approach to the storage and processing of the feature space, once it is created. Good multimedia representations and formats can help a lot. In the next section of this paper, we describe a technology for storing and mining multimedia feature spaces efficiently and accurately.

2 Peano Count Trees (P-trees)

In this section, we discuss a data structure, called the Peano Count Tree (or P-tree), and its algebra and properties. First, we note again that in most multimedia data mining applications, feature extraction is used to convert the raw multimedia data to relational or tabular form, and then the tuples or rows are data mined. The P-

tree data structure is designed for just such a data mining setting. P-trees provide a lossless, compressed, data mining-ready representation of the relational data set [7].

Given a relational table (with ordered tuples or rows), the data can be organized in different formats. BSQ, BIL and BIP are three typical formats. The Band Sequential (BSQ) format is similar to the relational format. In BSQ format, each attribute is stored as a separate file and each individual band uses the same tuple ordering. Thematic Mapper (TM) satellite images are in BSQ format. For images, the Band Interleaved by Line (BIL) format stores the data in line-major order, i.e., the first row of all bands, followed by the second row of all bands, and so on. SPOT images, which come from French satellite platforms, are in BIL format. Band Interleaved by Pixel (BIP) is a pixel-major format. Standard TIFF images are in BIP format.

We propose a new generalization of BSQ format called bit Sequential (bSQ), to organize any relational data set with numerical values [7]. We split each attribute into separate files, one for each bit position. There are several reasons why we use the bSQ format. First, different bits make different contributions to the values. In some applications, the high-order bits alone provide the necessary information. Second, the bSQ format facilitates the representation of a precision hierarchy. Third, bSQ format facilitates compression. P-trees are basically quadrant-wise, Peano-order-run-length-compressed, representations of each bSQ file. Fast P-tree operations, especially fast AND operation, provide the possibilities for efficient data mining.

In Figure 3, we give a very simple illustrative example with only two bands in a scene having only four pixels (two rows and two columns). Both decimal and binary reflectance values are given. We can see the difference of BSQ, BIL, BIP and bSQ formats.

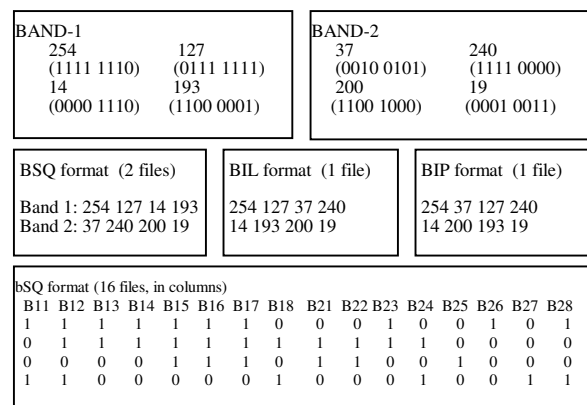


Figure 3 BSQ, BIP, BIL and bSQ formats for a two-band 2x2 image

2.1 Basic P-trees

In this subsection we assume the relation is the pixel relation of an image so that there is a natural notion of rows and columns. However, for an arbitrary relations or table, we can consider the row order to be Peano order (in 1-D, 2-D, 3-D or higher dimensions) and achieve the very same result. Using an X-Y image is the simplest setting in which to introduce the idea of P-trees.

Given a Relation that has been decomposed into bSQ format, we reorganize each bit file of the bSQ format into a tree structure, called a Peano Count Tree (P-tree). The idea is to recursively divide the entire image into quadrants and record the count of 1-bits for each quadrant, thus forming a quadrant count tree [7]. P-trees are somewhat similar in construction to other data structures in the literature (e.g., Quadrees [3, 4, 5] and HHcodes [6]).

For example, given a 8x8 bSQ file (one-bit-one-band file), its P-tree is as shown in Figure 4.

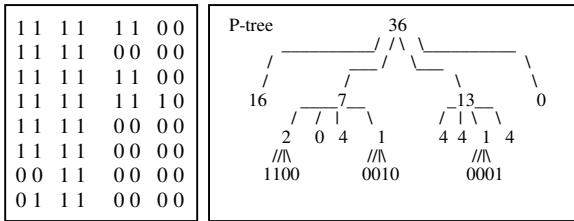


Figure 4 P-tree for a 8x8 bSQ file

In this example, 36 is the number of 1's in the entire image, called root count. This root level is labeled level 0. The numbers 16, 7, 13, and 0 at the next level (level 1) are the 1-bit counts for the four major quadrants in raster order. Since the first and last level-1 quadrants are composed entirely of 1-bits (called pure-1 quadrants) and 0-bits (called pure-0 quadrants) respectively, sub-trees are not needed and these branches terminate. This pattern is continued recursively using the Peano or Z-ordering (recursive raster ordering) of the four sub-quadrants at each new level. Eventually, every branch terminates (since, at the "leaf" level all quadrants are pure). If we were to expand all sub-trees, including those for pure quadrants, then the leaf sequence would be the Peano-ordering of the image. The Peano-ordering of the original image is called Peano Sequence. Thus, we use the name Peano Count Tree for the tree structure above.

The fan-out of a P-tree need not be fixed at four. It can be any power of 4 (effectively skipping levels in the tree). Also, the fan-out at any one level need not coincide with the fan-out at another level. The fan-out pattern can be chosen to produce maximum compression for each bSQ file. We use P-Tree-r-i-l to indicate the fan-out pattern, where r is the fan out of the root node, i is the fan out of all internal nodes at level 1 to L-1 (where root has level L, and

leaf has level 0), and l is the fan out of all nodes at level 1. We have implemented P-Tree-4-4-4, P-Tree-4-4-16, and P-Tree-4-4-64.

Definition 1: A basic P-tree $P_{i,j}$ is a P-tree for the j^{th} bit of the i^{th} band i . The complement of basic P-tree $P_{i,j}$ is denoted as $P_{i,j}^c$ (the complement operation is explained below). For each band (assuming 8-bit data values, though the model applies to data of any number bits), there are eight basic P-trees, one for each bit position. We will call these P-trees the basic P-trees of the spatial dataset. We will use the notation, $P_{b,i}$ to denote the basic P-tree for band, b and bit position, i . There are always $8n$ basic P-trees for a dataset with n bands. P-trees have the following features:

- P-trees contain 1-counts for every quadrant.
- The P-tree for any sub-quadrant at any level is simply the sub-tree rooted at that sub-quadrant.
- A P-tree leaf sequence (depth-first) is a partial run-length compressed version of the original bit-band.
- Basic P-trees can be combined to reproduce the original data (P-trees are lossless representations).
- P-trees can be partially combined to produce upper and lower bounds on all quadrant counts.

P-trees can be used to smooth data by bottom-up quadrant purification (bottom-up replacement of mixed counts with their closest pure counts).

P-trees can be generated quite quickly and can be viewed as a "data mining ready" and lossless format for storing spatial or any relational data.

2.2 P-tree variations

A variation of the P-tree data structure, the Peano Mask Tree (PM-tree, or PMT), is a similar structure in which masks rather than counts are used. In a PM-tree, we use a 3-value logic to represent pure-1, pure-0 and mixed quadrants (1 denotes pure-1, 0 denotes pure-0 and m denotes mixed). The PM-tree for the previous example is also given below. PMT requires less storage compared to PCT. PCT has the advantage of being able to provide the 1 bit count without traversing the tree. Since a PM-tree is just an alternative implementation for a Peano Count tree (PCT), we will use the term "P-tree" to cover both Peano Count tree (PCT) and Peano Mask tree (PMT).

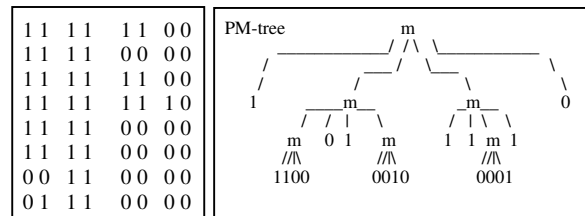


Figure 5. PM-tree

Other useful variations include P1-tree and P0-Tree. These are examples of a class of P-trees called **Predicate Trees**. Given a any quadrant predicate (a condition that is either true or false with respect to each quadrant), we use 1 to indicate true and 0 to indicate false for each quadrant at each level. The P1-tree (predicate is *pure-1*) and P0-tree of the example are.

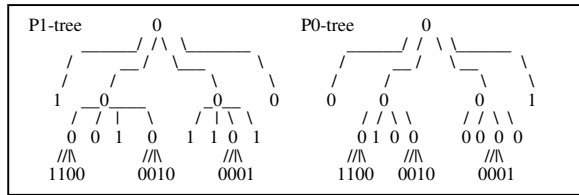


Figure 6 P1-tree and P0-tree

The predicate can be *not-pure-0* (NP0-tree), *not-pure-1-tree* (NP1-tree), etc.

A logical P-tree algebra including complement, AND and OR. The complement of a basic P-tree can be constructed directly from the P-tree by simply complementing the counts at each level (subtracting from the pure-1 count at that level), as shown in the example below. Note that the complement of a P-tree provides the 0-bit counts for each quadrant. P-tree AND/OR operations are also illustrated also.

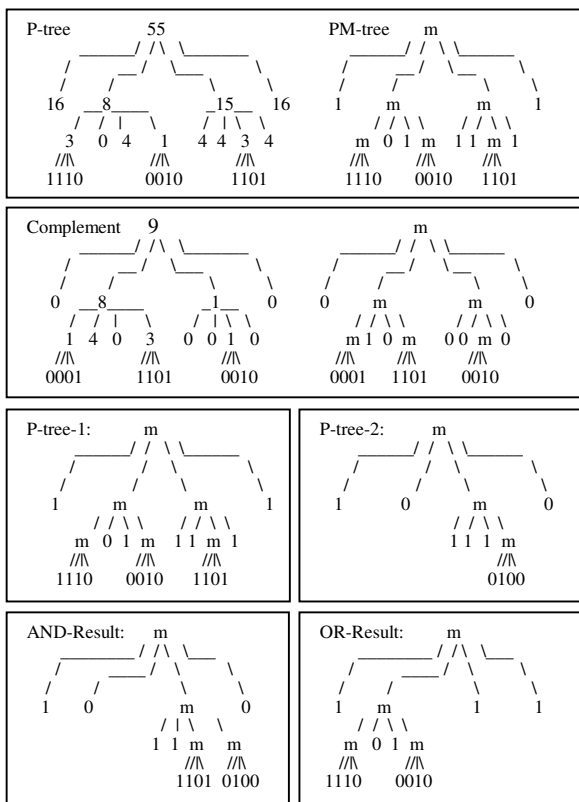


Figure 7. P-tree Algebra (Complement, AND, OR)

AND is the most important operation. The OR operation can be implemented in the very similar way. Below we will discuss various options to implement P-tree ANDing.

2.3 Level-wise P-tree ANDing

ANDing is a very important and frequently used operation for P-trees. There are several ways to perform P-tree ANDing. First let's look at a simple way. We can perform ANDing level-by-level starting from the root level. Table 1 gives the rules for performing P-tree ANDing. Operand 1 and Operand 2 are two P-trees (or sub-trees) with root X_1 and X_2 respectively. Using PM-trees, X_1 and X_2 could be any value among 1, 0 and m (3-value logic representing pure-1, pure-0 and mixed quadrant). Rules for P-tree ANDing are given in Table 1. For example, to AND a pure-1 P-tree with any P-tree will result in the second operand; to AND a pure-0 P-tree with any P-tree will result in the pure-0 P-tree. It is possible to ANDing two m's results in a pure-0 quadrant if their four sub-quadrants result in pure-0 quadrants.

Operand 1	Operand 2	Result
1	X_2	Sub-tree with root X_2
0	X_2	0
X_1	1	Sub-tree with root X_1
X_1	0	0
m	m	0 if four sub-quadrants result in 0; Otherwise m

Table 1 P-tree AND rules

2.4 P-tree AND using Pure-1 paths

In the following algorithm, we will assume P-trees are coded in a compact, depth-first ordering of the paths to each pure-1 quadrant. We use a hierarchical quadrant id (Qid) scheme below to identify quadrants. At each level, we append a sub-quadrant id number (0 means upper left, 1 upper right, 2 lower left, 3 lower right).

0	100	101	11
	102	103	
		12	13
2	3		

Figure 8 Quadrant id (Qid)

For a spatial data set with 2^n -row and 2^n -column, there is a mapping from raster coordinates (x, y) to Peano coordinates (called quadrant ids or Qids). If x and y are expressed as n -bit strings, $x_1x_2\dots x_n$ and $y_1y_2\dots y_n$, then the mapping is $(x, y)=(x_1x_2\dots x_n, y_1y_2\dots y_n) \rightarrow (x_1y_1 \cdot x_2y_2 \dots \cdot x_ny_n)$. Thus, in an 8 by 8 image, the pixel at $(3,6) = (011,110)$ has quadrant id $01.11.10 = 1.3.2$. For simplicity, we wrote the Qid as 132 instead of 1.3.2.

An example is given in below. Each path is represented by the sequence of quadrants in Peano order, beginning just below the root. Since a quadrant will be pure-1 in the result only if it is pure-1 in both/all operands, the AND is done as follows: scan the operands; output matching pure-1 paths.

The AND operation is effectively the pixel-wise AND of bits from bSQ files or their complement files. However, since such files can contain hundreds of millions of bits, shortcut methods are needed. Implementations of these methods have been done which allow the performance of an n -way AND of Tiff-image P-trees (1320 by 1320 pixels) in a few milliseconds. We discuss such methods later in the paper. The process of converting data to P-trees is also time consuming unless special methods are used. For example, our methods can convert even a large TM satellite image (approximately 60 million pixels) to its basic P-trees in just a few seconds using a high performance PC computer. This is a one-time process.

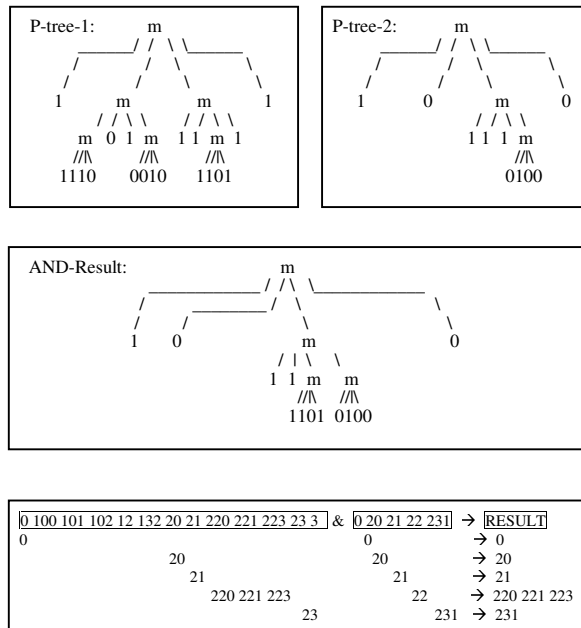


Figure 9 P-tree AND using pure-1 path

2.5 Value and Tuple P-trees

By performing the AND operation on the appropriate subset of the basic P-trees and their complements, we can construct P-trees for values with more than one bit.

Definition: A **value P-tree** $P_i(v)$, is the predicate P-tree for value equality with v at band i (v can be in 1-bit to 8-bit precision).

Value P-trees can be constructed by ANDing basic P-trees or their complements. For example, value P-tree $P_i(110)$ gives the count of pixels with band- i bit 1 equal to 1, bit 2 equal to 1 and bit 3 equal to 0, i.e., with band- i value in the range of [192, 224). It can be constructed from the basic P-trees as:

$$P_i(110) = P_{i,1} \text{ AND } P_{i,2} \text{ AND } P_{i,3}'$$

P-trees can also represent data for any value combination from any band, even the entire tuple. In the very same way, we can construct **tuple P-trees**.

Definition: A **tuple P-tree** $P(v_1, v_2, \dots, v_n)$, is the predicate P-tree for equality with (v_1, v_2, \dots, v_n) for $i=1..n$. We have,

$$P(v_1, v_2, \dots, v_n) = P_1(v_1) \text{ AND } P_2(v_2) \text{ AND } \dots \text{ AND } P_n(v_n)$$

If value v_j is not given, it means it could be any value in Band j . For example, $P(110, ,101,001, , , ,)$ stands for a tuple P-tree of value 110 in band 1, 101 in band 3 and 001 in band 4 and any value in any other band.

Definition: An **interval P-tree** $P_i(v_1, v_2)$, is the predicate P-tree for band- i membership in the interval of $[v_1, v_2]$. We have,

$$P_i(v_1, v_2) = \text{OR } P_i(v), \text{ for all } v \text{ in } [v_1, v_2].$$

Definition: A **box P-tree** $P(l_1, h_1, \dots, l_n, h_n)$, is the predicate P-tree for membership in the box, $[l_1, h_1] \times \dots \times [l_n, h_n]$. We have,

$$P(l_1, h_1, \dots, l_n, h_n) = \text{AND } P_i[l_i, h_i], \text{ for } i=1..n.$$

Any predicate P-tree can be constructed by performing one multi-way AND of the appropriate basic P-trees and their complements (and possible an OR operation).

3 PROPERTIES OF P-TREES

In this section, we will discuss the good properties of P-trees. We will use the following notations:

$p_{x,y}$ is the pixel with coordinate (x, y) , $V_{x,y,i}$ is the value for the band i of the pixel $p_{x,y}$, $b_{x,y,i,j}$ is the j^{th} bit of $V_{x,y,i}$ (bits are numbered from left to right, $b_{x,y,i,0}$ is the leftmost bit). Indices: x : column (x -coordinate), y : row (y -coordinate), i : band, j : bit.

For any P-trees P, P_1 and P_2 , $P_1 \& P_2$ denotes P_1 AND P_2 , $P_1 | P_2$ denotes P_1 OR P_2 , $P_1 \oplus P_2$ denotes P_1 XOR P_2 , P' denotes COMPLEMENT of P .

$P_{i,j}$ is the basic P-tree for bit j of band i , $P_i(v)$ is the value P-tree for the value v of band i , $P_i(v_1, v_2)$ is the interval P-tree for the interval $[v_1, v_2]$ of band i , $rc(P)$ is the root count of P-tree P . P^0 is pure-0 tree, P^1 is pure-1 tree. N is the number of pixels in the image or space under consideration.

Lemma 1: For any two P-trees P_1 and P_2 , $rc(P_1 | P_2) = 0 \Rightarrow rc(P_1) = 0$ and $rc(P_2) = 0$. More strictly, $rc(P_1 | P_2) = 0$, if and only if $rc(P_1) = 0$ and $rc(P_2) = 0$.

Proof: (Proof by contradiction) Let, $rc(P_1) \neq 0$. Then, for some pixels there are 1s in P_1 and for those pixels there must be 1s in $P_1 | P_2$ i.e. $rc(P_1 | P_2) \neq 0$. But we assumed $rc(P_1 | P_2) = 0$. Therefore $rc(P_1) = 0$. Similarly we can prove that $rc(P_2) = 0$.

The proof for the inverse, $rc(P_1) = 0$ and $rc(P_2) = 0 \Rightarrow rc(P_1 | P_2) = 0$ is trivial. This immediately follows the definitions.

Lemma 2:

- a) $rc(P_1) = 0$ or $rc(P_2) = 0 \Rightarrow rc(P_1 \& P_2) = 0$
- b) $rc(P_1) = 0$ and $rc(P_2) = 0 \Rightarrow rc(P_1 \& P_2) = 0$.
- c) $rc(P^0) = 0$
- d) $rc(P^1) = N$
- e) $P \& P^0 = P^0$
- f) $P \& P^1 = P$
- g) $P | P^0 = P$
- h) $P | P^1 = P^1$
- i) $P \& P' = P^0$
- j) $P | P' = P^1$

Proofs are immediate.

Lemma 3: $v_1 \neq v_2 \Rightarrow rc\{P_i(v_1) \& P_i(v_2)\} = 0$, for any band i .

Proof: $P_i(v)$ represents all the pixels having value v for the band i . If $v_1 \neq v_2$, no pixel can have the values of both v_1 and v_2 for the same band. Therefore, if there is a 1

in $P_i(v_1)$ for any pixel, there must be 0 in $P_i(v_2)$ for that pixel and vice versa. Hence $rc\{P_i(v_1) \& P_i(v_2)\} = 0$.

Lemma 4: $rc(P_1 | P_2) = rc(P_1) + rc(P_2) - rc(P_1 \& P_2)$.

Proof: Let the number of pixels for which there are 1s in P_1 and 0s in P_2 is n_1 , the number of pixels for which there are 0s in P_1 and 1s in P_2 is n_2 and the number of pixels for which there are 1s in both P_1 and P_2 is n_3 .

Now, $rc(P_1) = n_1 + n_3$, $rc(P_2) = n_2 + n_3$, $rc(P_1 \& P_2) = n_3$

and $rc(P_1 | P_2) = n_1 + n_2 + n_3 = (n_1 + n_3) + (n_2 + n_3) - n_3$

$= rc(P_1) + rc(P_2) - rc(P_1 \& P_2)$

Theorem: $rc\{P_i(v_1) | P_i(v_2)\} = rc\{P_i(v_1)\} + rc\{P_i(v_2)\}$, where $v_1 \neq v_2$.

Proof: $rc\{P_i(v_1) | P_i(v_2)\} = rc\{P_i(v_1)\} + rc\{P_i(v_2)\} - rc\{P_i(v_1) \& P_i(v_2)\}$ (Lemma 4)

If $v_1 \neq v_2$, $rc\{P_i(v_1) \& P_i(v_2)\} = 0$. (Lemma 3)

Therefore, $rc\{P_i(v_1) | P_i(v_2)\} = rc\{P_i(v_1)\} + rc\{P_i(v_2)\}$.

4 DATA MINING TECHNIQUES USING P-TREES

The P-tree technology has been extended to work with a large number of data mining techniques. These include the following.

4.1 P-tree-based DTI Classifiers

This technique was used on large quantities of spatial data collected in various application areas, including remote sensing, geographical information systems (GIS), astronomy, computer cartography, environmental assessment and planning, etc. These data collections effectively arrive as streams of data since new data is constantly being collected. The problem with previous classifiers was that this presented a serious problem. Using P-tree technology, fast calculation of measurements, such as information gain, was achieved. The P-tree based decision tree induction classification and a classical decision tree induction method was experimental shown to be significantly faster than existing classification methods, making well suited for mining on streams and multimedia. [28]

4.2 Bayesian Classifiers

A Bayesian classifier is a statistical classifier, which uses Bayes' theorem to predict class membership as a

conditional probability that a given data sample falls into a particular class. The complexity of computing the conditional probability values can become prohibitive for most of the multimedia applications with a large attribute space. Bayesian Belief Networks relax many constraints and uses the information about the domain to build a conditional probability table. Naïve Bayesian Classification is a lazy classifier. Computational cost is reduced with the use of the Naïve assumption of class conditional independence, to calculate the conditional probabilities when required. Bayesian Belief Networks require build time and domain knowledge where as the Naïve approach loses accuracy if the assumption is not valid. The P-tree data structure allows us to compute the Bayesian probability values efficiently, without the Naïve assumption by building P-trees for the training data. Calculation of probability values require a set of P-tree AND operations that will yield the respective counts for a given pattern. Bayesian classification with P-trees has been used successfully on remotely sensed image data to predict yield in precision agriculture [30].

4.3 ARM

Association Rule Mining, originally proposed for market basket data, has potential applications in many areas. Extracting interesting patterns and rules from datasets composed of images and associated data can be of importance. However, in most cases the data sizes are too large to be mined in a reasonable amount of time using existing algorithms. Experimental results showed that using P-tree techniques in an efficient association rule mining algorithm P-ARM has significant improvement compared with FP-growth and Apriori algorithms. [28]

4.4 KNN and Closed KNN Classifiers

KNN classifiers typically have a very high cost associated with building a new classifier each time new data arrives. In this situation, k-nearest neighbor (KNN) classification is a very good choice, since no residual classifier needs to be built ahead of time. KNN is extremely simple to implement and lends itself to a wide variety of variations. The construction of the neighborhood is the high cost operation. By using P-tree technology and finding a closed-KNN set which does not have to be reconstructed. Experimental results show closed-KNN yields higher classification accuracy as well as significantly higher speed. [31]

4.5 P-tree Data Mining Performance

Based on the experimental work discussed above incorporation of P-tree technology into data mining applications has consistently improved performance. The data mining ready structure has demonstrated its potential for improving performance in multimedia data.

Many types of data show continuity in dimensions that are not themselves used as data mining attributes. Spatial data that is mined independently of location will consist of large areas of similar attribute values. Data streams and many types of multimedia data, such as videos show a similar continuity in their temporal dimension. The P-tree data structure uses these continuities to compress data efficiently while allowing it to be used in computations. Individual bits of the mining-relevant attributes are represented in separate P-trees. Counts of attribute values or attribute ranges can efficiently be calculated by an "AND" operation that all relevant P-trees. These "AND"-operations can be efficiently implemented based on the regular structure that compresses entire quadrants, while making use of pre-computed counts that are kept at intermediate levels of the tree structure.

5 IMPLEMENTATION ISSUES AND PERFORMANCE

The performance of the P-tree data structure is discussed with respect to P-tree storage and the execution time for AND operations. The amount of internal memory required for each P-tree structure is related to the respective size of the P-tree file stored in secondary storage. The creation and storing of P-trees is a one-time process. To make a generalized P-tree structure, the following file structure is proposed (table 2) for storing basic P-trees. .

1 byte	2 bytes	1 byte	4 bytes	2 bytes	
Format Code	Fan-out	# of levels	Root count	Length of the body	Body of the P-tree

Table 2 P-tree file structure

Format code: Format code identifies the format of the P-tree, whether it is a *PCT* or *PMT* or in any other format.

Fan-out: This field contains the fan-out information of the P-tree. Fan-out information is required to traverse the P-tree in performing various P-tree operations. The fan-out is decided at creation time. In the case of using different fan-outs at different levels, it will be used as an identifier.

of levels: Number of levels in the P-tree. This will indicate the number of levels in the P-tree for the given fan-out.

Root count: *Root count* i.e. the number of 1s in the P-tree. Though we can calculate the *root count* of a P-tree on the fly from the P-tree data, these 4 bytes of space can save computation time when we only need the root count of a P-tree to take advantage of the properties described in section 2.5. The root count of a P-tree can be computed at the time of construction with very little extra cost.

Length of the body: Length of the body is the size of the P-tree file in bytes excluding the header. The size of the

P-tree varies due to the level of compression in the data. To allocate memory dynamically for the P-trees, it is better to know the size of the required memory size before reading the data from disk. This will also be an indicator of the distribution of the data, which can be used to estimate the required AND time in advance for the given search space.

Body of the P-tree : This will contain a long stream of bytes representing the P-tree in the respective format.

We only store the basic P-trees for each dataset. All other P-trees (value P-trees and tuple P-trees) are created on the fly when required. This results in a considerable saving of space. Figure 10, 11 and 11 gives the storage requirements for various formats of data (TIFF, SPOT and TM scene) using various formats of P-trees (PCT or PMT) with different fan-out patterns. Fan-out pattern f1-f2-f3 will indicate a fan-out of f1 for the root level, f3 for the leaf level and f2 for all the other levels. The variation in the size is due to the different levels of compression for each bit in the image. It is important to note that P-tree is a lossless representation of the original data. Different representations have an effect on the computation of the Ptree operators. The performance of the processor against memory access should be taken into consideration when selecting a representation.

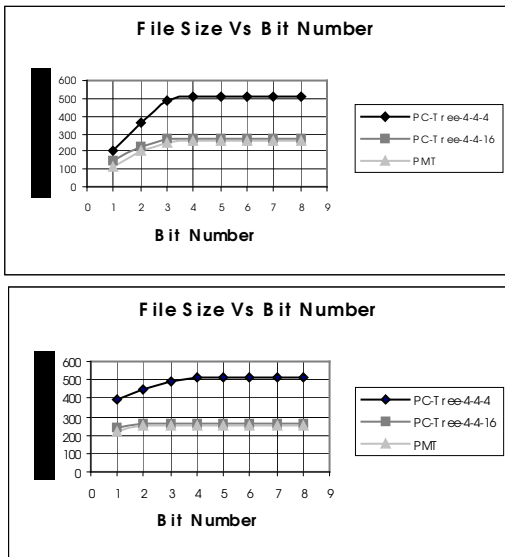


Figure 10 Comparison of file size for different bits of Band 1 & 2 of a TIFF image

The efficiency of data mining with the P-tree data structure relies on the time required for basic P-tree operators. The AND operation on 8 basic P-trees can be done in 12 milliseconds for an image file with 2 million pixels. Experimental results also show that the AND operation is scalable with respect to data size and the number of attribute bits. Figure 13 and 14 show the time required to perform the P-tree AND operation.

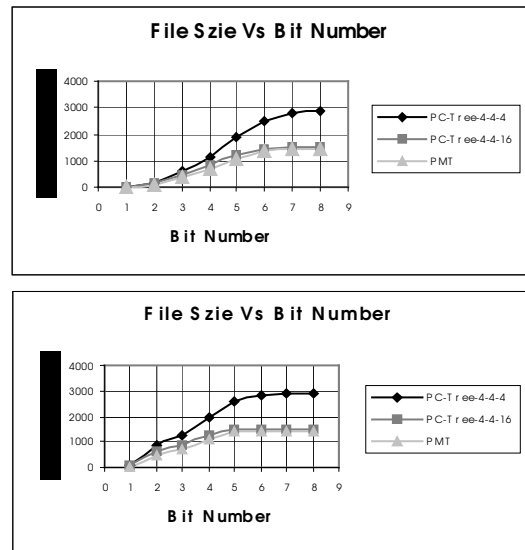


Figure 11 Comparison of file size for different bits of Band 3 & 4 of a SPOT image

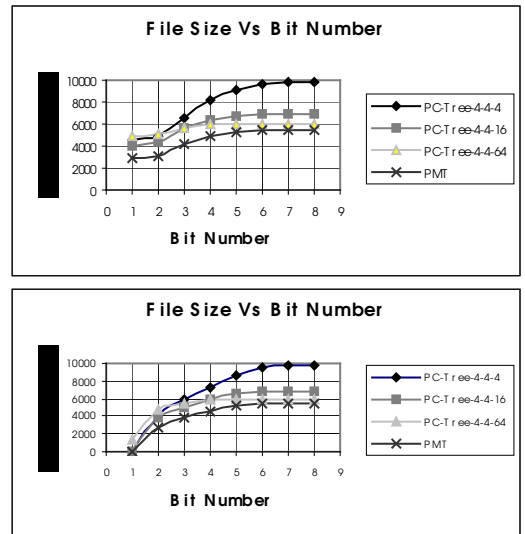


Figure 12 Comparison of file size for different bits of Band 5 & 6 of a TM image

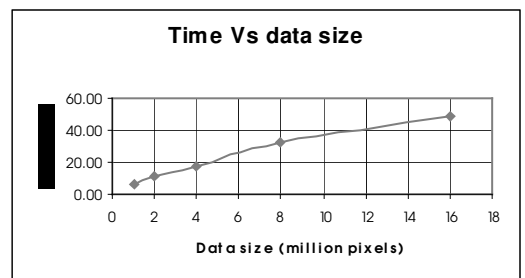


Figure 13 Comparison of time required to perform AND operation with different data sizes

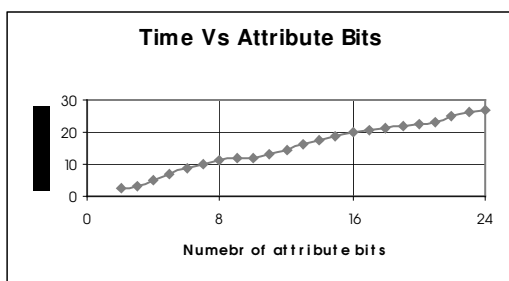


Figure 14 Time to perform AND operation for different number of attribute bits

The P-tree data structure provides an opportunity to use high performance parallel and distributed computing, independent of the data mining technique. The most common approach is to use a quadrant based partition, i.e. a horizontal partition. In this approach the AND operations on each partition can be accumulated to produce the global count. A vertical partition can also be used with a slight increase in communication cost. In this approach the AND operation on partially created value P-trees from each partition will produce the global count. Both these approaches can be used to mine distributed multi media data by converting the data into P-trees and storing it at the data source. The particular data mining algorithm will be able to pull the required counts through a high speed dedicated network or the Internet. If the latency delay is considerably high this approach may put a restriction on the type of algorithms to suit batched count requests from the P-trees.

6 RELATED WORK

Concepts related to the P-tree data structure, include Quadtrees [1, 2, 3, 4, 5] and its variants (such as point quadtrees [3] and region quadtrees [4]), and HH-codes [6].

Quadtrees decompose the universe by means of iso-oriented hyperplanes. These partitions do not have to be of equal size, although that is often the case. The decomposition into subspaces is usually continued until the number of objects in each partition is below a given threshold. Quadtrees have many variants, such as point quadtrees and region quadtrees.

HH-codes, or Helical Hyperspatial Codes, are binary representations of the Riemannian diagonal. The binary division of the diagonal forms the node point from which eight sub-cubes are formed. Each sub-cube has its own diagonal, generating new sub-cubes. These cubes are formed by interlacing one-dimensional values encoded as HH bit codes. When sorted, they cluster in groups along the diagonal. The clusters are order in a helical pattern, thus the name "Helical Hyperspatial".

The similarities among P-tree, quadtree and HHCode are that they are quadrant based. The difference is that P-trees focus on the count. P-trees are not index, rather they

are representations of the datasets themselves. P-trees are particularly useful for data mining because they contain the aggregate information needed for data mining.

7 CONCLUSION

This paper reviewed some of the issues of multimedia data mining and concludes that one of the major issues of multimedia data mining is the sheer size of the resulting feature space extracted from the raw data. Deciding how to efficiently store and process this high volume, high dimensional data will play a major role in the success of a multimedia data mining project. This paper proposes the use of a data mining ready data structure to solve the problem. To that end the Peano Count Tree (or P-tree), and its algebra and properties were presented. The P-tree structure can be viewed as a data-mining-ready structure that facilitates efficient data mining [7]. Previous work has demonstrated that using the P-tree algebra can perform standard data mining techniques efficiently while operating directly from a compress data storage.

8 REFERENCES

- [1] Volker Gaede and Oliver Gunther, "Multidimensional Access Methods", *Computing Surveys*, 30(2), 1998.
- [2] H. Samet, "The quadtree and related hierarchical data structure". *ACM Computing Survey*, 16, 2, 1984.
- [3] H. Samet, "Applications of Spatial Data Structures", Addison-Wesley, Reading, Mass., 1990.
- [4] H. Samet, "The Design and Analysis of Spatial Data Structures", Addison-Wesley, Reading, Mass., 1990.
- [5] R. A. Finkel and J. L. Bentley, "Quad trees: A data structure for retrieval of composite keys", *Acta Informatica*, 4, 1, 1974.
- [6] HH-codes. Available at <http://www.statkart.no/nlhdb/iveher/hhtext.html>
- [7] William Perrizo, Qin Ding, Qiang Ding and Amalendu Roy, "Deriving High Confidence Rules from Spatial Data using Peano Count Trees", Springer-Verlag, LNCS 2118, July 2001
- [8] Jochen Doerre, Peter Gerstl, Roland Seiffert "Text Mining: Finding Nuggets in Mountains of Textural Data"
- [9] Dan Sullivan "The Need for Text Mining in Business Intelligence"
- [10] Osmar R.Zaiane, Jiawei Han, Ze-Nian Li, Sonny H.Chee, Jenny Y.Chiang, "MultiMediaMiner: A System Prototype for MultiMedia Data mining", In pro.1998 ACM-SIGMOD Conf.on Management of Data, June 1998
- [11] Wei-Hao Lin, Rong Jin, Alexander Hauptmann, "Meta-classification of Multimedia Classifiers", First International Workshop on Knowledge Discovery in Multimedia and Complex Data

- [12] P. Indyk, R. Motwani, P. Raghavan "locality-preserving hashing in multidimensional spaces",
- [13] U. Fayyad, G. Piatesky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communication of ACM*, 39(11):27-34, November 1996.
- [14] Wei-hao lin, Rong Jin, Alexander Hauptmann, Meta-classification of Multimedia classifiers, First international workshop on knowledge discovery in multimedia and complex data, Taipei, Taiwan, May 6, 2002
- [15] William Baker, Arthur Evans, Lisa Jordan, Saurabh Pethe, "User Verification System" The Mid-Atlantic Student Workshop on Programming Languages and Systems Pace University, April 19, 2002
- [16] C. Aggarwal, "Re-designing Distance Functions and Distance-Based Applications for High Dimensional Data", SIGMOD 2001.
- [17] M. Gavrilov, D. Anguelov, P. Indyk, R. Motwani, "Mining The Stock Market: Which Measure Is Best?", KDD 2000
- [18] J. Caraca-Valente, I. Lopez-Chavarrias, "Discovering Similar Patterns in Time Series", KDD 2000
- [19] J. Yoon, T. Kim, and H. Lee, "The Information of Trading Volume in the Prediction of Stock Index returns: A Nonparametric Investigation", INFORMS & KORMS, 2000.
- [20] A. Hinneburg, C. Aggarwal, and D. Keim, "What Is the Nearest Neighbor in High Dimensional Spaces?", Proc. of the 26th VLDB Conference 2000.
- [21] C. Aggarwal, A. Hinneburg, and D. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space", ICDT 2001.
- [22] Chabane Djeraba, "Image Access and Data Mining: An Approach", PKDD 2000.
- [23] Chabane Djeraba, Henri Briand, "Temporal and Interactive Relations in a Multimedia Database System", ECMAST 1997.
- [24] Osmar R. Zaïane, Simeon J. Simoff, "Multimedia Data Mining for the Second Time", SIGKDD Explorations, Vol 3, N 2, January 2002.
- [25] Osmar R. Zaïane, Jiawei Han, Hua Zhu, "Mining Recurrent Items in Multimedia with Progressive Resolution Refinement", ICDE 2000.
- [26] Simeon J. Simoff, Osmar R. Zaïane, "Multimedia data mining", KDD 2000.
- [27] Osmar R. Zaïane, Jiawei Han, Ze-Nian Li, Jean Hou, "Mining Multimedia Data", *CASCON'98: Meeting of Minds*, 1998.
- [28] "Decision Tree Classification of Spatial Data Streams Using Peano Count Trees", Qiang Ding, Qin Ding and William Perrizo, Proceedings of ACM Symposium on Applied Computing (SAC'02), Madrid, Spain, March 2002, pp. 413-417.
- [29] "Association Rule Mining on Remotely Sensed Images Using P-trees", Qin Ding, Qiang Ding and William Perrizo, Proceedings of PAKDD 2002, Springer-Verlag, LNAI 2336, May 2002, pp. 66-79.
- [30] Mohamed Hossain, 'Bayesian Classification using P-Tree', Master of Science Thesis, North Dakota State University, December 2001.
- [31] "K-nearest Neighbor Classification on Spatial Data Stream Using P-trees", Maleq Khan, Qin Ding and William Perrizo, Proceedings of PAKDD 2002, Springer-Verlag, LNAI 2336, May 2002, pp. 517-528.
- [32] "Biological Systems and Data Mining for Phylogenomic Expression Profiling " Willy Valdivia-Granda*, Edward Deckard, William Perrizo, Qin Ding, Maleq Khan, Qiang Ding, Anne Denton