Feature-Based Adaptive Tolerance Tree (FATT): An Efficient Indexing Technique for Content-Based Image Retrieval Using Wavelet Transform

Dr. P. Anandha Kumar  
Department of Information Technology  
Madras Institute of Technology Anna University Chennai, Chennai, India  
Email ID: annauniv@edu

V.Balamurugan  
Research Scholar, Department of Information Technology  
Madras Institute of Technology Anna University, Chennai, India  
Email ID: vbalamurugan19@gmail.com

Abstract—This paper introduces a novel indexing and access method, called Feature-Based Adaptive Tolerance Tree (FATT), using wavelet transform is proposed to organize large image data sets efficiently and to support popular image access mechanisms like Content Based Image Retrieval (CBIR). Conventional database systems are designed for managing textual and numerical data and retrieving such data is often based on simple comparisons of text or numerical values. However, this method is no longer adequate for images, since the digital presentation of images does not convey the reality of images. Retrieval of images becomes difficult when the database is very large. This paper addresses such problems and presents a novel indexing technique, Feature Based Adaptive Tolerance Tree (FATT), which is designed to bring an effective solution especially for indexing large databases. The proposed indexing scheme is then used along with a query by image content, in order to achieve the ultimate goal from the user point of view that is retrieval of all relevant images. FATT indexing technique, features of the image is extracted using 2-dimensional discrete wavelet transform (2D-DWT) and index code is generated from the determinant value of the features. Multiresolution analysis technique using 2D-DWT can decompose the image into components at different scales, so that the coarsest scale components carry the global approximation information while the finer scale components contain the detailed information. Experimental results show that the FATT outperforms M-tree up to 200%, Slim-tree up to 120% and HCT upto 89%. FATT indexing technique is adopted to increase the efficiency of data storage and retrieval.

Index Terms— CBIR, FATT, indexing, wavelet transform.

INTRODUCTION

One of the challenges in the development of a content based indexing and retrieval application is to achieve an efficient indexing scheme. Efficient image indexing and accessing tools are of paramount importance in order to fully utilize the increasing amount of digital data available on the internet and digital libraries. Recent years have been a growing interest in developing effective methods for indexing large image databases. Image databases are utilized in diverse areas such as advertising, medicine, entertainment, crime detection, education, military and in the domain of museum and gallery image collections etc. This motivates the development of efficient image indexing and retrieval systems and algorithms. However as the database grow larger, the traditional keywords based method to retrieve a particular image becomes inefficient[21],[26],[33],[35].

Content- based image retrieval (CBIR) approach has emerged as a promising alternative. CBIR has been a particular challenge as the image content covers a vast range of subjects and requirements from end users are often very loosely defined. In CBIR, images are indexed by its own visual contents, such as color, texture and shape. The challenge in CBIR is to develop the methods that will increase the retrieval accuracy and reduce the retrieval time. Hence, the need of an efficient indexing structure for images supporting popular access mechanisms like CBIR arises,[26], [33], [35].

In order to overcome such problems several content based indexing and retrieval applications have been developed such as the MUVIS system [22],[23],[29], photobook [31], VisualSEEK[37], Virage[40], and VideoQ[9] all of which are designed to bring a framework structure for handling and especially retrieval of the digital multimedia items, such as images, audio, and/or video clips. In this way similarity between two database images can be retrieved by calculating the similarity distance between their feature vectors. This is the general query by example (QBE) scenario, which on the other hand is expensive and CPU intensive especially for large databases. The main challenge of implementing such an index structure is to make it capable of handling high level image relationships easily and efficiently during access[7], [14],[21].

The indexing techniques can be mainly grouped in two types (1) Spatial Access Methods (SAMs) and (2) Metric Access Methods (MAMs) are proposed[21]. Initial attempts of SAMs are such as KD-trees[1], R-trees[16], SS-tree[41], Hybrid tree[8]etc. Especially for content-based indexing and retrieval in large scale databases. SAMs have several drawbacks and limitations. SAM-based indexing technique partitions and works over a single feature space.SAMs while providing good results on low dimensional feature space and do not scale up well to high dimensional spaces. SAM-based indexing schemes even becomes less efficient than sequential indexing for high dimensions.(2)Static and Dynamic Access Methods(MAMs)[17] are proposed such as VP-tree[43], MVP-tree[46], GINAT[6] whereas the dynamic ones, M-tree[7] and
other M-tree variants such as M+-tree. The existing multidimensional index structures support CBIR by translating the content-similarity measurement into features level equivalence, which is a very difficult job and can result in erroneous interpretation of user’s perception of similarity.

The indexing structures so far addressed are all designed to speed up any Query By Example (QBE) process by some multidimensional index structure. However all of them have significant drawbacks and shortcomings for indexing of large scale databases [1], [6], [7], [21], [41]. In order to overcome such problems and provide efficient solutions to the aforementioned shortcomings of the indexing algorithm especially for image databases, we develop a MAM-based balanced and dynamic indexing technique called Feature-Based Adaptive tolerance Tree (FATT).

In this paper, we present FATT consists of root node (grandparent) is having the maximum equal parent nodes of 256 and depth of the tree depends upon the number of features considered and width of the tree depends upon the individual index value.

The rest of the paper is organized as follows: Section II presents the related work in the area of the indexing and retrieval. In Section III we introduce the general structure of FATT and algorithms. Section IV for wavelet transform and image coding. Section V presents the experimental results. Finally, Section VI concludes this paper.

RELATED WORK

There are several multidimensional indexing techniques for capturing the low-level features like feature based or distance based techniques, each of which can be further classified as data-partitioned [3], [7], [16], [44] for space partitioned based algorithm [27], [34]. Feature based indexing techniques project an image as a feature vector in a feature space and index the space. The basic feature based index structures are KD-tree [1], R-tree [16] etc. Researchers proposed several indexing techniques that are formed mostly in a hierarchical tree structure that is used to cluster the feature space. Initial attempts of such as KD-trees [1] used space-partitioning methods that divide the feature space into predefined hyperplanes regardless of the distribution of the feature vectors. Such regions are mutually disjoint and their union covers entire space. In R-tree [16] the feature space is divided according to the distribution of the database items and region overlapping may occur as a result. Both KD-tree and R-tree are the first examples of SAMs. Afterwards several enhanced SAMs have been proposed. R*-tree [2] provides consistently better performance by introducing a policy called “forced reinsert” than the R-tree and R+-tree [36]. R*-tree also improves the node splitting policy of the R-tree by taking overlapping area and region parameters into consideration. Lin et al proposed TV-tree [19], which called as telescope vectors. These vectors can be dynamically shortened assuming that only dimensions with high variance dimensions can be neglected. Berchtold et al [3] introduced X-tree, which is particularly designed for indexing high dimensional data. X-tree avoids overlapping of region bounding boxes in the directory structure by using a new organization of the directory as a result, X-tree outperforms both TV-tree and R*-tree significantly. It is 450 times faster than R-tree and between 12 times faster than the TV-tree when the dimension is higher than two and it also provides faster insertion times. Still, bounding rectangles can overlap in higher dimensions. In order to prevent this, White and Jain proposed the SS-tree [41], an alternative to the R-tree structure, which uses minimum bounding spheres instead of rectangles. Even though SS-tree outperforms R*-tree, the overlapping in the high dimensions still occur. Thereafter, several other SAM variants are proposed such as SR-tree [20], S*-tree [42], Hybrid-tree [8], A-tree [38], IQ-tree [5], Pyramid tree [4], NB-tree [13] etc. The aforementioned degradations and shortcomings prevent a wide usage of SAM based indexing structures especially for large scale image collections. In order to provide a more general approach to similarity indexing for several MAM- based indexing techniques have been proposed. Yianilos [43] presented VP-tree that is based on partitioning the feature vectors into two groups according to their similarity distances with respect to a reference point, a so-called vantage point. Bozkaya and Ozsoyoglu [47] proposed extension of the VP-tree, a so-called MVP-tree (multiple vantage point), which basically assigns m vantage points to a node with a fan out of m². They reported 20%-80% reduction of similarity distance computation comparing to VP-trees. Brin [6] introduced the geometric near-neighbor access tree (GNAT) indexing structure, which chooses k split points at the top level and each of the remaining feature vectors are associated with the closest split points. GNAT is then built recursively and the parameter k is chosen to be a different value for each feature set depending on its cardinality. Koikkalainen and Oja introduced TS-SOM [25] that is used in PicSOM [28] as a CBIR indexing structure. TS-SOM provides a tree-structure vector quantization algorithm. Other SOM-based approaches are introduced by Zhang and Zhong [45] and Sethi and Coman [39]. All SOM-based indexing method rely on training of the levels has a pre-fixed node size that has to arranged according to the size of the database. This brings a significant limitation, that is they are all static indexing structures, which do not allow dynamic construction or updates for particular database. Retraining and costly reorganizations are required each time the content of the database changes (i.e., new insertions and deletions), that is indeed nothing but rebuilding the whole indexing structure from the scratch. Similarly, the rest of the MAMs, so far addressed present several shortcomings. Contrary to SAMs, these metric trees are designed only to reduce the number of similarity distance computations, paying no attention to I/O costs (disk page accesses). They are also intrinsically static methods in the sense that the tree structure is built once and new insertions are not supported. Furthermore, all of them build the indexing structure from top to bottom and hence the resulting tree is not guaranteed to be balanced. Ciaccia et al [7] proposed the M-tree to overcome such problems. The M-tree is a balanced and dynamic tree, which is built from bottom to top, creating a new level only when necessary. The node size is a fixed number M, and therefore the tree height depends on M and the...
database size. Its performance optimization concerns both CPU computational time for similarity distances and I/O costs for disk accesses for feature vectors of the database items. Traina et al. [15] proposed Slim-tree, an enhanced variant of M-trees, which is designed for improving the performance by minimizing the overlaps between nodes. They introduced two parameters, “fat-factor" and “blot factor”, to measure the degree of overlap and proposed the usage of minimum spanning tree (MST)[24],[32],for splitting the node. Another slightly enhanced M-tree structure, a so-called M+-tree, can be found.[46].Serkan kiranyaz and Moncef Gabboj [21] proposed Hierarchical cellular tree(HCT) has no limit for cell size as long as cell keeps a definite compactness.Any action (i.e,an item insertion)it consequent reactions cannot last indefinitely due to the fact that each of them can occur only in a higher level and any HCT body has naturally limited number of levels. Ming Zhang and John Fulcher[48] proposed GAT-trees which is neural network based adaptive tree but it takes longer time to train the network and it gives accuracy 5% higher than general tree random noise is added, 7% accuracy when gamma value of Gaussian noise exceeds 0.3.

1. FATT indexing is MAM-based and have a hierarchical structure, i.e., levels (m).
2. FATT designed, dynamically and balanced in a top-to-bottom fashion to overcome the problem of overlap between nodes in metric spaces.
3. Root node (grant parent(A)) grows from the root node to leaves, each parent node is represented with several left child(lchild) and rightchild(rchild) nodes and maximum of 256(N) nodes.
4. Each child node immediately proceeding with equal left child(lchild) and rightchild(rchild). For both lchild and rchild of A+1 is at NA+N=N(N+1).Unless N(A+1)>m in which case (A+1) has no lchild.Hence this representation can clearly be used for all image indexing and fast searching process.

FATT STRUCTURE AND ALGORITHMS

FATT Structure
FATT is balanced, adaptive and an indexing tree .It is designed in a complete N-ary, where N is the number of nodes. The FATT is constructed based on the index code developed. The height of the tree depends upon the number of features considered and width of the tree depends upon the number individual index codes that are used in code generation. The leaf node contains the index value. The retrieval performance depends upon the tree construction. The size of the tree can be changed depending upon the number of images that has to be stored. The structure of the tree can be changed depending upon images found. The general structure (4 levels, m=4, N=256)of the FATT is shown in Fig.1 The maximum number unique index value for a m-levels, N-nodes tree is given as 

$$N^m$$

| No. of nodes at top level (m)=1 |
|---------------------------------|
| No. of nodes for each node at first level (m)=256 |
| No. of nodes for each node at second level (m)=256 |
| No. of nodes for each node at third level (m)=256 |
| No. of nodes for each node at fourth level (m)=256 |
| No. of individual index value= (1×256×256×256×256)=4^{10} |

Where $N=256, m=4$

FATT Algorithms and Operations
There are 3 FATT algorithms and operations (i.e) image insertion, searching and retrieval (indexing).

The insertion algorithm Insert (nextImage,levelNo) first performs the searching algorithm which hierarchically FATT from root node(grant parent) to the least child node .In order to locate most suitable image for nextimage.Once the least node is located, the image is inserted into the immediate parent node. Let the nextImage be the image to be inserted into a target level indicated by levelNo. Accordingly the insertion algorithm as follows and the sample insertion of 2 nodes ,4 levels,(N=2, m=4) 3 nodes 4 levels(N=3, m=4) is shown in Fig.2 and Fig.3

1. Insertion Algorithm
Insert(nextImage,levelNo)
Let the root level number, rootLevelNo and the single image in the top level:

- If(levelNo=rootLevelNo) then do;
  - Append nextImage into least lchild node
  - Assign rootLevelNo=levelNo
- If(levelNo<root levelNo) then do;
  - Append nextImage into least rchild node.

Fig.1. General structure of the FATT($N=256, m=4$)
Check least child node for post-processing
- If least child node is split then do;
- Create the least lchild node equal to immediate parent node
- Insert(nextImage, immediate root levelNo)
  Else if insertion of least child node then do;
  Let least lchild and nextImage be parent and new child node
  Else if unchanged the immediate parent node
  Return.

A. Insertion of 4-levels and 2-nodes (m=4, N=2)

Number of levels, m = 4
Number of nodes, N = 2

Then the total number of index values \(N^m\) is 16

If a complete FATT with \(N=2, m=4\) depth first search is represented sequentially as above then for then for grand parent node with index \(A, 1 \leq A \leq m\) we have,

Case (1) grandparent \((A)\) is at \((A/3)\) if \(A \neq 1\). When \(A=1, A\) is the root and has no parent.
Case (2) leftchild \((A)\) is at \(2A\) if \(2A \leq N\). If \(2A > m\), then grand parent \(A\) has no left child(lchild)
Case (3) middlechild \((A)\) is at \(2A+1\) if \(2A+1 \leq m\). If \(2A+1 > m\), then \(A\) has no right child(rchild)

Proof:
We prove case(2) and case(3) is immediate consequence of case(2) and the numbering of nodes on the same level left to middle and right. Case(1) follows from case (2) and (3). We prove case (2) by induction on \(A\). For \(A=1\), clearly lchild is at 3 unless \(3 > m\), then nodes immediately proceeding lchild(A+1), in the representations are the middle child of \(A\) and the lchild(A). The lchild(A) is at \(3A\), hence the lchild of \(3A+2\) unless \(3(A+1) > m\).

In general if a complete FATT with number of maximum nodes is \(256(N=256)\) (ie, depth = \(N^m\)) is represented sequentially as above then for grandparent node with index \(A, 1 \leq A \leq N\) we have,

Case (1) grandparent \((A)\) is at \((A/N)\) if \(A \neq 1\). When \(A=1, A\) is the root and has no parent.
Case (2) leftchild \((A)\) is at \(2A\) if \(2A \leq 256\). If \(2A > 256\) then \(A\) has no lchild, mchild(A) at \(2A+1\) if \(2A+1 \leq 256\). If \(2A+1 > 256\) then \(A\) has no mchild.
Case (3) rightchild \((A)\) is at \(2A+2\) if \(2A+2 \leq 256\).

If \(2A+2 > 256\) then \(A\) has no rchild. We prove that case(2), (3) and (4) is an immediate consequence of case (2) and the numbering of nodes on the same level from left to right case(1) follows from case (2), (3) and (4). We prove case (2) by induction on \(A\) at 2. Unless \(2 > 256\) in which case (1) has no lchild.

Then, the two nodes immediately proceeding lchild \((A+1)\) in the representations are the mchild A and the lchild of A. The lchild of A is at \(2A\). Hence the
lchild of A+1 is at 2A+2=2(A+1) unless 2(A+1)>256 in which case A+1 has no lchild.

2. Searching Algorithm

Depth first search algorithm is used for traversing the tree. In this algorithm search as deeply as possible by visiting a node, and then recursively performing depth -first search on each adjacent node.

Search(nextImage,levelNo)

- If the parentnode > root node then do;
  - Go to the lchild of the root node
  - Until no lchild is available
  - Check whether the indexing value is obtained
- Elseif parentnode< root node
  - Return the index value
- Else not found traverse step 1 and 2
Repeat the steps 1,2 and 3 until index value is found.

The searching algorithm search(nextImage,levelNo) first perform the searching with root node.
Case (1) If parentnode > root node searching beginning with lchild node until no leftmost child is available.
Case (2) If parentnode< root node then return the index value otherwise search for leftmost child node is available

The indexing algorithm index(nextImage,levelNo) first perform the indexing with root node
Case (1) If parentnode > root node indexing beginning with lchild node until no leftmost child is available.
Case (2) If parentnode< root node then return the index value otherwise index for leftmost child node is available

3. Retrieval Algorithm

FATT can index large image-scale database using index code generated and euclidean similarity measure used as distance metric

Index(nextImage,levelNo)

- If the parentnode > root node then do;
  - Go to the lchild of the root node
  - Until no lchild is available
  - Check whether the indexing value is obtained
- Elseif parentnode< root node
  - Return the index value
- Else not found traverse step 1 and 2
  - Repeat the steps 1,2 and 3 until index value is found.

By wavelet transform, we mean the decomposition of an image with family of real orthogonal bases ψm,n obtained through translation and dilation of a kernel function ψ known as mother wavelet.[11].

\[ \psi_{m,n}(x) = 2^{-m} \psi(2^{-m} x - a) \]  (1)

Where \( m \) and \( n \) are integers. Due to the orthonormal property, the wavelet coefficients of a signal \( f \) can be easily computed via

\[ c_{m,n} = \int_{-\infty}^{\infty} f(x) \psi_{m,n}(x) \]  (2)

and the synthesis formula

\[ f(x) = \sum_{m,n} c_{m,n} \psi_{m,n}(x) \]  (3)

can be used to recover \( f \) from its wavelet coefficients.

To construct the mother wavelet \( \psi \) we may first determine a scaling function \( \phi \) which satisfies the two-scale difference equation

\[ \phi(x) = \sqrt{2} \sum_{k} h(k) \phi(2x - k) \]  (4)

Then the wavelet kernel \( \psi \) is related to the scaling function via,
\begin{align*}
\psi(x) &= \sqrt{2} \psi_{j-1}(x) \\
\end{align*}
(5)

where

\begin{align*}
g(k) &= (-1)^k h(1 - k) \\
\end{align*}
(6)

The coefficients \( g(k) \) in (4) have to several conditions for the set of basis waveform functions in (1) to be unique, orthonormal, and have a certain degree of regularity [11], [18]. The coefficients \( h(k) \) play a very crucial role in a given discrete wavelet transform. To perform the wavelet transform does not require the explicit forms of \( \psi(x) \) but only depends on \( h(k) \) and \( g(k) \). Consider a J-level wavelet decomposition which can be written as

\begin{align*}
f(x) &= \sum_k (\psi_{j-1,k} \phi_{j-1,k} (x) + \sum \phi_{j,k} \theta_{j,k} (x)) \\
\end{align*}
(7)

Where coefficients \( \psi_{j,k} \) are given and coefficients \( \phi_{j,k} \) and \( \theta_{j,k} \) at scale \( j \) are related to the coefficients \( \phi_{j-1,k} \) and \( \theta_{j-1,k} \) at scale \( j-1 \) via

\begin{align*}
\phi_{j+1,k} &= \sum \phi_{j,k} \\
\theta_{j+1,k} &= \sum \theta_{j,k} \\
\end{align*}
(8)

\begin{align*}
\phi_{j+1,k} &= \sum \phi_{j,k} \\
\theta_{j+1,k} &= \sum \theta_{j,k} \\
\end{align*}
(9)

Where \( 0 \leq j \). Thus, (8) and (9) provide a recursive algorithm for wavelet decomposition through \( \phi_{j,k} \) and \( \theta_{j,k} \). By using a similar approach, we can derive a recursive algorithm for function synthesis based on its wavelet coefficients \( \psi_{j,k} \) and \( \phi_{j,k} \).

\begin{align*}
\psi_{j,k} &= \sum \phi_{j+1,k} h(k - 2n) + \sum \phi_{j+1,k} g(k - 2n) \\
\end{align*}
(10)

It is convenient to view the decomposition (8) as passing a signal \( \psi(x) \) through a pair of filters and with impulse responses \( \phi_{j,k} \) and \( \theta_{j,k} \) and down sampling the filtered signals by two (dropping every other sample), where \( \phi_{j,k} \) and \( \theta_{j,k} \) are defined as

\begin{align*}
\phi_{j,k}(n) &= \phi(-n) \\
\theta_{j,k}(n) &= \phi(-n) \\
\end{align*}
(11)

The wavelet packet basis functions can be generated from a given function \( \psi(x) \) as follows

\begin{align*}
W_{l,n}(x) &= \sqrt{2} \sum h(k) \\
W_{l,n+1}(x) &= \sqrt{2} \sum g(k) \\
\end{align*}
(12)

Where the function \( \psi(x) \) can be identified with the scaling function and with the mother wavelet. Then, the wavelet packet bases can be defined to be the collection of orthonormal bases composed of functions of the form \( \psi_{l,k} \), where \( l, k, n \). Each element is determined by a subset of the indices: a scaling parameter \( l \), a localization parameter \( k \), and an oscillation parameter \( n \).

The 2D wavelet (or wavelet packet) basis functions can be expressed by tensor product of two 1-D wavelet (or wavelet packet) basis functions along the horizontal and vertical directions. The corresponding 2-D filter coefficients can be expressed as

\begin{align*}
h_{l,k}(l, l) &= \phi(k)h(l), \ h_{l,k}(l, k) &= g(k)g(l) \\
\end{align*}
(13)

Where the first and second subscripts in (13) and (14) denotes the low pass and high pass filtering characteristics in the - and - directions respectively. [11] We have applied orthogonal wavelet transformation with dyadic subsampling. Wavelet decomposition of images is performed using db4 wavelet basis function. On application of the the above procedure, for an image of size \( 256 \times 256 \), db4 wavelet yields subband matrices of \( 128 \times 128 \) at the first level, \( 64 \times 64 \) at the second level, \( 32 \times 32 \) at the third level, \( 16 \times 16 \) at the fourth level of wavelet decomposition.

\section*{Image Coding}

In image coding, a six digit code is generated for each image feature which is to be stored and retrieved. The code generated depends upon the value of the determinant generated. The number of different codes that can be generated depends upon the value of the determinant. The algorithm for finding the code is given below

\begin{align*}
\text{Code} = \text{function} (\text{determinant value}) \\
\end{align*}

The code is generated depends upon the determinant value of the matrix. The function generates the code between 0 to F based upon the value of the determinant. It is a function in which the input to the function is the determinant value obtained. Based upon the range of the determinant obtained the function returns value which is the code for the image. The database to store the images is a single table with three fields. The three fields are serial number, image and index value.

\section*{Image Coding Algorithm}

The above algorithm is applied to the image and the following code is generated for the image:

\begin{align*}
\text{Code} = \text{function} (\text{determinant value}) \\
\end{align*}

The code is generated depends upon the determinant value of the matrix. The function generates the code between 0 to F based upon the value of the determinant. It is a function in which the input to the function is the determinant value obtained. Based upon the range of the determinant obtained the function returns value which is the code for the image.
Step 1: Get the matrix of the individual feature \((m\times m)\)

Step 2: Find the transpose of the matrix obtained
- Read the \(m\times m\) matrix
- Give the input of the \(m\times m\) matrix elements
- Transpose the \(m\times m\) of the input matrix elements

Step 3: Multiply the two matrices to get a square matrix.
- Read the order of \(m\times m\) for the first matrix
- Read the order of \(p\times q\) for the second matrix
- Give the input of the \(m\times m\) of the first matrix elements
- Give the input of the \(p\times q\) of the second matrix elements
- For \(i=0; i<m; i++\), \(j=0; j<q; j++\), \(k=0; k<p; k++\)
- \(C[i][j]=C[i][j]+(A[i][k]\times B[k][j])\)

Step 4: Get the determinant value of the matrix \((m\times m)\)
- Read the \(m\times m\) matrix
- Give the input \(m\times m\) matrix elements
- The determinant value of the \(m\times m\) matrix
- Determinant value given by
  \[
  A[i][j] \times (A[i+1][j+1] \times A[i+2][j+2] - A[i+2][j+1] \times A[i+1][j+2]) - A[i+1][j+1] \times (A[i+1][j] \times A[i+2][j+1] - A[i+1][j+2]) + A[i+1][j+2] \times (A[i+1][j] \times A[i+2][j+1] - A[i+1][j+1] \times A[i+2][j])
  \]

Step 5: Based on the range of the determinant value the code generated.

\[
\begin{align*}
  &\text{If } x = 0 \text{ and } x \leq 10 \text{ then } y = 00 \\
  &\text{Else if } x > 10 \text{ and } x \leq 20 \text{ then } y = 01 \\
  &\text{Else if } x > 20 \text{ and } x \leq 30 \text{ then } y = 02 \\
  &\text{Else if } x > 30 \text{ and } x \leq 40 \text{ then } y = 03 \\
  &\text{Else if } x > 40 \text{ and } x \leq 50 \text{ then } y = 04 \\
  &\text{Else if } x > 50 \text{ and } x \leq 60 \text{ then } y = 05 \\
  &\text{Else if } x > 60 \text{ and } x \leq 70 \text{ then } y = 06 \\
  &\text{Else if } x > 70 \text{ and } x \leq 80 \text{ then } y = 07 \\
  &\text{Else if } x > 80 \text{ and } x \leq 90 \text{ then } y = 08 \\
  &\text{Else if } x > 90 \text{ and } x \leq 100 \text{ then } y = 09 \\
  &\text{Else if } x > 100 \text{ and } x \leq 110 \text{ then } y = 10 \\
  &\text{Else if } x > 110 \text{ and } x \leq 120 \text{ then } y = 11 \\
  &\text{Else if } x > 120 \text{ and } x \leq 130 \text{ then } y = 12 \\
  &\text{Else if } x > 130 \text{ and } x \leq 140 \text{ then } y = 13 \\
  &\text{Else if } x > 140 \text{ and } x \leq 150 \text{ then } y = 14 \\
  &\text{Else if } x > 150 \text{ then } y = 15 \\
  &\text{Endif}
\end{align*}
\]

D. Error Handling Phase

Errors may occur during retrieval of images from the database. The error is caused by the code change during the generation purpose. The code generated is made up of six parts each obtained from each individual feature. The error occurs when determinant value obtained returns a code that differs from the code of the image stored in the database. This changes the code entirely. The severity of the error lies in the level it occurs. When the error occurs at the bottom most level the severity is minimum. Whenever such error occurs searching the neighboring nodes can rectify it. This only possible only if the error occurs at the last level. If the error occurs at higher level tracing the tree becomes difficult.

Example
Let the image in the database have the node, 122300
Let the code generated for the image which is being given as the input be, 122301
Searching the adjacent nodes, which would return the image required, could rectify this error.

EXPERIMENTAL RESULTS

We have tested the FATT to index and retrieve the images with three different databases through a query process such as query by example (QBE). Database 1-D1 consists of 1000 natural scene images of size 256×256 of 60 selected categories are from D1:various beach images, sunsets, waterfalls, clouds, mountains and glaciers etc from COREL. Database 2-D2 consists of 10000 natural scene images of 100 categories similar contents with D1 from COREL. Database. Selected categories are rose, sunflower, dinosaur, bus, lion, elephant etc. We implemented in MATLAB 7.4 under Windows vista. The experiments were performed on a Pentium Core 2 Duo, 1.66GHz PC with 1GB RAM. Our implementations for node access, distance computations and retrieval process.

One indexing and retrieval results from D1 shown in Fig.6, D2 is shown in Fig.7 and for D3 shown in Fig.8. The effectiveness of the proposed tree have been validated by a large number of experiments.
Since FATT is balanced and dynamic tree the performances like I/Os and computation costs is compared with existing balanced and dynamic tree such as M-tree, Slim-tree, HCT. The FATT consistently outperforms irrespective of the database size .Fig.9, Fig.11, Fig.13 shows the I/Os versus number of objects in D1, D2 and D3. From the experimental though database size increased but the I/O cost maintains less and uniformly upto 83% for the D1, upto 81% for D2 and upto 82% for D3 the performance is achieved.

Fig.10, Fig.12, Fig.14 shows the computation costs versus no. of objects for D1 and D2. The computation cost the for D1 decreased up to 82% up to 81% for D2 and up to 83% for D3. Average distance computations performance also outperforms for D1 and D2. Experimental results show that the performance is no degradation irrespective of the database size is increased. FATT indexing technique can be conveniently used for retrieval and to achieve the users goal to retrieve the most relevant images.

**Complexity of the algorithm**

Suppose we are given $n$ images A, B, C...Z and suppose the images are inserted in order into a FATT node. There are $n$ permutations are required for $n$ images. Each such permutations will give rise to a corresponding tree. It can be shown that the average depth of the $n$ tree is approximately $\log_2 n + 1$. Accordingly, the average running time $f(n)$ to search
for image in FATT with n images is proportional to 
\[ \log_2 n, (i.e.) f(n) = O(\log^2 n + 1). \]

| Table III | COMPLEXITY ANALYSIS |
|-----------|----------------------|
| \[ \log n \] | \[ n \] | \[ n\log n \] | \[ n^2 \] | \[ n^3 \] |
| 1         | 2         | 2         | 4         | 8         |
| 1.584     | 3         | 4.754     | 9         | 27        |
| 2         | 4         | 8         | 16        | 64        |
| 2.321     | 5         | 11.609    | 25        | 125       |
| 2.584     | 6         | 15.509    | 36        | 216       |
| 7.807     | 7         | 19.651    | 49        | 343       |
| 3         | 8         | 24        | 64        | 512       |
| 3.169     | 9         | 28.529    | 81        | 729       |
| 3.321     | 10        | 33.329    | 100       | 100000    |
| 8         | 256       | 20.48     | 65536     | 16777216  |

FATT nodes represent the database images, this requires prior calculation of the relative similarity distances and hence yields a \( O(n^2) \) computational cost. The nodes are constructed dynamically since images can be at any time inserted since such operation requires \( O(n) \). Therefore, an incremental FATT algorithm is adaptive based on leaf nodes. This is a sequential algorithm and \( O(n) \) computational complexity per image and hence \( O(n^2) \) overall cost as desired.

**CONCLUSION AND FUTURE WORK**

In this paper, a novel indexing technique for content based image retrieval is designed particularly for fast insertion, searching and indexing, moreover to tackle the problem of overlap between the nodes. The M-tree has fixed cell policy due to its natural deficiency not suitable in dynamic environments and data are subject to permanent changes. Slim-tree is the extension of M-tree that speed up insertion using node splitting algorithm while also improving the storage utilization, unfortunately this algorithm does not guarantee the minimal occupation of each node. Whereas HCT has flexible cell policy but the levels are limited. Our proposed FATT greatly improves the performance comparing with M-tree, Slim-tree and HCT indexing schemes in I/Os costs, computation costs and average distance computations. The insertion speed of our FATT improves up to 84% compared with M-tree, up to 75% compared with Slim-tree, up to 70% compared with HCT. The searching speed of improves up to 146% compared with M-tree, up to 105% compared with Slim-tree, up to 89% compared with HCT. The possibility of using clustering, neural networks and other multimedia databases such as audio, video will also give attractive performances could be explored in future.

**REFERENCES**

[1] J.L.Bentley, “Multidimensional binary search trees used for associative searching,” In Proc.of communications of the ACM,vol.18,no.9,pp.509-517, Sept.1975.

[2] N.Beckmann,H-P.Kriegel,R.Schneider and B.Seeger, “The R*-tree:An efficient and robust access method for points and rectangles,” In Proc.of ACM SIGMOD Int.conf.on management of data,US,pp.322-331.1990.

[3] S.Berchtold,D.A.Keim,and H.Kriegel, “The x-tree:an index structure for high dimensional data,” In Proceedings of the 22nd International Conference on Very Large Databases,pp.28-39, India,Sept.1996.

[4] S.Berchtold,D.A.Keim and H.Kriegel, “The pyramid-technique:Towards breaking the curse of dimensionality,” In Proc.of 1998 ACM SIGMOD International Conference on Management of Data,pp.142-153, Seattle, Washington,US Jun. 01-04,1998.

[5] S.Berchtold,C.Bohm,H.V.Jagadish,H.P.Kriegel and Sander, “Independent Quantization:An index compression technique for high dimensional data spaces,” In Proc.of the 16th Int.Conf.on Data Engineering,US,pp.557-588, Feb.2000.

[6] S.Brin, “Near neighbor search in metric spaces,” In Proc.Int.Conf.on Very Large Databases(VLDB), pp.574-584, 1995

[7] P.Ciaccia, M.Patella, and P.Zezula, “M-tree:An efficient access method for similarity search in metric spaces,” In Proceedings of the 23th VLDB International Conference. Athens,Greece, pp.426-435,August 1997.

[8] K.Chakabarti and S.Mehrotra, “The hybrid tree:An index for high dimensional data spaces,” In Proceedings of the IEEE International Conference on Data Engineering, Australia pp.440-447,Mar. 1999.

[9] S.F.Chang,W.Chen,J.Meng,H.Sundaram and D.Zhong, “VideoQ::An automated content-based search system using cues,” In Proc.ACM Multimedia,Seattle,WA,1997.

[10] K.Chatterjee and S-C Chen, “Affinity hybrid tree:An indexing technique for content-based image retrieval in multimedia databases,” In Proc.of the Eighth IEEE International Symposium on Multimedia(ISM’06),pp.276-2755,2006.

[11] T.Chang and C-C Jay Kuo, “Texture analysis and classification with tree-structured wavelet transform,” IEEE Trans.on Image Processing,vol.2,no.4.Oct. 1993.

[12] R.Fagin “Fuzzy queries in multimedia database systems,In Proceedings of Sventeenth ACM SIGACT SIGMOD-SIGART symposium on principles of database systems,US, pp. 1-10, 1998.

[13] J.J.Fonseca,J-A.Jorge, “Indexing High –Dimensional Data for Content–Based Retrieval in Large Databases”, In Proc.of Eighth International Conference on Database Systems for Advanced Applications(DASFAA 03).Kyoto-Japan,Mar.26-28, pp.267-274,2003.

[14] C.Traina,A.J.M.Traina, B.Seeger, and C.Faloutsos, “Slim–trees :High performance metric trees minimizing overlap between nodes,” In Proc.Oedbt 2000, Konstanz,Germany, pp.51-65. Mar. 2000.

[15] C.Traina,A.J.M.Traina, C.Faloutsos, and B.Seeger, “Fast indexing and visualization of metric data sets using slim–trees,” IEEE Transactions on Knowledge and Data Engineering,vol.14.,no2. pp. 244-260. Mar/Apr.2002.

[16] A.Guttman. “R–trees: A dynamic index structure for spatial searching”. In Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, Pages 47-57, U.S,Jun. 1984.

[17] V.Gaede and O.Gunther, “Multidimensional Access Methods,” ACM Computing Surveys,vol.30,no.2,pp.170-231,1998.

[18] W.Y.Ma and B.S.Manjunath, “A comparison of wavelet transform features for texture image annotation,” In Proc.IEEE International Conference on Image Processing,1985.

[19] K.Lin,H.V.Jagadish, and C.Faloutsos. “The TV-tree:an index for high dimensional data,” Very Large Databases (VLDB)Journal,3(4),pp.57-543,1994.

[20] N.Katayama, S.Satoh, “The SR-tree: An index for high –dimensional nearest neighbor queries”, In Proc.of 1997 ACM SIGMOD International Conference on Management of Data, Tucson,Azorino,US,May 11- 15, pp.369-380,1997.

[21] S.Kiranyanz and M.Gabbouj, “Hierarchical cellular tree:An efficient indexing scheme for content based retrieval on multimedia based databases,” IEEE Transactions on Multimedia,vol.9.no.1.pp :102-119.Jan.2007.

[22] S.Kiranyanz,K.Caglar,O.Guldogan,and E.Karaoglu, “MUVIS: A multimedia browsing, indexing and retrieval framework,” In Proc.Third Int.Worshop on Content Based Multimedia Indexing,CBMI,Rennes,France,Sept 22-24,2003

[23] S.Kiranyanz,K.Caglar,E.Guldogan,O.Guldogan and M.Gabbouj, “MUVIS: A content-based multimedia indexing and retrieval framework,” in Proc.Seventh Int.Symposium on Signal Proc.and its Applications,ISSPA 2003,Parriss,France, Jul. 1-4, pp.1-8,2003.

[24] J.R.Kruskal, “On the shortest spanning subtree of a graph and the travelling salesman problem,” in Proc.AMS 71,1956

[25] P.Koikkalainen and E.Oja, “Self-organizing hierarchical feature maps,” in Proc.Int.Joint Conf.Neural Networks,San Diego,CA,1990.
[26] Manesh Kokare, Student Member IEEE, P.K. Biswas, Member IEEE and B.N. Chatterji, “Texture Image Retrieval Using New Rotated Complex Wavelet Filters,” IEEE Transactions on Systems, Man and Cybernetics - Part B, vol. 35, no. 6, pp. 1168-1178, Dec. 2005.

[27] D.B. Lomet and B. Salzberg, “The b+-tree: A multiattribute indexing method with good guaranteed performance,” ACM Transactions on Database Systems, 15(4), pp. 625-658, 1990.

[28] J.T. Laaksonen, J.M. Koskelo, S.P. Laakso, and E. Oja, “PicSOM-content-based image retrieval with self organizing maps,” Pattern Recognition Letter, vol. 21, no. 13-14, pp. 1199-1207, Dec. 2000.

[29] MUVIS. [Online]. Available:http://muvis.cs.tut.fi

[30] A. Motro Vague, “A user interface to relational databases that permits vague queries,” ACM Transactions on Office Information Systems, 6(3), pp. 625-658, 1990.

[31] A. Pentland, R.W. Picard, and S. Sclaroff, “Photobook: Tools for content based manipulation of images,” in Proc. SPIE (Storage and retrieval for image and video databases II), vol. 2185, pp. 34-37, 1994.

[32] R.C. Prim, “Shortest connection matrix network and some generalizations,” Bell Syst. Tech. J., vol. 36, pp. 1389-1401[2]

[33] Y. Rui, T. S. Huang, and S.-F. Chang, “Image retrieval: Current techniques, promising directions, and open issues,” J. Vis. Commun. Image Representation, vol. 10, pp. 39-62, Nov. 1999.

[34] J. Robinson, “The k-d-b-tree: A search structure for large multidimensional dynamic indexes,” in Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, Ann Arbor, Michigan, U.S., pp. 10-18, Apr. 1981.

[35] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, “Content-based image retrieval at the end of the early years,” IEEE Trans. Pattern Anal. Machine IntelI., vol. 22, pp. 1349-1380, Dec. 2000.

[36] T.K. Sellis, N. Roussopoulos, C. Faloutsos, “The R+-Tree: A Dynamic Index for Multi-Dimensional Objects,” in Proc. of the 13th International Conference on Very Large Data Bases, pp. 507-518, Sept. 01-04, 1987.

[37] J.R. Smith and Chang, “VisualSEEK: A fully automated content-based image query system,” in Proc. ACM Multimedia, Boston, MA, Nov. 96.

[38] Y. Sakurai, M. Yoshikawa, S. Uemura, H. Kojima, “The A-tree: An index structure for high-dimensional spaces using relative approximation.” In Proc. of the 26th International conference on Very Large Databases, pp. 516-26, September 10-14, 2000.

[39] I.K. Sethi and Coman, “Image retrieval using hierarchical self-organizing feature map,” Pattern Recognition Letter, vol. 20, pp. 1337-1345, 1999.

[40] Virage. [Online]. Available: www.virage.com

[41] D.A. White and R.Jain, “Similarity indexing with ss-tree.” In Proceedings of the 12th International Conference on Data Engineering, pp. 516-523, New Orleans, LA, U.S., Feb. 1996.

[42] H. Wang, C.-S. Peng, “The S tree: An Index Structure for successive matching of spatial objects”, In Proc. of 5th Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Hong Kong, 2001.

[43] P.N. Yianilos. “Data structures and algorithms for nearest search in general metric spaces,” In Proceeding of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 311-321, Philadelphia, PA, U.S., January 1993.

[44] P. Zezula, P. Ciaccia, and P. Rabitti, “M-tree: A dynamic index for similarity queries in multimedia databases,” In Technical Report 7, HERMES ESPRIT LTR Projects, 1996.

[45] H. Zhang and D. Zhong, “A scheme for visual features based image indexing,” In Proc. SPIE/IS&T Conf. Storage and Retrieval for Image and Video Databases III, San Jose, CA, vol. 2420, pp. 36-46, Feb. 9-10, 1995.

[46] X. Zhou, G. Wang, J.X. Yu, and G. Yu, “M+-tree: A new dynamical multidimensional index for metric spaces,” In Proc. Fourteenth Australian Database Conf. Database Technologies, Adelaide, Australia, pp. 161-168, Feb. 2003.

[47] T. Bozkaya and Z. M. Ozsoyoglu, “Distance-based indexing for high dimensional metric spaces,” in Proc. ACM-SIGMOD, pp. 557-368, 1997.

[48] Ming Zhang and John Fulcher, “Face recognition using artificial neural network group-based adaptive tolerance trees,” IEEE Trans. on Neural Networks, vol. 7, no. 3, May 1996.