IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 10, Issue 1, Ser. I (Jan. - Feb. 2020), PP 20-28
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Indexing Images Using AMR for Efficient Storing And Retrieval

M.Anandha Kumar[1], Prof. M. Shanmuga Priya[2]

[1]*Ph.D Scholar/Associate Professor (CSE Department), MAM College of Engineering, Tiruchirappalli.*
[2]*Professor/Supervisor (ECE Department), MAM College of Engineering, Tiruchirappalli.*

Abstract: *In-memory memory mechanisms become the new trend for multiple high-performance systems such as Database and Real-time processing systems. With the proliferation of social networks and mobile multimedia applications, Approximate Nearest Neighbor (ANN) search has become even more demanding as it needs to maintain high search accuracy over a massive amount of multimedia content. To achieve a high search accuracy, ANN search framework focus on two main process: i) the efficient index/storage ii) the similarity search. An approximate media storm indexing mechanism was introduced to handle fast incoming images. This mechanism indexes the new descriptor vectors however it does not provide parallelization while indexing new descriptors. Also this mechanism provides high latency and high CPU issue while performing ANN search since multiple duplicate images will be indexed. Hence in the proposed method, the approximate media storm indexing mechanism utilizes the Map Reduce paradigm to parallelize the sequential indexing mechanism by indexing the images in a non-complex data structure and eliminate the duplicate indexing in the database. The query image is processed and retrieved with existing images using Euclidean and Manhattan distance method thus reducing the I/O latency while performing image search and retrieval.*

Keywords: *ANN, Multimedia Content, Indexing Mechanism, Map-Reduce Framework, Query based Image Retrieval.*

I. Introduction

Big data is a term for data sets that are so large or complex that traditional data processing application software's are inadequate to deal with them. Challenges include capture, storage, analysis, data curation, search, sharing, transfer, visualization, querying, updating and privacy. The term "big data" often refers simply to the use of predictive analytics, user behavior analytics, or certain other advanced data analytics methods that extract value from data, and seldom to a particular size of data set [1].

Data sets grow rapidly - in part because they are increasingly gathered by cheap and numerous information-sensing mobile devices, aerial (remote sensing), software logs, cameras, microphones, radio-frequency identification (RFID) readers and wireless sensor networks. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s; as of 2012, every day 2.5 Exabyte (2.5×1018) of data are generated. Relational database management structures and computing device records- and visualization-programs often have difficulty coping with big records. The work can also require "massively parallel software program running on tens, hundreds, or even hundreds of servers". What counts as "big information" varies depending on the competencies of the customers and their equipment, and increasing competencies make massive records a moving goal.

Definition

The term big data has been in use since the 90s, with a few giving credit score to John Mashey for coining or at the least making it popular. Big data generally includes data sets with sizes past the potential of generally used software gear to capture, curate, manipulate, and process information within a tolerable elapsed time. Big data "length" is a constantly shifting target, as of 2012 starting from some dozen terabytes to many petabytes of information.

In a 2001 research record and associated lectures, META Group (now Gartner) defined data increasing challenges and possibilities as being three-dimensional, i.e. Increasing volume (quantity of facts), velocity (speed of facts inside and outside), and variety (range of information kinds and sources). Gartner, and now much of the enterprise, maintain to apply this "3Vs" version for describing large information. In 2012, Gartner updated its definition as follows: "Big data is high volume, high velocity, and/or high varietyof information that require new types of processing to enable stronger decision making, perception discovery and process optimization."

Big Data Analytics

Big data analytics is the process of examining large data sets to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. The primary goal of big data analytics is to help companies make more informed business decisions by enabling data scientists, predictive modelers and other analytics professionals to analyse large volumes of transaction data, as well as other forms of data that may be untapped by conventional Business Intelligence (BI) programs. That could include Web server logs and Internet click streams data, social media content and social network activity reports, text from customer emails and survey responses, mobile-phone call detail records and machine data captured by sensors connected to the Internet of Things [2].

Content Based Image Retrieval

Content based image retrieval (CBIR) has been an active research area since 1970. It applications has increased many fold with availability of low price disk storages and high speeds processors. Image databases containing millions of images are now cost effective to create and maintain. Image databases have significant uses in many fields including medicines, biometric security and satellite image processing. Accurate image retrieval is a key requirement for these domains.

The visual contents of images, such as color, texture, shape [14] and region [15], are extensively explored for indexing and representation of the image contents. These low level features of an image are directly related to the contents of the image. These image contents could be extracted from image and could be used for measuring the similarity amid the queried image and images in the database using different statistical methods. In content-based retrieval systems different features of an image query are exploited to search for analogous images features in the database [16][17].

Image Storm Indexing Mechanism

Multimedia has becoming an inevitable part of any presentation. It has found a variety of applications right from entertainment to education. The evolution of internet has also increased the demand for multimedia content. Multimedia uses multiple forms of information content and their processing e.g. text, audio, video, graphics, animation and interactivity to inform or entertain its user'.

Approximate nearest neighbor (ANN) search over large amounts of multimedia content is an interesting and fundamental problem in multimedia analysis and retrieval [3][4]. With the proliferation of social networks and mobile multimedia applications, ANN search has become even more demanding, as it needs to maintain high search accuracy over a massive amount of multimedia content [10]. To achieve high search accuracy, ANN search frameworks focus on two main processes: i) the efficient index/storage; and ii) the similarity search.

Several real-world applications require an efficient media storm indexing algorithm, for example, the continuously growing and massive image collections that come from social networks like Flickr, a public picture sharing site, which reported that it has received 1.42, 1.6 and 1.83 million per day, on average, in 2012, 2013 and 2014, respectively, making clear that the number of incoming images will keep increasing every year Processing media storms is computational intensive; therefore, the challenge is to support a fast indexing mechanism to store and accurately search over these massive and fast incoming image collections and correctly index them in a limited time to maintain high search accuracy.

Meanwhile, indexing media storms has been studied in [3]; however, after the media storms have been indexed, the complex data structures of multiple randomized KD-trees are used, thus, having high computational cost in the online search [5]. Additionally, all the aforementioned similarity search strategies are designed as a disk-based processing mechanism, which requires a vast amount of I/O operations in order to process the incoming image collections.

Therefore, performing a distributed indexing mechanism of media storms poses the following challenges:

i) Scalability is a required characteristic of indexing media storms in order to index a massive amount of images in a streaming mode into distributed databases and efficiently search in high accuracy;

ii) Indexing media storms should be performed at low latency and no bottlenecks should be experienced by the incoming image collections;

iii) To perform an efficient distributed mechanism, a robust communication messaging system is needed in order to exchange data using message passing.

II. Related Work

DimitriosRafailidis [6] proposed a parallel media storm indexing algorithm in the CDVC framework, which is built on Flink1, an open-source platform for stream data processing. The reasons for selecting Flink, instead of other platforms, such as Spark2, are because Flink provides a better resource usage of the cluster when recovery is required, as well as, Flink provides incremental iterations in its dataflow model. To verify that the media storms have been correctly indexed in CDVC, evaluated the search accuracy of CDVC by indexing media storms with the sequential indexing algorithm of, and the proposed approach.

MohammadNorouzi [7][18][13] proposed an approach called multi-index hashing, as binary codes from the database are indexed m times into m different hash tables, based on m disjoint substrings. Given a query code, entries that fall close to the query in at least one such substring are considered neighbor candidates. Candidates are then checked for validity using the entire binary code, to remove any non-r-neighbors. To be practical for large-scale datasets, the substrings must be chosen so that the set of candidates is small and storage requirements are reasonable. The key idea here stems from the fact that, with n binary codes of q bits, the vast majority of the 2q possible buckets in a full hash table will be empty, since 2q _ n.

A novel approximate indexing scheme for efficient content-based image search and retrieval is presented, called Multi-Sort Indexing (MSIDX) [8][9]. The proposed scheme analyses high dimensional image descriptor vectors, by employing the value cardinalities of their dimensions. The dimensions' value cardinalities, an inherent characteristic of descriptor vectors, are the number of discrete values in the dimensions. As expected, value cardinalities significantly vary, due to the existence of several extraction methods. Since dimensions with high value cardinalities have more discriminative power, a multiple sort algorithm is used to reorder the descriptors' dimensions according to their value cardinalities, in order to increase the probability of two similar images to lie within a close constant range. The proposed scheme is fully suitable (a) for real-time indexing of images, and (b) for searching and retrieving relevant images with an efficient query processing algorithm.

HerveJegou [10] proposed method constructs short code techniques using quantization. The goal is to estimate distances using vector to centroid distances, i.e., the query vector is not quantized; codes are assigned to the database vectors only. This reduces the quantization noise and subsequently improves the search quality. To obtain precise distances, the quantization error must be limited. Therefore, the total number k of centroids should be sufficiently large, e.g., k = 264 for 64-bit codes. This raises several issues on how to learn the codebook and assign a vector. First, the number of samples required to learn the quantizer is huge, i.e., several times k. Second, the complexity of the algorithm itself is prohibitive. Finally, the amount of computer memory available on Earth is not sufficient to store the floating point values representing the centroids.

Mohammad Norouzi [11] proposed a method for learning similarity- preserving hash functions that map high- dimensional data onto binary codes. The formulation is based on structured prediction with latent variables and a hi

nge-like loss function. Compact binary codes are particularly useful for ANN. If the nearest neighbours of a point are within a small hypercube in the Hamming space, then ANN search can be performed in sub linear time, treating binary codes as hash keys. Even for an exhaustive, linear scan through the database, binary codes enable very fast search. To preserve a specific metric (e.g., Euclidean distance) one can use binary similarity labels obtained by thresholding pair- wise distances. The loss function we advocate is specific to learning binary hash functions, and bears some similarity to the hinge loss used in SVMs. It includes a hyper-parameter, which is a threshold in the Hamming space that differentiates neighbours from non-neighbours.

MayurDatar [20] proposed a novel Locality-Sensitive Hashing scheme for the Approximate Nearest Neighbour Problem, based on portable distributions. The proposed scheme improves the running time of the earlier algorithm for the case of the l2 norm. It also yields the first known provably efficient approximate NN algorithm for the case $p < 1$. We also show that the algorithm finds the exact near neighbour in time for data satisfying certain “bounded growth” condition. Proposed algorithm also inherits two very convenient properties of LSH schemes. The first one is that it works well on data that is extremely high-dimensional but sparse. The second property is that our algorithm provably reports the exact near neighbour very quickly, if the data satisfies certain bounded growth property.

YannisAvrithis [17] proposed an efficient data structure for approximate nearest neighbour search that explores different randomization strategies, and an efficient implementation, GeRaF, that is found competitive against existing implementations of several state-of-the-art methods. PQ is consistently faster and more accurate at search, but is significantly slower to build, which is impractical when the dataset is updated. This method is consistent on both synthetic and real datasets of a wide range of dimensions and cardinalities. Proposed approach omits backtracking problem, and optimize distance computations, thus accelerating queries.

To avoid the construction of complex data structures that cannot handle the high-dimensional data, hashing methods have been widely used. Hashing strategies are divided into data-independent and data-dependent and are widely used for ANN search because of their low storage cost and fast query speed. The main goal of hashing strategies, such as Spectral Hashing [12], Iterative Quantization [11] and Anchor Graph Hashing [13] is to preserve the similarities of the training data using linear or nonlinear functions when projecting the data to the Hamming space. However, the aforementioned hashing strategies provide no parallelization while indexing new descriptors and performing ANN search.

III. Methodologies and Techniques

Image dataset consist of images from different multimedia sources. Each image has various features: Color, Shape, Texture, Edge and Corner. For fast and improve Image retrieval performance we are using color and corner feature extraction. The images are kept in database called image database.

Image Pre-processing:

The purpose of pre-processing is to improve image data by suppressing unwanted distortions and enhances some important feature of the image for further processing. Pre-processing takes the input as set of images in D-dimensional descriptors which are stored in the distributed databases, applies Gray scale conversion and Gaussian Smoothing methods and generates the dimensional value cardinality vectors C(m)based on M different predefined lower lb(m) and upper ub(m) dimensions' bounds, with m = 1.M different dimension value cardinality vectors C(m) are merged into a global dimension value cardinality vector C. The output is a priority index vector p which is calculated by sorting the global dimension value cardinality vector C in a descending order.

A. Gray Scale Conversion:

Grayscale conversion is simply reducing the complexity: from a 3D pixel value (R, G, and B) to 2D (B and W) value. Grayscale is the collection or the range of monochromic (Gray) shades, ranging from pure white on the lightest end to pure black on the opposite end. Grayscale only contains luminance (brightness) information and no color information; that is why maximum luminance is white and zero luminance is black; everything in between is a shade of Gray. That is why grayscale images contain only shades of Gray and no color.

The methods that are implemented in this approach to convert an image from RGB to Gray scale is the Weighted or Luminosity method. It averages the values (R,G and B), but it forms a weighted average to account for human perception. So the new formula for luminosity is (0.3* R) + (0.59* G) + (0.11 * B).

B. Image Smoothing:

Image filtering is often used to reduce noise within an image (low pass filter) or to enhance an image (high pass filter). For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening and edge enhancement. In Image processing, a kernel or convolution matrix or mask is a small matrix used for blurring, sharpening, embossing, edge detection and more. This is accomplished by doing a convolution between a kernel and an image.

Most smoothing methods are based on low pass filters that are employed to remove high spatial frequency noise from a digital image. An image is smoothed by decreasing the disparity between pixel values by averaging nearby pixels. A low pass filter tends to retain the low frequency information within an image while reducing the high frequency information.

Various types of low pass filters are available in image smoothing. In this proposed approach, Gaussian smoothing is employed. The Gaussian smoothing operator is a 2-D convolution operator that is used to `blur' images and remove detail and noise. In this sense it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian (`bell-shaped') hump. Mathematically, applying a Gaussian smoothing to an image is the same as convolving the image with a Gaussian function. Equation (1) gives the formula of a Gaussian function in one dimension.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad (1)$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution. We have assumed that the distribution has a mean of zero (i.e., it is centered on the line x=0).

In two dimensions, it is the product of two such above Gaussian functions, one in each dimension given in equation (2).

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

Values from this distribution are used to build a convolution matrix which is applied to the original image. The 2D Gaussian distribution is illustrated in figure 1.

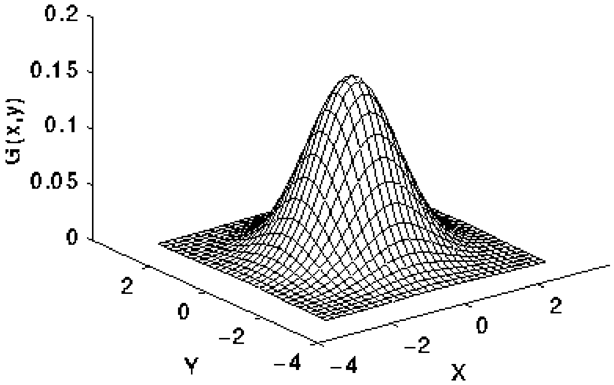
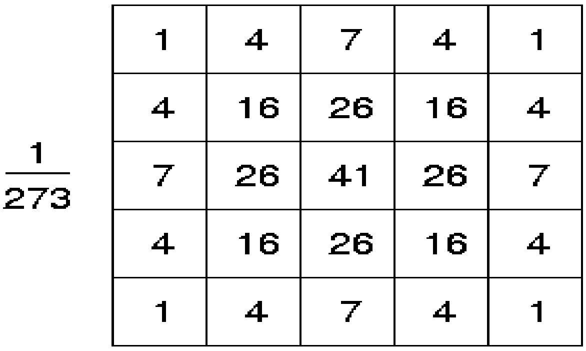


Figure 1: 2D Gaussian distribution with mean (0,0) and σ = 1

The idea of Gaussian smoothing is to use this 2-D distribution as a `point-spread' function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. Figure 2 shows a suitable integer-valued convolution kernel that approximates a Gaussian with a σ of 1.0



$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Figure 2: Discrete approximation to Gaussian function with σ = 1.0

Corner Feature Extraction:

Feature extraction is the process of by which certain features of interest within an image are detected and represented for further processing. It is a critical step in image processing solutions because it marks the translation from pictorial to non-pictorial data representation. The resulting representation can be subsequently used as an input to a number of pattern recognition and classification techniques.

The features of the image are represented in the feature vector form which represents the object. In the domain of image retrieval, each 'n' dimensional feature vector may be considered as a point in the 'n' dimensional vector space. Thus, a feature vector is mapped to a point in the 'n' dimensions. Each image has following features: Color, Texture, Shape, Corner and Edge. For fast and improve Image retrieval performance, corner feature extractions are used in our implementation.

Corner detection is an approach used to extract certain kinds of features and infer the contents of an image. A corner is a point whose local neighborhood stands in two dominant and different edge directions. Corners are the important features in the image and they are termed as interest points which are invariant to translation, rotation and illumination. In this proposed method, Harris corner detection algorithm is employed to extract corner feature from the image. This algorithm defines a corner to be a point with low self-similarity.

Algorithm for Harris Corner Extraction:

Step 1: Color to Grayscale: The first step is to convert the input image into a grayscale image, which will enhance the processing speed.

Step 2: Compute x and y Gaussian derivatives at each pixel.

Step 3: Compute second moment matrix M in a Gaussian window around each pixel.

Step 4: Compute corner response function R.
Step 5: Threshold R.
Step 6: Find local maxima of response function.

Image indexing with map-reduce framework:
An approximate indexing mechanism is proposed using the Map Reduce paradigm (AMR) to reduce the high latency and high CPU issue. Two approaches of the media storm indexing mechanism are described:

i) An exact Map Reduce approach that is presented in and identifies the position of the incoming descriptors in the double linked list L;

ii) A Map Reduce approach that assigns the logical position to each descriptor in an approximate manner using the notion of the root descriptor, in order to reduce the computational complexity and latency of the exact approach.

At training time, the Feature Map Reduce is empty, where the Map distributes feature vector according to the image id to different nodes. Map Function is the first step in Map Reduce Algorithm which distributes features according to the image id along with the calculated priority vector p. Each mapper reads the data iteratively as a key/value pair record, by combining the descriptors with same primary key, process it and outputs key/value pair bound for a Reduce function.

The Feature Map subsamples the input features by emitting one out of every input skip features, thus duplicate images will not be indexed again if the feature vectors are matched. If not, all records with the same key go to the same Reduce task. The Reduce function merges the features of different vectors and thus indexing the incoming features.

Algorithm for Approximate Indexing Mechanism with Map Reduce (AMR):
MAP1: The incoming descriptors χ are divided into M computational nodes. Each node takes input as viϵ χ and the priority index. The value of the first dimension of the descriptor vi is set as a primary key pk. Theoutput of the MAP 1 phase is a pair <pk, vi>.
COMBINE: The descriptors with the same primary key pk are grouped together in the same set χpk⊆ χ. The output of the COMBINE phase is a pair <pk, χpk>.
MAP2: For each primary key pk, the m-th computational node, with m ∈ 1 . . . M, fetches the sets χpk from the COMBINE phase. Then, the m-th node compares and reorders the subset χpk of the incoming descriptors along with the subset Vpk of the already stored descriptors primary key pk and eliminates the descriptors which are matched.
REDUCE: The M different set of descriptors produced by the MAP 2 phase are merged, thus indexing the incoming descriptors of set χ.

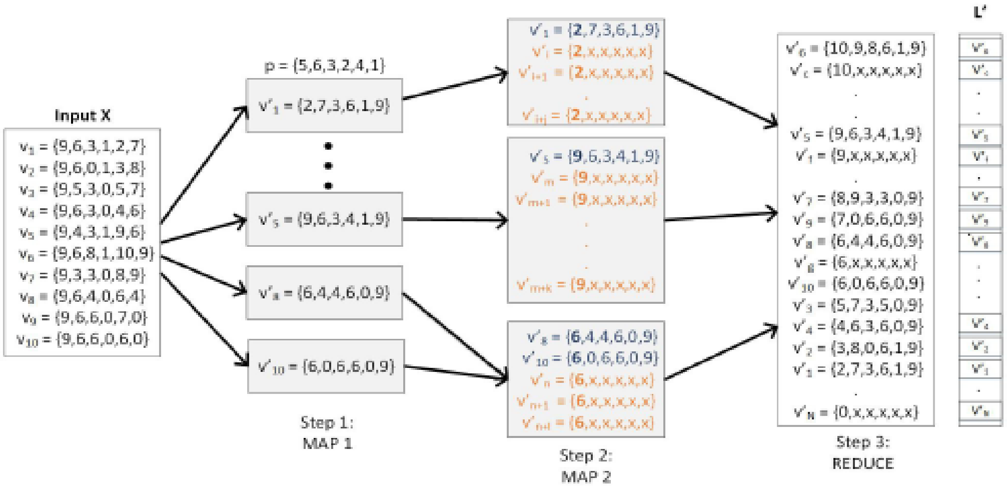


Figure 3: Running example of the proposed indexing mechanism. In the Map2 step, blue fonts denote the sets χpk of the incoming batch χ, while red fonts denote the already stored descriptors Vpk, with the same primary key pk.

Fig 3 presents a running example of the proposed approximate media storm indexing mechanism using map reduce paradigm. The input is the set of χ =10 descriptors v_i, where iϵ 1..10, with D=6 dimensions and the priority index vector p = {5; 6; 3; 2; 4; 1}. In the running example, ten descriptors are considered as an incoming batch at a time step of one second within the storming time frame. In Step #1, the dimensions of the ten descriptors are reordered according to priority index vector p, generating ten reordered descriptors v'. In Step #2, according to the primary key pk, the reordered vectors χpk are grouped with the preprocessed vectors in the respective set Vpk of the already stored descriptors. For instance, χpk={v'_i} is grouped with Vpk={v'_i ,…,v'_i+j}, because they have the same primary key pk=2. The respective group of descriptors are reordered, thus generating M different sublists L(m). In Step #3 of Figure 2,the M different sublists L(m) are merged to update the double linked list L accordingly and to store the incoming descriptors of χ.

Query Processing and Image Retrieval:

In this phase, the query image is processed same as above process namely pre-processing and Gaussian smoothing and thus feature vector of query image is obtained. The feature vector of query image is compared with existing feature vectors that are stored in the image database. While comparing two feature vectors, we calculate the distance between two vectors. We have used methods such as Euclidean distance and the Manhattan distance for calculating the matching distance. According to the calculated distance, existing images are sorted in ascending order (ranking).

A. Euclidean Distance

Input: X = {x1, x2, x3,…..,xn} be the set of data points , Y= {y1,y2,y3…yn} be the set of data points and V = {v1,v2,v3,….,vn} be the set of centers.

Step 1: Select 'c' cluster centers arbitrarily

Step 2: Calculate the distance between each pixels and cluster centers using the Euclidean Distance metric as follows:

$$\mathrm{Dist}(X, Y) = \sqrt{\sum_{j=1}^{n}(X_{ij} - Y_{ij})^2} \quad (3)$$

Where X, Y are the set of data points

Step 3: Pixel is assigned to the cluster center whose distance from the cluster center is minimum of all cluster centers

Step 4: New cluster center is calculated using

$$V_i = \frac{1}{C_i}\sum_{1}^{ci} x_i \qquad (4)$$

where V_i denotes the cluster center, c_i denotes the number of pixels in the cluster

Step 5: The distance among every pixel and new obtained cluster facilities is recalculated

Step 6: If no pixels were reassigned then stop otherwise repeat steps from 3 to 5

B. Manhattan Distance:

Input: X = {x1, x2, x3,…..,xn} Y= {y1,y2,y3,…,yn}be the set of data points and V = {v1,v2,v3,….,vn} be the set of centers.

Step 1: Select 'c' cluster centers arbitrarily

Step 2: Calculate the distance between each pixels and cluster centers using the Manhattan metric as follows:

$$\mathrm{Dist}(X, Y) = \|x_{ij} - y_{ij}\| (5)$$

Step 3: Pixel is assigned to the cluster center whose distance from the cluster center is minimum of all cluster centers

Step 4: New cluster center is calculated using

$$V_i = \frac{1}{C_i}\sum_{1}^{ci} x_i (6)$$

where x_i denotes as data points, V_i denotes as cluster centroids and ci denotes the number of pixels in the cluster

Step 5: The distance between each pixel and new obtained cluster centers is recalculated

Step 6: If no pixels were reassigned then stop otherwise repeat steps from 3 to 5

For similarity comparison between the query image and the database picture. Using a suitable threshold, images that are semantically closer are retrieved from the database and displayed as a thumbnail.

The proposed architecture for content based image retrieval is as follows:

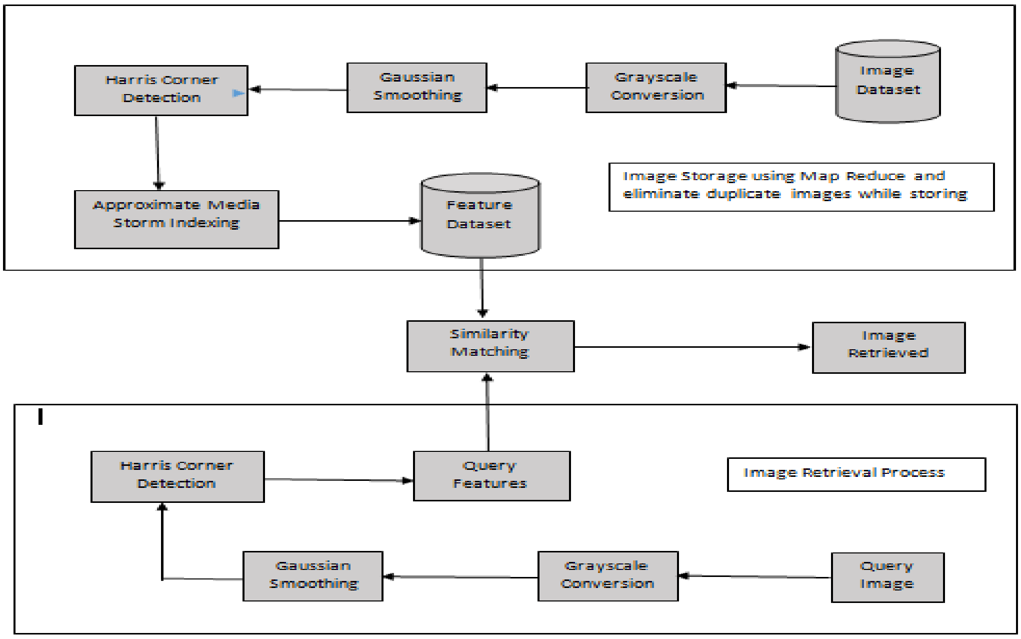


Figure 4: Proposed work Architecture

IV. Experimental Results

Experimental results have the results of the proposed work. Proposed work was implemented using MATLAB tool. Experimental result shows that the proposed work achieves efficient image storage and retrieval.

V. Conclusion

In this paper an indexing mechanism of multimedia streaming method is proposed which is based on the performance analysis of various distance metrics. The dramatic rise in the sizes of images databases has stirred the development of effective and efficient retrieval systems. The application performs a simple corner-based search in an image database for an input query image, which are similar to the input image as the output. The number of search results may vary depending on the number of similar images in the database. Map Reduce framework was constructed to remove the repeated occurrence of same images. Indexing mechanism provided to perform easy image retrieval from large dataset. Indexing was developed based on the features of the images. The similarity metrics have been used based on distances like Euclidean distance and Manhattan distance. The corner features of the image are used to extract the number of images based on the query image as input. Similarity comparison, extracting feature signatures of every image based on its pixel values and defining rules for comparing images. Distance metric or matching criteria is the main tool for retrieving similar images from large image databases for all the above categories of search. The Manhattan distance is used to determine similarities between a pair of images in the content based image retrieval application.

References

[1]. H. Zhang, G. Chen, B. C. Ooi, K. Tan, and M. Zhang, "In-memory big data management and processing: A survey," IEEE Transition Knowledge Data Eng., vol. 27, no. 7, pp. 1920–1948, 2015.

[2]. R. Want, B. N. Schilit, and S. Jenson, "Enabling the internet of things," IEEE Computer, no. 1, pp. 28–35, 2015.

[3]. D. Moise, D. Shestakov, G. T. Gudmundsson, and L. Amsaleg,"Terabyte-scale image similarity search: Experience and best practice,"in Proceedings IEEE International Conference on Big Data, 2013,pp. 674–682.

[4]. M. Muja and D. G. Lowe, "Scalable nearest neighbour algorithmsfor high dimensional data," IEEE Transactions on Pattern Analysisand Machine Intelligence, vol. 36, no. 11, pp. 2227–2240, 2014.

[5]. S. Antaris and D. Rafailidis, "Similarity search over the cloud based on image descriptors' dimensions value cardinalities," IEEE Transactions on Multimedia Computing, Communications and Applications,vol. 11, no. 4, p. 51, 2015.

[6]. D. Rafailidis and S. Antaris, "Indexing media storms on flink," inBig Data (Big Data), 2015 IEEE International Conference on, Oct 2015, pp. 2836–2838.

[7]. E. Tiakas, D. Rafailidis, A. Dimou, and P. Daras, “Msidx: Multi-sortindexing for efficient content-based image search and retrieval, ”IEEE Transaction on Multimedia, vol. 15, no. 6, pp. 1415–1430, 2013.

[8]. Norouzi, Mohammad, Ali Punjani, and David J. Fleet. "Fast exact search in hamming space with multi-index hashing." IEEE transactions on pattern analysis and machine Intelligence 36, no. 6 (2014): 1107-1119.

[9]. M. Norouzi, A. Punjani, and D. J. Fleet, “Fast exact search in hamming space with multi-index hashing,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 6, pp. 1107–1119, 2014.

[10]. Jegou, Herve, MatthijsDouze, and CordeliaSchmid (2011),‘Product quantization for nearest neighbor search’, IEEE transactions on pattern analysis and machine intelligence, Vol.33, No. 1,pp.117-128.

[11]. Y. Gong and S. Lazebnik, “Iterative quantization: A procrustean approach to learning binary codes,” in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, June 2011, pp. 817-824.

[12]. Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in Advances in Neural Information Processing Systems 21, D. Koller,D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2009, pp. 1753–1760.

[13]. W. Liu, J. Wang, and S. fu Chang, “Hashing with graphs,” in InICML, 2011.

[14]. A. J. K. Iqbal, M. O. Odetayo, “Content-based image retrieval approach for biometric security using colour, texture and shape features controlled by fuzzy heuristics,” Journal of Computer and System Sciences, vol. 78,p. 12581277, 2012.

[15]. X. L. M.A. Nascimento, V. Sridhar, “Effective and efficient region-based image retrieval,” Journal of Visual Languages and Computing, vol. 14,pp. 151-179, 2003.

[16]. X. Q. M. Royal, R. Chang, “Learning from relevance feedback sessions using a k-nearest-neighbor-based semantic repository,” IEEE International Conference on Multimedia and Expo (ICME07), Beijing, China,pp. 1994–1997, 2007.

[17]. X. Q. K. Shkurko, “A radial basis function and semantic learning space based composite learning approach to image retrieval,,” IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP07), vol. 1, p. 945-948., 2007.

[18]. A. Babenko and V. Lempitsky.The inverted multi-index. In Proc.IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[19]. D. Greene, M. Parnas, and F. Yao. Multi-index hashing for information retrieval. In IEEE Symposium on Foundations of Computer Science, pages 722–731, 1994.

[20]. C. Strecha, A. Bronstein, M. Bronstein, and P. Fua.Locality-Sensitive Hashing: improved matching with smaller descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(1):66–78, 2012.