Implementation of Content based Image Retrieval using LBP and Avg RGB Algorithms

Manas. M. N.¹ Dr. Nagaraja.G. S.² Rajashekaramurthy.S³
¹PG student, ¹Professor, ¹Associate Professor Dept. of CSE
¹, ², ³RVCE, Bangalore, India

Abstract--- The effective multimedia information has drastically increased over the network since complexity and overheads are very high due to network traffic and network applications. Users are not satisfied with the traditional way of context or text based image retrieval due to more time and inaccurate results. Image retrieval is more challenging due to the large set of database in the network. In order to reduce the complexity and easy access a new technique is introduced to retrieve the image contents efficiently and more precisely. We proposed integrated solution by using Local Binary Pattern, Average RGB algorithms for extracting image features, which are then compared for similarity for more accurate results from the existing database.

Keywords: LBP(Local Binary pattern),CBIR(Content Based Image Retrieval)

I. INTRODUCTION

Content based image retrieval has become an area of wide interest nowadays in many applications such as crime prevention, medicine, military imagery, architecture and engineering design and historical research. The similarities between images are represented by image descriptors like texture, color, shape, intensity and so on. Then the image features of the query image and that of other database images are extracted with respect to the algorithm(s), and compared using Euclidean distance measure. Then the similarities of images are compared against a user specified threshold, and all images satisfying this threshold are output to the user. The proposed system is built using Java with Java Swing used for designing the Graphical User Interface (GUI) of the system.

Image retrieval is concerned with techniques for storing and retrieving images both efficiently and effectively. Early image retrieval methods locate the the desired images by matching keywords that are assigned to each image manually. However, as a result of the large number of images in collections, manual processing has become impractical. It is unlikely to foresee all the query keywords that will be used in a retrieval process. It is also impractical to assign keywords to every image, so the effectiveness of classic image retrieval is very limited.

Content-based image retrieval (referred to as CBIR), which is based on automatically extracted primitive features such as color, shape, texture, and even the spatial relationships among objects, has been employed since the 1990’s to meet the over increasing multimedia data.

“Content-based” means that the search will analyze the actual contents of the image, rather than user created contents like metadata which is used to refer an image. The term ‘content’ in this context might refer colors, shapes, textures, or any other information that can be derived from the image itself. Without the ability to examine image content, searches must rely on metadata such as captions or keywords. Such metadata must be generated by a human and stored alongside each image in the database.

In a CBIR system each image that is stored in the database has its features extracted and compared to the features of the query image. It involves two steps.

II. RELATED WORKS

The availability of images over the Internet emerged the imperative need to address the challenge of content based searching, in order to find visually similar content. Exhaustive search is infeasible for large scale applications due to its extensive time requirements. Thus, indexing methods are needed, able to provide efficient search time and retrieval accuracy. However, multimedia objects like compressed images, video and audio streams are usually described by sequences of descriptor vectors with over than a thousand dimensions, and their similarity is examined by nearest neighbor search. In the last decade, many dimensionality reduction methods and hashing techniques have been proposed to overcome the problems of the tree-based methods and thus to provide efficient solutions for high-dimensional data. In the first case, dimensionality reduction methods try to reduce the number of dimensions of the high-dimensional data. In particular, the data are transformed into a lower-dimensional space by using dimensionality reduction methods and then an index is built on it. In the second case, the encoding of such data into binary codes using appropriate hash functions enables higher scalability due to the compactness of the data and their efficient indexing. Similar high-dimensional objects are mapped to similar binary codes. Therefore, approximate nearest neighbor search is performed by only examining similar binary codes. However, the hashing methods often fail to achieve high accuracy in their approximations, especially when the hashing functions are drawn independently from the data, or when a short binary code length is selected. Moreover, for long binary code lengths a significant preprocessing time is required.

A. Tree-based Methods

Several tree-based indexing methods have been proposed for the problem of nearest neighbor search, such as: KD-trees [8], R-trees [10], M-Trees [7], Quad-Trees [20], Vantage Point [6] Trees (VPT), Voronoi Trees (VT), etc. Additionally, several tree-based indexing methods for approximate nearest neighbor search have also been proposed, such as: Spatial Approximation Tree (SAT), Approximating Eliminating Search Algorithm (AESA).
The main strategy for all tree-based indexing methods is to prune tree branches on the established bounding distances in order to reduce the node accesses. However, in high-dimensional spaces, where the multimedia objects lie, the tree-based indexing methods are inefficient, performing worse than exhaustive search.

**B. Dimensionality Reduction Methods**

Dimensionality reduction methods aim at mapping the data into a lower-dimensional subspace. The main idea is to make such a transformation without losing much information and build an index on the subspace. Many local and global dimensionality reduction methods have been proposed. Global dimensionality reduction methods map the whole dataset into a much-lower dimensional subspace. For example, the Isometric Feature Mapping method estimates geodesic distances and uses them to project the data into the embedded.

Local dimensionality reduction methods divide the dataset into correlated clusters and then each cluster is reduced in subspaces independently. For example, the Locally Linear Embedding method projects the data to a low-dimensional space, while preserving local geometric properties.

The preprocessing cost of such transformations is often high, due to dense matrix operations (especially products, eigenvector and eigenvalue calculations).

Dimensionality reduction methods can be used either for approximate or exact similarity search. In the first case, the similarity search is performed only into the transformed subspace. In the second case, firstly the similarity search is performed into the transformed space, where lower bounds on the distances are used for filtering, then a resulting set of candidates is returned, and finally the candidates are refined in the original space with exact search.

**C. Hashing Methods**

The basic idea of the hashing methods is (a) to encode the distances between the data into the form of compressed sequences of bits by using hash functions, and (b) to store the encoding distances into buckets, in order to ensure that the probability of collision is much higher for data that are close to each other than those that are far apart. Then, they approximate exact similarity measures by comparing hash codes, using a hamming distance on binary codes or other measures.

Different strategies are followed during the preprocessing for the generation of the binary codes. The existing hashing methods can be broadly categorized as data-independent and data-dependent.

In data-independent hashing methods, the hashing functions are defined independently from the data. One of the most popular methods is Locality Sensitive Hashing (LSH) [9], which is based on projection onto random vectors drawn from a specific distribution. According to the Johnson-Lindenstrauss Theorem, at least $O(\ln n/\epsilon^2)$ projection vectors are required (where $n$ is the dataset size), so as to preserve the pair wise distances with a relative error $\epsilon$. Therefore, in order to decrease the relative error and increase the probability that similar objects have similar hash codes, the random projection based methods require many random vectors to generate the hash tables (each table corresponds to one random vector), leading to a large storage space and a high computational cost. However, the data-independent hashing methods are often inefficient, especially for short lengths of binary codes, due to the fact that their hashing functions are drawn independently from the data.

In data-dependent hashing methods, the hashing functions are defined only for a preselected training dataset, which is usually a subset of the data, and involve similarity calculations for the training dataset. They try to fit the data distribution to the feature space in order to group the similar items and preserve locality. Notable examples of data-dependent hashing methods are: Spectral Hashing, which is based on spectral graph partitioning; K-means based hashing, which uses K-means clustering in the generation process of the binary codes; Subspaces Product Quantization, which decomposes the feature space into a Cartesian product of low-dimensional subspaces, each subspace is quantized separately, and the asymmetric distances are computed between the query and the quantized codes with the help of lookup tables; Kernelized Locality-Sensitive Hashing, which generalizes hashing to any Mercer kernel; Semi-Supervised Hashing, which exploits label information of the training set; Multiple Feature Hashing, which combines multiple features (i.e., global feature HSV color histogram and local visual features LBP) of videos, in order to learn the hash codes of the training key frames and a series of hash functions in a joint framework; Iterative Quantization, which minimizes quantization error by rotating zero-centered PCA projected data; Joint Optimization, which jointly optimizes both search accuracy and search time using compact binary codes; Random Maximum Margin Hashing which constructs hash functions by using large margin classifiers with arbitrarily sampled data points that are randomly separated into two sets.

In all aforementioned hashing methods, the most common technique for assigning the binary codes is to partition the metric space of the projected data points with appropriate hyper planes and set two different codes for each side. In a very recent approach, Spherical Hashing, hyperspheres (instead of hyper planes) are used, so as to partition the data points and to compute the binary codes. In the experimental evaluation of, authors showed that Spherical Hashing outperforms other state-of-the-art hashing methods.

**III. METHODOLOGY**

In the current CBIR system, a user can select a particular image from database as a query image. In this system features like color and texture are automatically extracted from query image and stored temporarily for comparison. In the similar way features are extracted from database images as well and then are compared with the features of query image and similar images are output. In this system, the user first selects an input query image and then has the freedom to select algorithm(s) (viz. Average RGB, LBP) for image retrieval. Then the image features of the query image and that of other database images are extracted with respect to the algorithm(s), and compared using Euclidean distance measure. Then the similarities of images are compared against a user specified threshold, and all images satisfying this threshold are output to the user. Two important features here are feature extraction and matching. Feature extraction
is a process to extract the image features to a distinguishable extent. Matching is the second step which involves matching the features to yield a result that is visually similar. Given a query image, with single / multiple object present in it; mission of this work is to retrieve similar kind of images from the database based on the features extracted from the query image. For this the features used are LBP (Local Binary Pattern), Average RGB Color.

IV. CONSTRAINTS

This product is entirely based on image processing and requires JPEG format of the image to process with. Only one query image can be given as an input to the CBIR system at a time. Algorithms like GCH, LCM and Average RGB work better on low quality images, whereas Edge detection and co-occurrence matrix work well on any quality.

V. ARCHITECTURAL STRATEGIES

The CBIR system is developed using Top-Down approach. In this method, the whole system is designed first then the components of the system are identified and built, finally all the components are assembled and integrated to build the required software. Java swing runs in the front end. MySQL is used to maintain the database. MySQL connector is being used in Eclipse IDE to connect java back end code with database. The CBIR system checks for the type of input file given as input, and it shows an error message if any other type of file is inserted instead of image file. The CBIR system also checks for the extension of the input image file, if it is other than JPEG format, it sends an error message to the user.

VI. SYSTEM ARCHITECTURE

Various design considerations shape the system architecture. These include end user requirements, services that the system is designed to provide, efficiency and performance considerations etc. The main purpose is to gain a general understanding of how and why the system is being decomposed, and how the individual parts work together to provide the desired functionality. The Fig-1 shows the System Architecture of Content Based Image Retrieval System.

VII. WORKING OF LOCAL BINARY PATTERN

In this method we have considered an image and the image is divided into pixels. We generally consider the odd value images like 3*3, 5*5 etc. From below fig we can see the center of image is considered and the difference is found and we find the threshold Then we multiple all 1s of the threshold with $2^0$,$2^1$ and so on. The Multiplication of 0 with powers of 2 has no effect.

VIII. AVG RGB

The image is considered and the pixel value of Red, Green, Blue are taken and the avg is considered. Considering the threshold values for Red,green,blue Avg RGB is found out. AvgRGB = Red + Green + Blue / 3.

IX. PREPROCESSING ALGORITHM

Feature extraction is a process where the features of the images are extracted then a collection of images in the database for image retrieval. Feature Vector is a collection of features of all images in the database stored in a vector. Similarity Matching occurs where the feature of input query image is compared with that of features of database images. Then images are retrieved i.e. Collection of output images are retrieved from the database after similarity matching.
The Algorithm works for sorting an image. First an LBP, Avg RGB values are calculated. Then it checks all the LBP values, and sorts the image. If LBP is same for the images then the image which has LBP same is sorted based on Avg RGB.

In the Structure chart of Reordering a Data Base system and input query processing system, set of images are considered and can be stored in the database. Similar features are extracted and we use LBP, Avg RGB. Considering the high level descriptor vectors of the images and we do the indexing and sorting. Fig 3 shows the Structure chart of reordering a Data Base system. Fig 4-system and input query processing system

![Fig. 4: system and input query processing system](image)

**X. EXPERIMENTAL ANALYSIS**

After sorting with LBP, AVG RGB, the reordered database looks as below in fig 5.

**Reordered Database**

<table>
<thead>
<tr>
<th>LBP</th>
<th>AVG RGB</th>
<th>Global Image Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 12424900</td>
<td>769280086</td>
<td>sahib</td>
</tr>
<tr>
<td>0 769280086</td>
<td>12424900</td>
<td>sahib</td>
</tr>
<tr>
<td>0 21825127</td>
<td>231042591</td>
<td>bon</td>
</tr>
<tr>
<td>0 231042591</td>
<td>21825127</td>
<td>bon</td>
</tr>
<tr>
<td>0 100185286</td>
<td>2147483647</td>
<td>sahib</td>
</tr>
<tr>
<td>0 2147483647</td>
<td>100185286</td>
<td>sahib</td>
</tr>
<tr>
<td>0 108146772</td>
<td>2147483647</td>
<td>sahib</td>
</tr>
<tr>
<td>0 2147483647</td>
<td>108146772</td>
<td>sahib</td>
</tr>
<tr>
<td>1 9437183</td>
<td>191701945</td>
<td>x</td>
</tr>
<tr>
<td>1 9437183</td>
<td>191701945</td>
<td>x</td>
</tr>
<tr>
<td>1 9500604</td>
<td>267467277</td>
<td>moh</td>
</tr>
<tr>
<td>1 267467277</td>
<td>9500604</td>
<td>moh</td>
</tr>
<tr>
<td>1 11014165</td>
<td>179702370</td>
<td>x</td>
</tr>
<tr>
<td>1 179702370</td>
<td>11014165</td>
<td>x</td>
</tr>
<tr>
<td>1 11014165</td>
<td>179702370</td>
<td>x</td>
</tr>
</tbody>
</table>

![Fig. 5: output of reordered database after sorting with LBP, AVG RGB](image)

**XI. CONCLUSION**

The CBIR systems uses just a single algorithm, the proposed CBIR system provides flexibility for the user in selecting appropriate algorithm(s) based on their preference. It uses Local binary pattern and Avg RGB for sorting the images. When comparing the images first it calculates the value of Local binary pattern and Avg RGB and image is sorted. If LBP value is same then the comparison is done based on Avg RGB values for sorting the image.

**ACKNOWLEDGEMENT**

We would like to thank RVCE research team for their moral support towards the project.

**REFERENCES**

[1] Eleftherios Tiakas, Dimitrios Rafailidis, Anastasios Dimou and petros Daras, Member, IEEE, “Multi Sort indexing for efficient Content based Image search and retrieval 2013 IEEE”


[4] Eleftherios Tiakas, Dimitrios Rafailidis, Anastasios Dimou and petros Daras, Member, IEEE, “Multi Sort indexing for efficient Content based Image search and retrieval 2013 IEEE”


