Efficient Region Based Indexing and Retrieval for Images with Elastic Bucket Tries

Center For Visual Information Technology
International Institute Of Tehcnology
Hyderabad, 500032
Suman Karthik C.V.Jawahar
sumankarthik@students.iiit.ac.in, jawahar@iiit.ac.in

Abstract

Retrieval and indexing in multimedia databases has been an active topic both in the Information Retrieval and computer vision communities for a long time. In this paper we propose a novel region based indexing and retrieval scheme for images. First we present our virtual textual description using which, images are converted to text documents containing keywords. Then we look at how these documents can be indexed and retrieved using modified elastic bucket tries and show that our approach is one order better than standard spatial indexing approaches. We also show various operations required for dealing with complex features like relevance feedback. Finally we analyze the method comparatively and and validate our approach.

1 Introduction

Multimedia data is hard to index using regular database management systems. Traditionally the CBIR community widely preferred spatial data structures for deploying multimedia databases. These were predominantly R-Trees [6], X-Trees [2], S-Trees, Variants of S-trees and R-Trees like R*-Trees [1] ,S*-trees and later there were also TV-trees [7]. All of these tried to efficiently index spatial data or multi dimensional data like image, video and audio for building efficient retrieval systems. These data structures are better suited for global feature based image retrieval schemes than region based methods. Yet some have adapted them to work for region based image retrieval as done by Carson *et al.* [4]. However the situation has changed with the incorporation of relevance feedback into mainstream image retrieval. Once relevance feedback is used the traditional "spherical" or "window" queries in the feature space are transformed into highly elliptical and other shapes making the usual spatial data structures very inefficient. Since relevance feedback changes the query continuously in shape and dimension the spatial data structures have been found to be inefficient. Though much work has been done on improving the precision of retrieval by finding better relevance feedback schemes, little work has been done in the direction of data structures for region based image retrieval. The region based image retrieval problem is harder to tackle than the traditional global feature based retrieval as any indexing and querying on the global space is subdivided into many distinct queries in a region based image retrieval system.

Here we introduce a modified elastic bucket trie for indexing and retrieval scheme for image databases. It is much more efficient than the traditional spatial data structures used to access multimedia data and is at least one order better than these schemes. Our scheme is also able to work without any modification with relevance feedback schemes as is required by spatial indexing and retrieval schemes.

In this paper we propose a novel transformation of images into text documents where segments of the images become keywords or words of the text document. This virtual textual representation of images makes the CBIR problem a modified text retrieval problem and there by allowing us to employ modified elastic bucket tries to efficiently index and retrieve the images(Section 2). We define a modified elastic bucket trie for image retrieval and also provide a description of the operations that should be supported by such a structure to aid in efficiency without any performance trade off in contemporary CBIR architectures(Section 3). We then comparatively analyze our methods to validate their efficiency and performance.(Section 4). Finally we conclude with a few remarks(Section 5).

2 Virtual Textual Description

Images by their nature are subjective. Their content cannot be effectively described in a quantitative manner. When humans describe an image they do so by extracting ob-

jective features or concepts like sky, clouds, flowers, cars, bikes, people etc. This however cannot be done by a contemporary CBIR system as it is not capable of comprehending these concepts. Instead the image can be seen or interpreted by these systems in the form of primitive features. These low level features are computed from pixels or patches. There is a gap between these low-level representations and the high-level concepts, popularly known as the semantic gap. In order to bridge this gap of subjective visual features and objective high level concepts, Carson *et al.* [4] and Wang *et al* [8] developed an objective low level feature representation and retrieval framework called region based image retrieval. In these methods generally the image is divided into objective segments such that each segment is homogeneous in nature in some visual characteristics. Which means that the image is a collection of segments that are visually coherent concepts in themselves. The aim of region based image retrieval is to find some mapping of the concept that the user is looking for on to a set of segments [5]. If this can be successfully done the concept can be deduced as a set of segments by the system, there by being able to bridge the semantic gap to some extent. We take region based retrieval one step further by proposing that a set of visual segments representing a visual concept is much like a set of words representing a subjective intention, or like a set of words making a coherent essay with a central theme. Drawing such parallels to text documents we further try to objectify the visual concepts by converting the segments into words and the image into a text document comprising of these words.

In our virtual textual representation an image is referred to as a document and its segments are referred to as keywords. Such a transformation is both intuitive and advantageous as one can now solve the CBIR problem as a modified or a special case of text document retrieval problem. Once the image has been divided or partitioned into visually coherent and compact units or segments, each segment is transformed into a string called a keyword. These keywords are obtained by binning visual features and applying a linear or nonlinear transformation. The segments are transformed into words such that segments that are visually similar to each other have the least hamming distance in their strings. Such a transformation may at first seem lossy however such a transformation actually improves the generalization capabilities of the system. Once the segments have been transformed to keywords and the images converted to documents we cannot directly use cosine distance to find the distance between two images as done in text retrieval. This is because in text documents each word is an atomic unit where changing even a character would mean the meaning of the word is lost. However in our virtual textual representation each character is an atomic unit and these atomic units put together to form a keyword. Hence we need to solve the problem differently. The quantization or binning of the feature space itself poses some challenges. There is no one quantization scheme that'll fit all the problems well. Each problem must be studied to see which class of quantization techniques are suitable. The situation of one type of feature space being mismatched to an improper quantization or binning schema is not desirable. Such a mismatch will result in drastic deterioation of both performance and efficiency of the system. Suppose there is a feature space where the data points are uniformly distributed. In such a scenario a simple uniform quantizatio of each feature space axis into one symbol might be enough to bin the points. This would result in the featurespace being tranformed into uniform hyper cubes where each hypercube represents a possible string in the feature space. But such a scheme will not however work for a highly clustered feature space. Here uniform quantization would leave a lot of hypercubes or strings representing a small number or no datapoints at all. And other strings or hypercubes will be over crowded which will result in a drop in accuracy and efficiency. Such a clustered feature space would require us to use another method of quantization or building a codebook that is much more suited to it. One way to do this is to employ a density based feature space quantization scheme where the size of the hypercubes adaptively changes to fit a predetermined number of data points. Hence here a non uniform quantization of the feature space much better suited. The various ways in which the featurespace can be quantized so that it is suitable to our indexing schemes must yet be explored.

3 Elastic Bucket Tries

Tries are ordered tree data structures that are used as associative retrieval entities that retrieve a record for the given string. Bucket tries and elastic bucket tries [3] are variants that have the ability to pool various records with common key prefixes of a certain length into one bucket or block until the bucket overflows when more than N records are inserted into the bucket or block. Here N is the maximum number of records allowed in a block. Here it is advisable to have each block of size 4096 bytes or one page for the x86 architecture based systems. This ensures that any block is loaded into the main memory with the least amount of disk access which is the evident bottle neck. Here we have a special situation where all the possible strings or all the keywords of the document image are of the same length. So the maximum depth of the trie is $(m + 1)$ where m is the length of all the strings. The root node is a null character that acts as an entry point to all the other strings. Each level also has an extra $Null$ character node to accommodate for partial string matching in other than a prefix sense.

Buckets This data structure is designed to a cater to image databases of varying size from only a few hundred images to millions of images. Since this is for a dynamically scalable data structure and is designed to be deployed on anything from a workstation to a server it needs to allocate buckets or blocks on a demand basis. Though the entire trie can be populated with the leaves pointing to blocks right at the time of initialization, as the alphabet at each level is already known we do not do that because of efficiency and storage considerations. It is also due to the fact that a fully realized trie in the form of keys could be very sparsely populated as far as records go. This is the reason why new buckets are created or allocated only when existing buckets overflow.

Records Each record is a representative of a segment from an image in the database. It has the image name, handle or id. It has one string representative of the segment called the keyword of the segment. This keyword is used to decide which bucket this record falls into.

Insertion When a record r is to be inserted into a modified EBT(Elastic Bucket Trie) T the keyword of the record or the string representative of the segment within the record is obtained. From the root node r_n which is a null string the record descends through the trie until it reaches a bucket B at some level L such that $L \leq (m+1)$ where m is the size of all keywords or strings of the trie. Once the bucket is reached the record is inserted. If an overflow occurs the bucket is split into num_b new buckets where num_b is the size of the alphabet of the next character in the string. All the new buckets are placed one level lower than the original bucket after adding one character to the prefix of each bucket. This splitting though costly is used to dynamically allocate space to the records on demand rather than allocating all the space at once, and this splitting only continues till the level $(m+1)$ where an overflow will result in another bucket or block being appended to the original bucket to contain the overflow. Hence buckets at the bottom level are not split. Hence limiting the total number of splits to a constant number.

The modified EBT does not have any deletion mechanism for the records. This is in harmony with the cheap secondary storage and dynamically increasing multimedia databases of today where deletion is treated as an unnecessary overhead. We hence avoid all the costs of merging buckets or blocks.

Retrieval Retrieval in our modified EBT is very efficient and is designed for and incorporated into a region based image retrieval framework in such a way that the trie need not change to accommodate for the change in the query due to relevance feedback. Hence retrieval is made independent of the dynamic nature of the interactions between the user and the system. When an image I is given as a query and I is a set of all the segments representing the image then for each segment S_i. $T = T \cup Retrieve(S_i, EBT)$ where $i = 1 \rightarrow n$. Where n is the total number of segments in I and T is the set of all the records retrieved by querying for all the segments. The images whose handle occur the most are retrieved from storage in descending order ensuring that the image with the highest number of similar segments is first retrieved. Here partial segment matching is also taken care of due to the multiple levels at which buckets can occur. And every time there is relevance feedback from the user and the system is adapted a new pseudo image is given as a query and the same process continues over again.

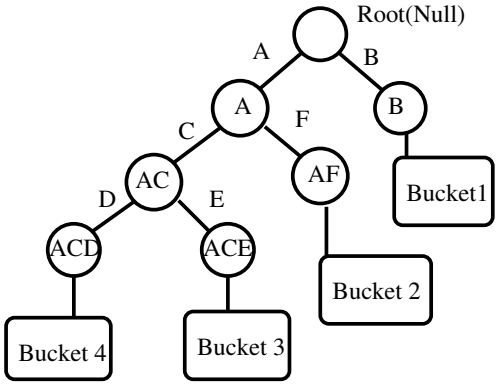

Figure 1. A Simple Bucket Trie

4 Analysis

It can be shown that the modified EBT is far superior to standard spatial data structures for indexing and retrieval in a region based framework with a simple comparative scenario. We analyze the costs associated with insertion and retrieval in an R-tree and our modified EBT by comparing the worst case scenario complexities in both R-tree and the EBT. A record r is inserted into both the R-tree R_t and the EBT T_r. Then this record must be retrieved from the data structure. We calculate the standard costs of these operations while ignoring their variable costs. Lets assume the number of dimensions of the feature space is the same as the string length of all the keywords in the trie which is m, this is true because here each character represents one dimension. We also assume that an equally variable number of node n_i exist at every level i of the structures as one needs equal ground to compare both the data structures. Splitting is not accounted for while counting the cost.

R-tree Following is the cost associated with insertion and retrieval in an R-tree for a given image I.

1. Obtain record r_i from image. –constant time C

Data Structure	R-Tree	EBT
Complexity	$2(m * n_i * C) * l * t$	$(n_i * C) * l * t$
Operations	Arithmetic	Logical
No. Of Splits	Indefinite	Fixed
RF Support	NO	YES
Efficiency	Low	High

Table 1. Comparison of R-tree and EBT

2. Start at root node of the R-tree.
   - Compare lower bound for m dimensions using floating point comparison n_i times. – cost of operation $m * n_i * C$
   - Compare upper bound for m dimensions using floating point comparison n_i times. – cost of operation $m * n_i * C$
3. If the target block is at level l repeat above l times. - cost of operation $2(m * n_i * C) * l$
4. If target block reached insert record or retrieve block. – constant cost C
5. Repeat from 2 t times where t is the number of segments in I. – Total cost $2(m * n_i * C) * l * t$

EBT Following is the cost associated with insertion and retrieval in a modified EBT.

1. Obtain record r_i from image. –constant time C
2. Start at root node of the R-tree.
   - Compare single character using EXOR n_i times – cost of operation $n_i * C$
3. If the target block is at level l repeat above l times. - cost of operation $(n_i * C) * l$
4. If target block reached insert record or retrieve block. – constant cost C
5. Repeat from 2 t times where t is the number of segments in I. – Total cost $(n_i * C) * l * t$

From the table 1 we see that the modified EBT clearly out performs the R-tree by an order. That is the EBT performs one order better than the R-Tree. Such performance improvement was made possible due to the transformation of images into documents and segments into keywords. Hence by converting the spatial indexing and retrieval with relevance feedback into a problem that can be solved by EBT we have overcome inefficiencies. This data structure is both scalable and adaptable with minimum change to other modules in the system. Its inherent capability to merge well with relevance feedback of any type makes it an ideal data structure in dynamic CBIR systems.

5 Conclusion

In this paper we have proposed a new and efficient indexing and retrieval scheme for region based image indexing and retrieval. This was built on top of a novel virtual textual representation for images. We have also breifly alluded to various ways in which this quantization of images to text can be achieved. We have also shown how our indexing scheme is robust to relevance feedback schemes and is able to work without any changes. We have also shown or validated our method by comparing it with one of the predominant spatial data structures used for indexing and retrieval in multimedia databases.

References

[1] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD*, pages 322–331. ACM Press, 1990.

[2] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The x-tree : An index structure for high-dimensional data. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases*, pages 28–39. Morgan Kaufmann, 1996.

[3] P. E. Black. Elastic bucket trie, dictionary of algorithms and data structures.

[4] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*. Springer, 1999.

[5] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *Proceedings of the International Conference on Computer Vision*, 2005.

[6] A. Guttman. R-trees: A dynamic index structure for spatial searching. In B. Yormark, editor, *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984*, pages 47–57. ACM Press, 1984.

[7] K.-I. Lin, H. V. Jagadish, and C. Faloutsos. The tv-tree: An index structure for high-dimensional data. *VLDB J.*, 3(4):517–542, 1994.

[8] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture LIbraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.