Int. j. inf. tecnol. (February 2022) 14(1):49–68
https://doi.org/10.1007/s41870-021-00806-8

ORIGINAL RESEARCH

Content based image retrieval on big image data using local and global features

Suresh Kumar Kanaparthi[1] · **U. S. N. Raju**[1]

Received: 13 May 2021 / Accepted: 6 September 2021 / Published online: 23 September 2021

Abstract In this paper, processing of huge number of images is achieved to retrieve a queried image using MapReduce paradigm with different modes. These systems are useful in cases where the traditional single computer cannot process such huge image data. Nevertheless, such processing with a single computer system will take a long time to complete the processing. A total of six types of modes for processing the image data is proposed in this paper. To show the performance of the systems, the results are shown with different number of workers involved in processing the image data. The results show that the proposed MapReduce paradigm with different modes are performing as expected when there is a change in the number of workers involved in processing i.e., the time taken to complete the job is indirectly proportional to the number of workers considered. Even though the time to complete the task has changed, the performance measures: Precision, Recall, F-Measure, Retrieval Rank and Minimum Retrieval Epoch are same for all modes. The computational time for two image datasets: Corel1K and VisTex for a total of five image retrieval methods are evaluated. For completing all the five image retrieval methods on Corel1K, the time saved is 43%, 45% and 68% respectively for the number of workers as 4*vs*2, 2*vs*1 and 4*vs*1 workers. Similarly for VisTex it is 42%, 46% and 68%. The algorithm used for getting the features from the image are the authors recently published algorithms.

Keywords Big image data · MapReduce · CBIR · Local and global features

1 Introduction

In present days, exponential increase in usage of digital cameras and mobile phones makes the size of image dataset gigantic. Maintaining such kind of large image dataset is an extremely tedious and troublesome job. So, efficient technique is required to retrieve desired images from such kind of huge image dataset. One of the effective solutions of such retrieval problem is Content Based Image Retrieval (CBIR). The term "content" signifies that images are retrieved based on some features which can be calculated from the actual content of images. Retrieval process depends on similarity between query image and all the images of image dataset. Feature vector comparison is one of possible way to find similarity between the corresponding images. Features of an image can be classified as Color features, Texture features and Shape features.

A color model can be defined as a coordinate system where each point uniquely describes a color. One of the widely used color model for CBIR is RGB. RGB color model has a limitation that the color information contained in R, G and B channels is highly correlated thereby not being informative for texture and shape features directly. The color model which gives information about dominant color, its purity and brightness is HSI (Hue, Saturation, Intensity). Hence, it is used in our proposed method. In HSI model, hue represents the dominant color and saturation represents the degree to which a pure color is diluted by

✉ U. S. N. Raju
usnraju@nitw.ac.in

Suresh Kumar Kanaparthi
sureshkonline@gmail.com

[1] Department of Computer Science and Engineering, National Institute of Technology Warangal, Warangal, Telangana State 506004, India

white light whereas intensity represents the brightness of a pixel. Both hue and saturation together give the complete color feature. The HSI model decouples color-carrying information from intensity component of an image. Intensity indicates the value of a color which is related to texture. Sadegh Fadaei et al. [1] proposed a new CBIR scheme where uniform partitioning scheme is applied on HSI color model to calculate Dominant Color Descriptor (DCD). Various curvelet and wavelet features are used as texture features to subdue the problem of image translation and image noise. Color and texture features are often concatenated to improve the performance of CBIR.

1.1 Content based image retrieval

The name Content Based Image Retrieval (CBIR) implies that the image retrieving process is based on contents of the image. So color, texture and shape information are chosen as features of an image. To produce color features of an image, feature extraction procedures like color histogram [2], color correlogram [2], color autocorrelogram [3, 4], inter-channel voting between hue and saturation [5] can be applied on image. For texture feature extraction, LBP [6], ULBP [6], CS_LBP [7], LEP [8], LDP [9], LTrP [10] can be used. One more texture feature descriptor is GLCM, which reveals knowledge about pixel pair co-occurrence of the image [11, 12]. For extracting shape information, HOG [13], angular pattern and binary angular pattern [14], Wavelet Fourier descriptor [15], Convex Hull [16] etc. can be used. In [17–19], different image retrieval methods have been covered and discussed.

1.1.1 Local binary patterns

One of the most effective texture feature used in CBIR is Local Binary Patterns (LBP) proposed by Ojala et al. [6]. It considers $N_8(p)$ to calculate the binary pattern which results into an 8-bit pattern. The process of getting the 8-bit pattern is shown in Fig. 1. The value of $N_8(p)$ is compared with the value of 'p' and based on $\leq$ or $>$ it results into 0 or 1 respectively. Once after getting the pattern, for each of the pixel in the image, will be converted into equivalent decimal number which is the final result of the processed pixel. These converted decimal number's histogram is considered as the feature vector of the image with length of 256. Note that, when converting the binary pattern to decimal numbers, out of 8 positions the place value can start at any of the eight bits, but the same place values needs to be considered for the entire image.

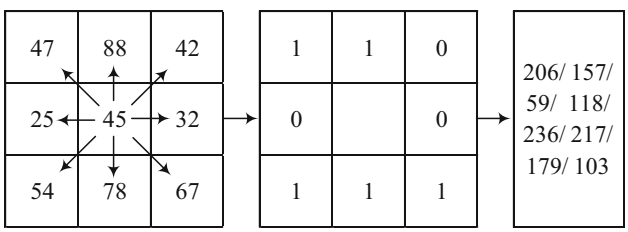


Fig. 1 LBP calculation (**a**) Original Image (**b**) Resultant binary pattern for (**a**). **c** Decimal equivalent for (**b**)

1.1.2 Uniform local binary patterns

To reduce the feature vector length resulted by LBP and also because most of the features in many of the image datasets are covered with 59 bins as given in [6], Uniform Local Binary Patterns (ULBP) is used. In ULBP, the number of bins as well as the length of the feature vector, is reduced from 256 to 59 based on the number of transitions in the binary patterns. The decimal numbers are given in Fig. 2, have the number of transitions ≤ 2. All the remaining decimal numbers in the range of 0 to 255, the number of transitions is 4, 6 or 8. So these decimal numbers with the number of transitions ≤ 2 are considered as same as that values resulting in 58 bins since we have 58 such decimal numbers. The remaining numbers are considered as the 59th bin. From Fig. 1, the LBP (45) is 206. If we apply ULBP on this, as '206' is not in the list of numbers in Fig. 2, 206 is considered to be in the 59th bin of ULBP.

1.1.3 Color histogram

One of the most prime and basic color features is the color histogram, which mainly provides the color frequency information in a particular color model [2]. In any color model, firstly it is decomposed into a different component. Then for each color component, a separate histogram is obtained and then the resultant histogram are concatenated to obtain the feature vector. To reduce the length of the feature vector, before obtaining the histogram, the pixels can be quantized into different bins for each color component.

1.1.4 Color correlogram and color autocorrelogram

Color Correlogram is proposed by Haung et al. [3]. This feature not only gives the information about the frequency of each color pixel but also focuses on the co-occurrence of pixel pairs on a specified distance k. To measure the distance, D_8 distance is used, which is defined in Eq. (1):

$$D_8(p,\ q) = \text{Max}\left|(p_x - q_x), (p_y - q_y)\right| \tag{1}$$

where p and q are two-pixel values of an image having a location (p_x, p_y) and (q_x, q_y).

0	255	1	2	3	4	6	7	8	12	14	15	16	24	28	30
31	32	48	56	60	62	63	64	96	112	120	124	126	127	128	129
131	135	143	159	191	192	193	195	199	207	223	224	225	227	231	239
240	241	243	247	248	249	251	252	253	254	All other values in range of 0-255					

Fig. 2 ULBP-59 bins. The decimal numbers with number of transition with ≤ 2 are shown

Color Correlogram can be expressed by a matrix C^k of size $N \times N$, in which each cell value, C^k (i, j), the joint probability of occurrence of a pixel pair (i, j), separated by a specified distance k. Color Correlogram can be calculated using Eq. (2).

$$C^k(i,\ j)\ =\ \frac{1}{N_i \times 8k}\sum_{m=1}^{M}\sum_{n=1}^{N} 1 - \left[\frac{1}{2}\left(\frac{|I(m,\ n)-i|}{L_{\max}}\right) + \frac{1}{2}\left(\frac{|I(m+\Delta x,\ n+\Delta y)-j|}{L_{\max}}\right)\right] \tag{2}$$

$$\forall \Delta x,\ \Delta y \in \{(0,\ 1,\ 2, \ldots\ldots k) and (\max(\Delta x,\ \Delta y) = k)\}$$

and $\forall i, j \in \{0,\ 1,\ 2, \ldots L_{\max}\}$.where, N_i is the histogram of color 'i', which is given by Eq. (3).

$$N_i = \sum_{m=1}^{M}\sum_{n=1}^{N} 1 - \left[\frac{|I(m,\ n)-i|}{L_{\max}}\right] \tag{3}$$

Color Correlogram gives the joint probability of occurrence of all possible pixel levels, which results in a feature vector of length $N \times N$. To reduce feature vector length *Color Autocorrelogram* was proposed [3] and also discussed in [4]. This color feature concentrates only on the co-occurrence of the same color, results in a feature vector α^k of length N, which is the diagonal values of color correlogram matrix C^k. Color autocorrelogram can be calculated using Eq. (4). Figure 3 shows the original image and its autocorrelogram.

$$\alpha^k(I) = C(i,j), \forall (i,j,l) \in \{0, 1, 2, \ldots, L_{\max}\} \text{ and } \quad i = j = l \tag{4}$$

1	1	2	2
3	2	1	3
3	4	3	4
1	1	1	4

(a)

0.167	0.167	0.125	0.083

(b)

Fig. 3 Color autocorrelogram calculation. (**a**) Original image. **b** Color autocorrelogram of (**a**)

1.1.5 Inter-channel voting

The two color feature extraction methods discussed above concentrate only on each color channel to extract color feature descriptor. Suresh et al. [20] proposed a new color feature named inter-channel voting among the three components of HSI image. This method explores the interrelationship among three components Hue (H), Saturation (S) and Intensity (I) of a color image. To perform inter-channel voting between H and I channel, both channels are quantized into bins and then added with the quantized I value to respective H bin and vice versa. This process is shown in Fig. 4. Due to non-commutative characteristics of inter-channel voting, feature vector produced by inter-channel voting between H and I is different from feature vector generated by inter-channel voting between I and H. The process of inter-channel voting is applied on hue and saturation, hue and intensity, and intensity and saturation component of the image separately which creates a total of 6 different feature vectors. To create the final feature vector all six feature vectors are concatenated. Feature vector creation for I and H and H and I are shown in Eqs. (5–8).

$$Range_I = \frac{\max(I(i,\ j)) - \min(I(i,\ j))}{I_{\text{level}}}; \forall i, j \in I \tag{5}$$

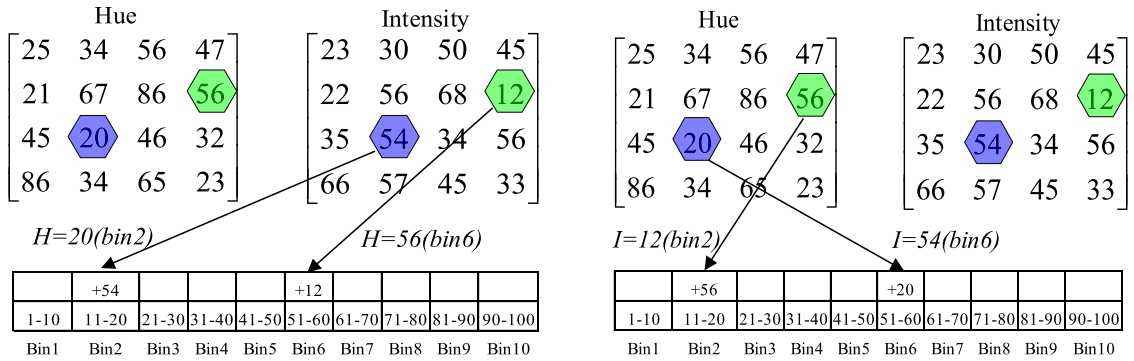


Fig. 4 Inter-channel voting. The first 1-D array is the result of the process between 'Hue & Intensity' while the second one is the result of the process between 'Intensity & Hue' components

$$I_{new}(i,j) = \begin{cases} I_{level} - 1, & I(i,j) = \max(I(i,j)) \\ \left\lfloor \dfrac{I(i,j)}{Range_I} \right\rfloor, & else \end{cases} \quad (6)$$

where I_{level} is the number of quantization levels for Intensity.

$$Bin_{IH}(I_{new}(i,j)) = Bin_{IH}(I_{new}(i,j)) + H(i,j), \forall i,j \in I \quad (7)$$

$$Bin_{IS}(I_{new}(i,j)) = Bin_{IS}(I_{new}(i,j)) + S(i,j), \forall i,j \in I \quad (8)$$

1.2 Big image data processing

The digital devices are evolved from with very less storage capacity, less processing capacity and with bigger in size to with more storage, more processing power and small in size. Because of this evolution, as of today, these digital devices generating lot of diverse and complex data. Because of this, the existing computing devices are suitable for storing and processing such data. The astronomy and genomics are the first to experience such data explosion in the 2000s coined the term *BigData* [21]. The word *Big* cannot be quantified, it is a moving target. What is considered '*Big*' today will not be so years ahead. The data in 'Big Data' can have three properties: Volume, Variety and Velocity. Note that it does not mean that it must have all the three characteristics. As the years passing the number of Vs also increased to 4Vs in 2012, 7Vs in 2013 [22] and to 10Vs by 2014. [23].

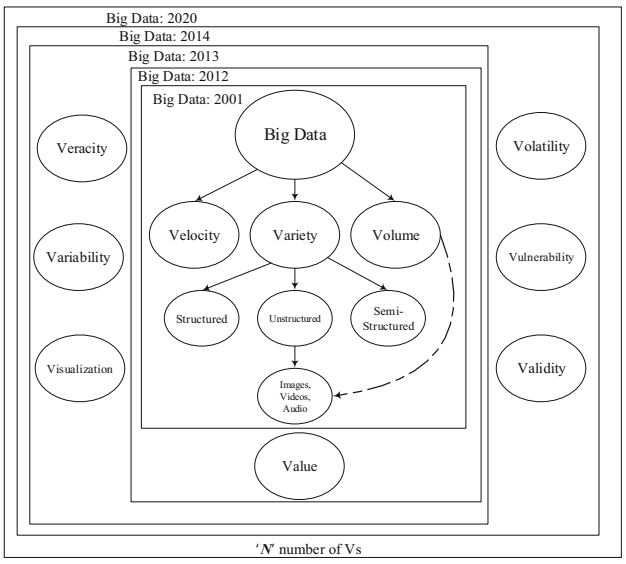


Fig. 5 Evolving of Big Image/Vodeo Data Processing. In the year 2001 there were 3Vs in Big Data Processing. In 2012 one more V was added. In 2013 and 2014 three more Vs were added which results total 10 Vs. As of 2020 more Vs are deliberated

Out of the entire world's data, 80% is unstructured data essentially containing photos and videos [24]. In developed countries like UK, billions of videos per year are recorded by millions of CCTV cameras [25]. As these billions of videos need to be stored and processed, demand for storing and searching has increased substantially. Based on the kind of data that 'Big Data' technologies handle, image and videos comes under unstructured data and the relationship of Image Processing and video processing is shown in Fig. 5. This can be called as *Big Image/Video Data Processing*. With this, many technological challenges including compression, storage, transmission, analysis and recognition which cannot do address by existing technologies can now be addressed [26–29].

BigData has its application in several important areas such as Manufacturing, Healthcare, Fraud Detection, Transportation Service, Communication, Banking Sector, Media and Insurance Service. In the healthcare system, diseases like Genomics, Cancer, Chronic Obstructive Pulmonary Disease (COPD) and Tumor can be predicted, diagnosed and monitored [30, 31]. In the development of smart cities, the Transportation services plays an important role by controlling traffic, planning route, managing revenue and providing travel guidance to the urban residents [32]. In [33], a method for automatically calculating traffic volume and vehicle speed by pattern analysis using pixel

data extracted from CCTV Video image is proposed. In the field of e-Commerce, analyzing customer feedback, shopping patterns, and identifying market areas guide to a superior decision-making process. With the help of smart devices with GPS functionality and social media, analysis of behavior of the customer facilitates a reduction in insurance and banking sector risks [34].

1.2.1 Hadoop

One of Apache more successful project is Hadoop [35]. It is used to handle Big Data state of affairs for storage and processing in distributed environment. It is an open-source Java based framework. It used MapReduce paradigm for programming. In this environment, a large number of computers (nodes) can be grouped together to form a cluster. With this cluster the more storage and more processing power will be obtained when compared with a single node. But to work with this, it need not be always a cluster of computers, even with pseudo distributed mode of Hadoop i.e. with only one computer also it can be used. Hadoop offers flat scalability curve which is the major advantage of this when compared with Message Passing Interface (MPI). Hadoop is responsible for breaking up the input data into chunks, forwarding the chunk to each of the node, executing the code on each of the chunk, examining if the code has executed, then forwarding results, if any, either to the proceeding processing stages (known as Job) or to the final location of the output, carrying out the sorting action between the map and the reduce stages and forwarding each chunk of the sorted data to the right node, and writing debugging information on each job's progress, among other things [35, 36]. Some of the other noteworthy implementations of MapReduce are Infinispan, Disco Project, CouchDB, MongoDB and Risk. The two components of Hadoop, one for storage and another for processing are Hadoop Distributed File System (HDFS) and MapReduce model.

1.2.2 Hadoop distributed file system

It is one of the file system used for distributed environments. When a cluster is formed with multiple nodes, all the nodes contribute some amount of memory to HDFS. HDFS is the backbone of the Hadoop system which stores the data by replication and makes different copies of data on to a different rack for the purpose of fault tolerance. The replication factor can be any value but conventionally 3 is used. For storage purpose, it maintains three Java Virtual Machine Process Status Tool (jps): Name Node, Secondary Name Node and Data Node. The entire cluster will have only one Name Node and one Secondary Name Node but can have *n* Data Nodes. For processing it maintains two jps: Resource Manager and Node Manager, here also the only one Resource Manager and *n* Node Managers in the cluster. In this paper, HDFS is used to store the large number of images where it is not possible to store all these images in a single system. To reduce the processing time on these large number of images, MapReduce model of programming is used.

1.2.3 MapReduce

MapReduce is a programming model for processing huge amount of data by taking the data from either HDFS [37, 38] or also from Local File System (LFS). It consists of two phases: *map* phase and *reduce* phase. The map phase uses a function known as *mapper*, which takes the input data in the form of a series of <key, value> pairs and outputs also in the form of <key, value> pair. The intermediate <key, value> pairs will be combined by *reducer*, which is also a function, this phase is known as reduce phase.

1.3 Map reduce paradigm for image retrieval

When dealing with a large number of images, the MapReduce paradigm is one of the best solutions to get the results in less time than that on doing it on a single system. This is a programming paradigm in which the execution takes place where the data resides. The execution takes place in three stages: map, Shuffle and Sort, and Reduce stages. The Map stage takes in the input in <key, value> pair and produces the output also as <key, value> pair. Then the Shuffle and Sort stage will sort this based on the 'key'. Then the reducer will consolidate the work for each of the key and produces the final output. For storing, the data in intermediate steps Distributed File System can be used. This data can be in any form: Text, Images, Videos, Log Data, etc.

Sarmad Istephan et al. [39] proposed a method to retrieve an image from unstructured medical image big data with a case study on epilepsy. They have used two types of criteria to validate the feasibility of the proposed framework: accuracy and ability. The accuracy is tested by executing the query on data that contain both structured and unstructured data. To test the ability of the framework, the results are compared by executing the query on different sized Hadoop clusters. The same kind of ability is tested in [40] also. One novel CBIR framework was proposed by Lan Zhang et al. [41], known as PIC, where cloud computing is used for searching an image from a large image dataset while securing the privacy of input data Here to deal with massive images, they have designed a system suitable for distributed and parallel computation to expedite the search process. Le Dong [29] proposed an effective

processing framework named Image Cloud Processing (ICP) to deal with the data explosion in the image processing field. The ICP framework consists of two mechanisms: Static ICP (SICP) and Dynamic ICP (DICP), where SCIP is designed to cooperate with the Map-Reduce paradigm and DICP implemented through a parallel processing procedure working with the traditional processing mechanism of the distributed system. To validate the ICP framework, they have used the ImageNet dataset.

1.4 Performance measures for CBIR

1.4.1 Average precision rate (APR) and average recall rate (ARR)

Precision is defined as a ratio between the number of total relevant images retrieved and the number of total images retrieved for a given query. Recall is defined as the ratio between the number of total relevant images retrieved and the number of total images having the same class as a query image. Average precision for different step sizes m_1,m_2,…m_k is known as APR. Similarly, average recall for different step sizes is known as ARR.

1.4.2 F-Measure

It is represented by a single value to reflect the relationship between precision and recall. It is obtained by assigning equal weight to both precision and recall in the harmonic mean calculation as given in Eq. (9).

$$\text{F - Measure}(n) = \frac{(2 \times APR \times ARR)}{(APR + ARR)} \tag{9}$$

1.4.3 Average normalized modified retrieval rank (ANMRR)

It is used to measure the retrieval accuracy. To calculate ANMRR for each image we consider only those images whose rank is less than 2 × (number of images in the class). If an image's rank is less than 2 × (number of images in the class) then score of that image is rank of the image, else it is a predefined fixed number. Now the average score is calculated and then normalized score.

1.4.4 Total minimum retrieval epoch (TMRE)

It is used to measure the minimum number of images to be traversed to retrieve all the relevant images.

2 Methodology

The process of feature extraction and obtaining different performance measures (APR, ARR, F-Measure, TMRE, ANMRR) is done by using MapReduce paradigm. The different texture features used for CBIR are: LBP and ULBP. In addition to textures features, two types of color features Color Histogram and Color Autocorrelogram are used. Finally, a fused features i.e. Interchannel voting with DS_GLCM [20] are used for CBIR. All these five method are explained in Introduction section. In this section, the detailed description of MapReduce paradigm used to retrieve the queried images from a given image dataset is given. Then each MapReduce job is explained in detail. In the proposed method, a total of 8 MapReduce Jobs: Job0 to Job7 are used for the image retrieval process and the block diagram is shown in Fig. 6. Detailed description of all the eight jobs is also given.

2.1 Job0 functionality

Job0 Functionality:

Job0 : Creating Sequence Files	*Mapper*		*Shuffle and Sort*	*Reducer*	
	Input	*Output*	All file names are sorted based on image file names	*Input*	*Output*
	<Folder,Image Files>	<FileName, Pixels value of the image>		<FileName, Pixels value of the image>	<Seq.File-1> <Seq.File-2> <Seq.File-w>

The given image dataset can be stored in the LFS or in HDFS. The Job0 functionality is shown in Fig. 7. The mapper will take the images with any extension (jpg/png/tif…) as input and gives key, value pair: <FileName, Pixel values of the image> as the output. Note that, if the given image is a color image, all the three channels i.e. Red, Green and Blue pixels are stored as part of <key, value> pair. All these <FileName, Pixel values of the image> are the input to shuffle and sort, where all these will be sorted based on key. Now, this sorted <FileName, pixel values of the image> are the input to reducer, which will convert it into sequence files. The number of <key, value> pairs in each sequence file is depending on the size of the sequence file supported by that software.

2.2 Job1 functionality

Job1 Functionality:

Job1 : Feature Extraction	*Mapper*		*Shuffle and Sort*	*Reducer*	
	Input	*Output*	All file names are sorted based on image file names	*Input*	*Output*
	<FileName, Pixels value of the image>	<FileName, Feature Vector>		<FileName, Feature Vector>	<FileName, Feature Vector>

The Mapper takes the unbundled data as the input i.e. <FileName, Pixels value of the image> . If this is for a color image, the mapper will convert it into a gray image and then calculates the Feature Vector (LBP, ULBP). But for the other three methods color images are used as is. The output of the Mapper is <FileName, Feature Vector> . All these < FileName, Feature Vector> are the input to shuffle and sort, where all these will be sorted based on key. Now, this sorted < FileName, Feature Vector> are the input to reducer, which will convert it into sequence files. The Job1 functionality is shown in Fig. 8.

2.3 Job2 functionality

Job2 Functionality:

Job2 : Formating Rank Matrix	*Mapper*		*Shuffle and Sort*	*Reducer*	
	Input	*Output*	All file names are sorted based on image file names	*Input*	*Output*
	<FileName, Feature Vector>	<FileName, ***n*** distances w.r.t. to 1 to *n* images>		<FileName, ***n*** distances w.r.t. to 1 to *n* images>	<FileName, ***n*** image numbers based on rank>

The mapper considers each image and calculates distance from the image from the 'key' part of the pair with all *n* images in the dataset, results into <FileName, *n* distances w.r.t. to 1 to *n* images> . It will do the same process for all the 'key' part i.e. for all the images. i.e. <image 1, (0, 15, 45, 6,…)> , <image 2, (15, 0, 4, 61,…)> . In, <image 1, (0, 15, 45, 6,…)> , '0' represents the distance between image1 to image1, '15' represents the distance between image 1 and image 2 and '45' is the distance between image 1 to image 3 and so on. You need to observe that, the distance from image 1 to image 2 is same as the distance between image 2 to image 1. As in previous Jobs, here also the shuffle and sort will be sorting the data based on image number. The reducer will be giving the ranks to the image numbers based on the distances given as part of 'value' part. Based on these ranks, the image numbers will be written i.e. <image1, (1, 101, 205, 900,…)> , Here, the number in value part (1, 101, 205, 900,…) represents the image numbers close to image 1 based on the distance. i.e. '1' in value part represents, image 1(value part) is 1st closest w.r.t. to image 1 (key part) 101 represents, images 101 is 2nd closest w.r.t. to image 1, image 205 is the 3rd closest w.r.t. image 1, …is ranked 102 w.r.t to image 1, image 3 is ranked. One more example: <image2, (108, 2, 43, 66,…)> . To sum up, the output <key, value> pairs of the Job 2 are the columns in our rank matrix representation given in Fig. 9. The Job2 functionality is shown in Fig. 10.

2.4 Job3 functionality

Job3 Functionality:

Job3 : Summing up counts based on step size and group size	*Mapper*		*Shuffle and Sort*	*Reducer*	
	Input	*Output*	Sorted based on Group	*Input*	*Output*
	<FileName, ***n*** image numbers based on rank>	$<m_1$, count> $<m_2$, count>		$<m_1$, count> $<m_2$, count>	$<m_1$, Total> $<m_2$, Total>

The mapper reads the data as <FileName, n image numbers based on the rank> , then for each key it checks and update the count whether that image number which is part of the 'value' part is of that group, if so accordingly it update the count. The same process is followed for m_1, m_2, m_3……images. $\forall\ m_i \leq k$, where k is the number of images in that group. The output of mapper is <m_1, (6, 7, 8, 2,…a total of n numbers)> , here 6 represents for image 1, out of top m_1, 6 are of the same group. 7 means for image 2, out of m_1, 7 are of the same group… Like this for every m_2, m_3,… are also given as the output. The shuffle and sort will sort the data based on key i.e. m_1, m_2, m_3,….. The reducer will add all the numbers in the value part and gives the output as: <m_1, number of images of that particular group> Example: <m_1, 6918> means out of $m_1 \times n$ number of images, 6918 images of the same group. <m_2, 12,353> means out of $m_2 \times n$ number of images, 12,353 are of same group. Like this it will be calculated for all m_is. The entire process is given in Fig. 11.

The functionality of this job is to calculate TMRE. The mapper takes the rank matrix as the input, the same input which is used as the input for Job3. The mapper need to go through the images until it completely finds all the images of the image given in the 'key'. The output of the mapper is <commonkey, count> for 1st column of the rank matrix, <commonkey, count> for the 2nd column of the rank matrix, etc., for all the images in the data set, where 'count' represents upto which rank we have to visit to find out all the images of this group. The reducer will take this as input and calculates the parameter TMRE. This process is given in Fig. 13.

2.5 Job4 functionality

Job4 Functionality:

Job4 :Calculating APR, ARR and F-Measure	*Mapper*		*Shuffle and Sort*	*Reducer*	
	Input	*Output*	Sorted based on: APR, ARR and F_msr	*Input*	*Output*
	<m_1, Total> <m_2, Total>	<APR, (for m1, for m2, ..)> <ARR, (for m1, for m2, ..)> <F-msr, (for m1, for m2, ..)>		<APR, (for m1, for m2, ..)> <ARR, (for m1, for m2, ..)> <F-msr, (for m1, for m2, ..)>	<APR, Result> <ARR, Result> <F-msr, Result>

The input to the mapper is a table of counts of matches for group for each step size. Now these counts will be converted into percentages by the mapper which results into the output as <APR, (for $m1$, for $m2$, …)> , < ARR, (for $m1$, for $m2$, …)> and <F_Meaasure, (for m_1, for m_2, …)> . These three will be sorted by shuffle and sort phase by using the keys: APR, ARR and F-Measure. The reducer discards all except APR of top $m1$ matches, ARR for top k and F-Measure also for top k. This is the final and same result obtained using non MapReduce paradigm. The entire functionality of Job4 is shown in Fig. 12.

2.6 Job5 functionality

Job5 Functionality:

Job5 : TMRE Calculation from Rank Matrix	*Mapper*		*Shuffle and Sort*	*Reducer*	
	Input	*Output*	No change with shuffle and sort	*Input*	*Output*
	<FileName, ***n*** image numbers based on rank>	<SingleKey, count> ***n*** times		<SingleKey, count> ***n*** times	<TMRE, Result>

2.7 Job6 functionality

Job6 Functionality:

Job6 : Formating Image Based Matrix	*Mapper*		*Shuffle and Sort*	*Reducer*	
	Input	*Output*	All file names are sorted based on image file names	*Input*	*Output*
	<FileName, Feature Vector>	<FileName, ***n*** distances w.r.t. to 1 to *n* images>		<FileName, ***n*** distances w.r.t. to 1 to *n* images>	<FileName, ***n*** ***ranks*** based on image vs image>

The mapper and shuffle and sort functionality of this job is exactly like that of Job2's mapper and shuffle and sort. The output of the reducer is a matrix based on images as shown in Fig. 14, which is called as Image based matrix. The out of the reducer will in the in form of <image 1, (1, 101, 205, 900,…)>, here, the number in *value* part (1, 101, 205, 900,…) represents the ranks between image 1 vs image 1, image 1 vs image 2, image 1 vs image 3,…. In this Image based matrix it has to be observed that the principal diagonal elements are all 1 s, because the rank between image *i* vs image *i* is always 1. The entire process of Job 6 is shown in Fig. 15.

2.8 Job7 functionality

Job7 Functionality:

Job7 : ANMRR Calculation	*Mapper*		*Shuffle and Sort*	*Reducer*	
	Input	*Output*	All file names are sorted based on image file names	*Input*	*Output*
	<FileName, ***n*** ***ranks*** based on image vs image>	<FileName, NMRR Value>		<FileName, NMRR Value>	<ANMRR, Result>

In this job7, the mapper takes the input which is image based matrix with the ranks. The output of the mapper is NMRR values for each images i.e. FileName 1, NMRR>, <FileName 2, NMRR>, … and sorted version of this is given as the input to Reducer. The reducer will calculate the ANMRR value and displays it. The process is shown in Fig. 16.

The entire process that has been explained can be implemented by using different options. The options are based on storage mechanism and processing mechanism. The storage can be LFS or HDFS. The processing can be any of Non-MR Model *or* Matlab's MR Model *or* Hadoop's MR Model. With these options, the seven different modes of implementation are given in Table 1. All the seven methods gives the same CBIR results for all the five parameters: APR, ARR, F-Measure, TMRE and ANMRR, but the only difference is the time to complete the process.

3 Results and discussions

The experimentation is done on two of the image dataset, one is Corel-1k, which is a natural image dataset and other one is texture image dataset: VisTex, The same datasets are used by Netalkar Rohan Kishor et al. [42] for CBIR by extracting the features in frequency domain specifically in Discrete Cosine Transorm. The five methods of image retrieval explained in Introduction section are considered for retrieving the images. By using, Mode-6 from Table 1, the results are given here. The experiment is carried out by using 1, 2 and 4 parallel workers in MATLAB R2020b Version. The detailed results are explained here for each of the dataset.

3.1 Dataset-1 (Corel1K)

This dataset [43], consists of a total of 1000 images with 10 categories where each category consists of 100 images. The different categories are Africans, Beaches… Food. The size of each image in this image dataset is 384 × 256 or 256 × 384. Three images from each group, a total of 30 images of this dataset are shown in Fig. 17. The performance measures for all the five CBIR methods are given in Table 2. Figures 18, 19, 20, 21, 22 shows the time in seconds for each of the seven jobs and for the total time for all the five methods. Table 3 shows the total time taken for each of the five methods and also the total time taken for all the five methods of image retrieval, by considering 1, 2 and 4 workers in parallel execution. Table 3 also shows the percentage of time saved for different number of workers.

From these graphs in Figs. 18, 19, 20, 21, 22, it is clearly evident that, the time to complete with 4 workers is very

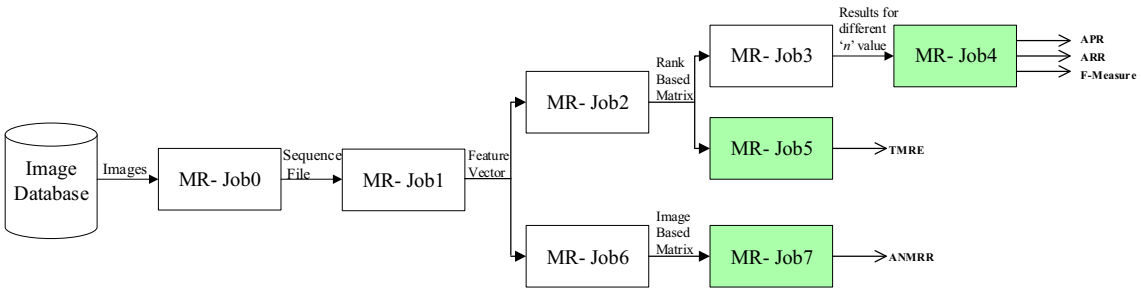


Fig. 6 Block diagram of the proposed system using MR Paradigm. For each job input and output are shown and performance measures values are calculated

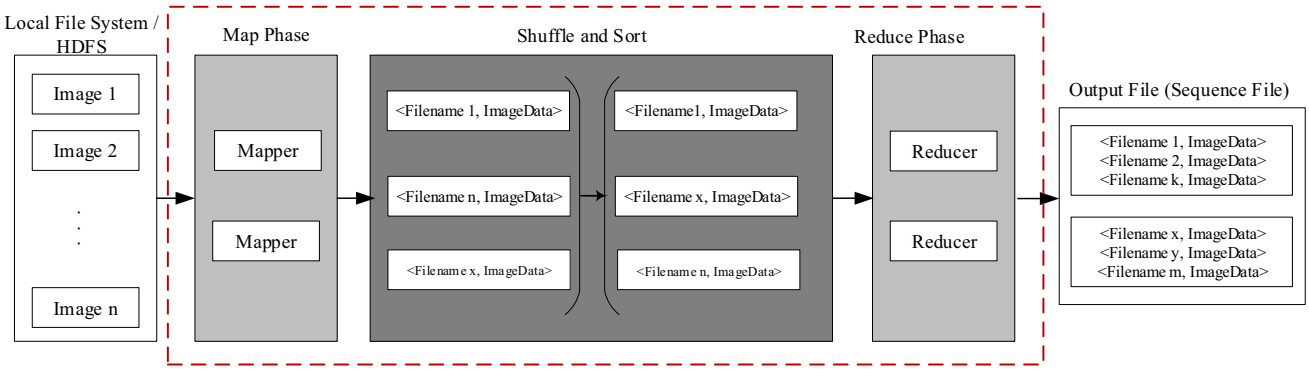


Fig. 7 Outline of MapReduce Job0. It is used to convert the given images into sequence files

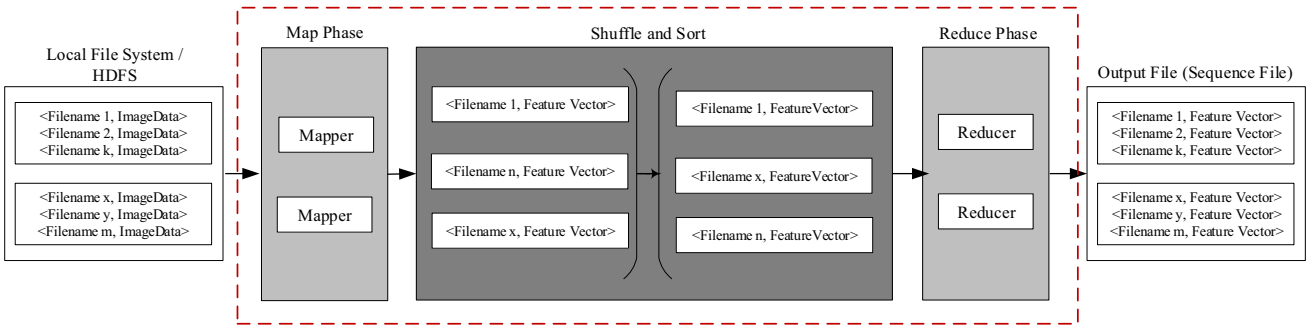


Fig. 8 Outline of MapReduce Job1. It is used for calculating the Feature vectors for all the images

		Query Iamges Considered from the Image Dataset									
		Class-1				…	…	…	**Class-10**		
		Image1	**Image2**	…	**Image100**	…	…	…	…	**Image999**	**Image1000**
Ranks given to Images in the Dataset	1	Image1	Image2		Image100						Image1000
	2	Image301									
	3	Image108									
	1000										

Fig. 9 Rank matrix representation for 1000 images dataset. (*i*, *j*)*th* position of this matrix contains *ith* similar image of image *j*

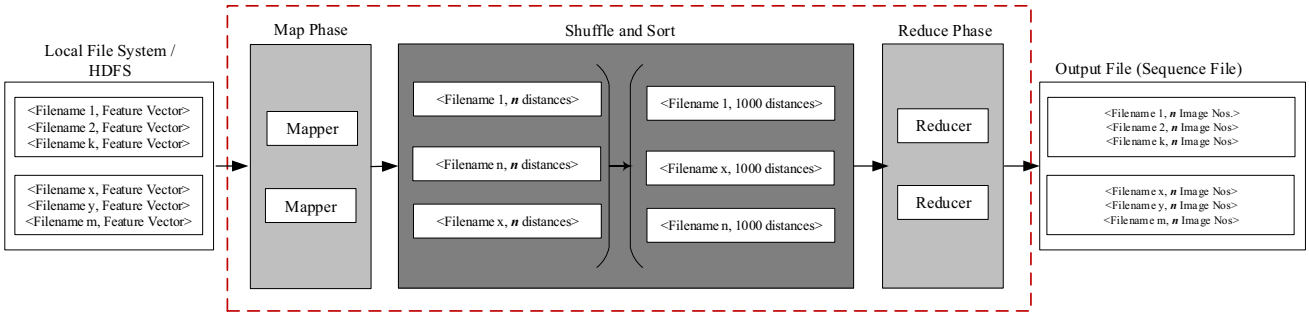


Fig. 10 Outline of MapReduce Job2. The purpose of job2 is to construct rank matrix by considering the distance between feature vector of query image and of images in the dataset

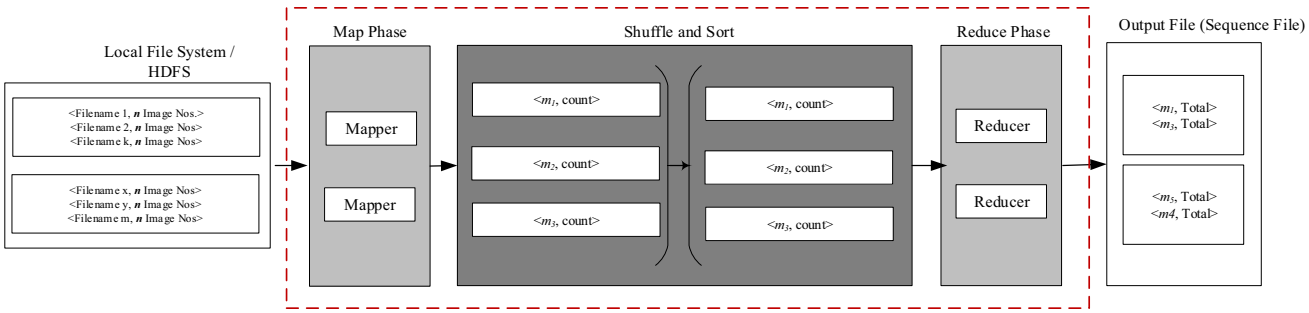


Fig.11 Outline of MapReduce Job3. It is used for obtaining the counts for the number of correct retrieval for each of the query images of the image dataset

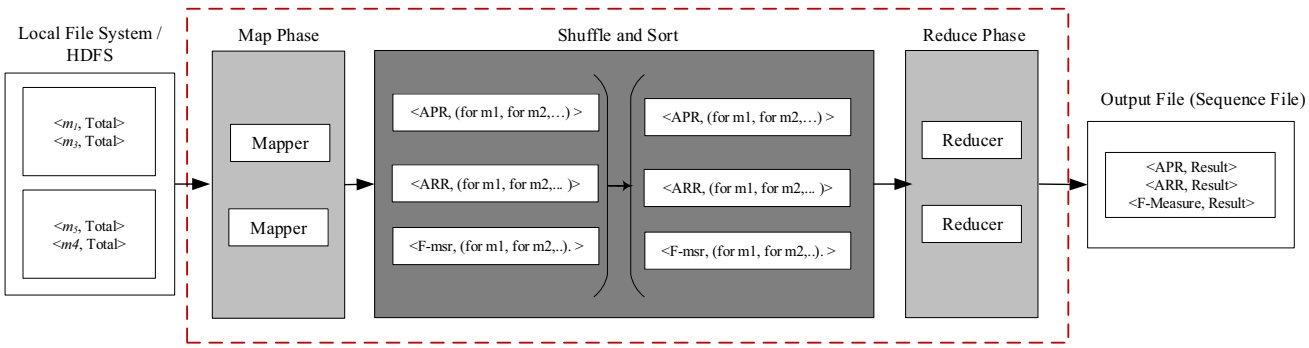


Fig. 12 Outline of MapReduce Job4. It is used for calculating APR, ARR and F-Measure

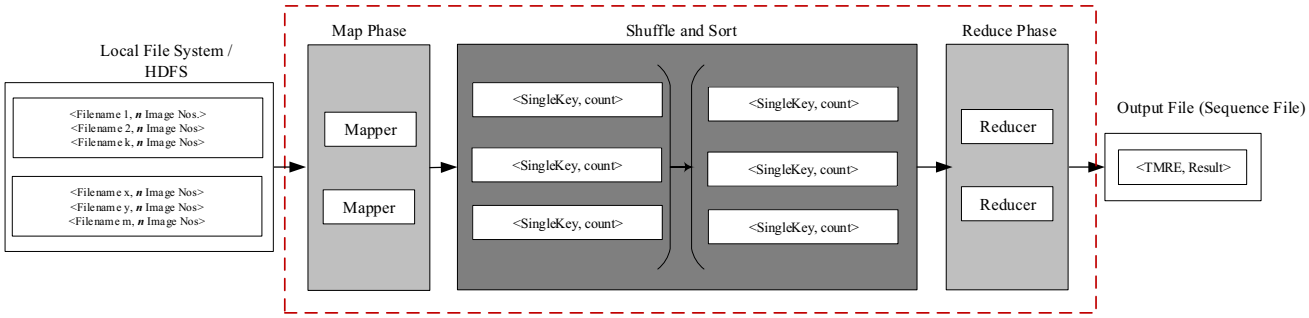


Fig. 13 Outline of MapReduce Job5. Output of this job is TMRE value

		Query Iamges Considered from the Image Dataset									
		Class-1				…	…	…	**Class-10**		
		Image1	**Image2**	…	**Image100**	…	…	…	…	**Image999**	**Image1000**
Image Numbers in the Dataset	1	**Rank 1**									
	2	Rank 102	**Rank 1**								
	3	Rank 5		**Rank 1**							
	1000										**Rank 1**

Fig. 14 Image based matrix representation for 1000 images dataset. $(i, j)th$ position of this matrix contains similarity of image i with respect to image j

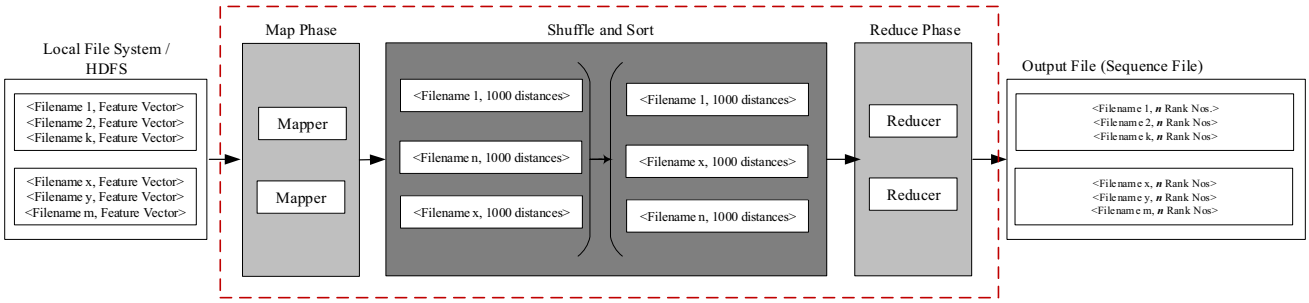


Fig. 15 Outline of MapReduce Job6. It is used for obtaining the Image Based Matrix from by considering the distance between feature vectors of query image and of images in the dataset

less when compared to single system and as well as with 2 workers setup. From Table 3, the time saved by using 4 workers instead of 2 worker is around 41% for all the five CBIR methods and similarly the time saved by using 4 workers instead of 1 worker is around 68%.

3.2 Dataset-2 (VisTex)

This image dataset is the first texture image dataset considered VisTex texture dataset [44] consists of a total of 484 images. Out of these 484 texture images, 40 are considered for experimentation. The actual image dimension is 512 × 512. Each image of these 40 is made into 16 nonoverlapping sub-images where each sub-image is of dimension 128 × 128, which results in a total of 640 texture image datasets. From these 640, images 1, 17, 33, 49 …625 which are the 1st sub-image of each of 40 actual texture images, are shown in Fig. 23. The performance measures for all the five CBIR methods are given in Table 4. Figures 24, 25, 26, 27, 28 shows the time in seconds for each of the seven jobs and for the total time for all the five methods. Table 5 shows the total time taken for each of the five methods and also the total time taken for all the five methods of image retrieval, by considering 1, 2 and 4 workers in parallel execution. Table 5 also shows the percentage of time saved for different number of workers.

From these graphs in Figs. 24, 25, 26, 27, 28, it is clearly evident that, the time to complete with 4 workers is very less when compared to single system and as well as with 2 workers setup. From Table 5, the time saved by using 4 workers instead of 2 worker is around 42% for all the five CBIR methods and similarly the time saved by using 4 workers instead of 1 worker is around 68%.

4 Conclusions

The results clearly shows that the MapReduce paradigm is working as expected. As the number of workers are involved are more in number, the time for computing the whole process is reduced accordingly. For all the five image retrieval methods used the final results of performance measures: Average Precision Rate, Average Recall Rate, F-Measure, Average Normalized Modified Retrieval Rank and Total Minimum Retrieval Epoch are exactly same as in single computer execution. Even irrespective of the method used for image retrieval, the times for all the five methods are relatively same. For completing all the

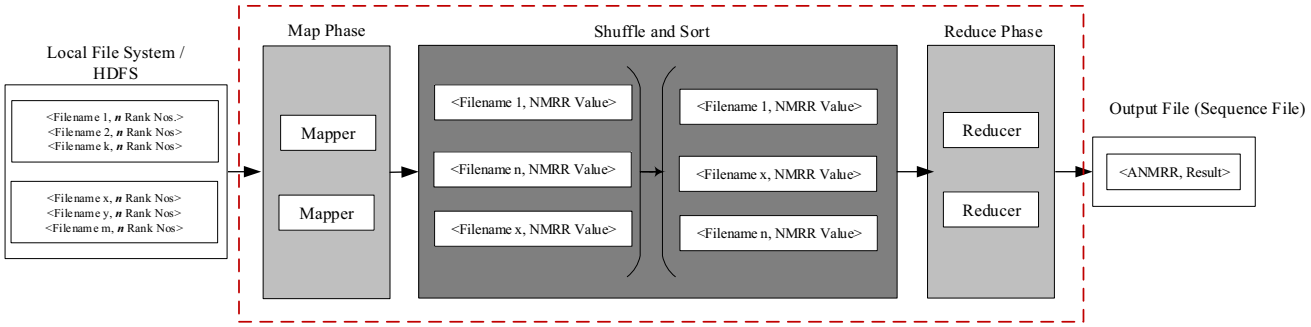


Fig. 16 Outline of MapReduce Job7: it is used for calculating ANMRR value

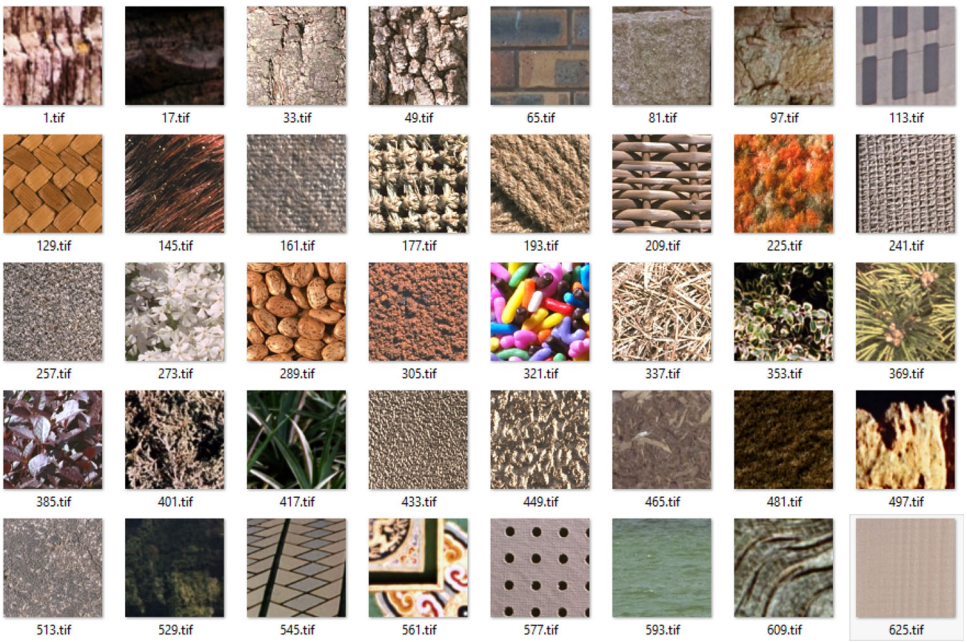


Fig. 17 Corel-1K samples. Three images from each of the 10 categories are displayed

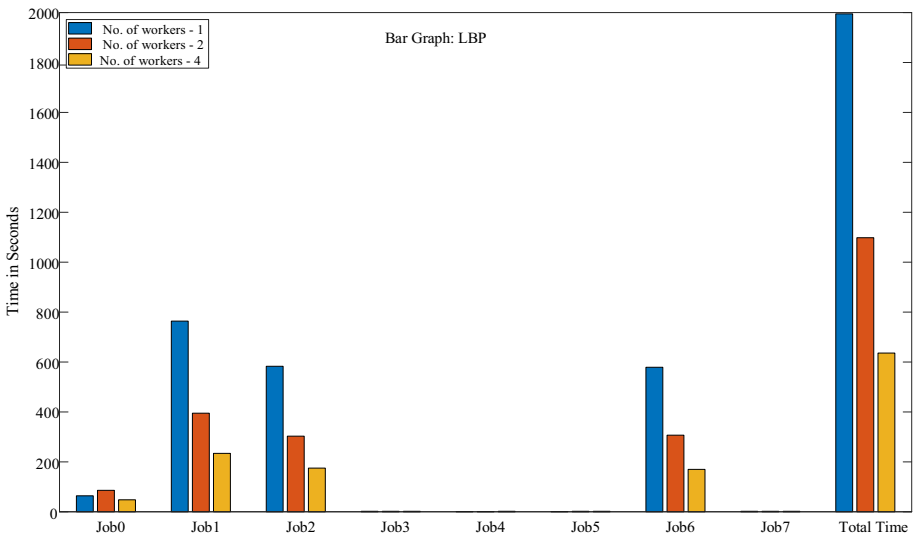


Fig. 18 Time graph for LBP for Corel1K dataset. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also

Table 1 Different MapReduce Paradigms for CBIR

Mode-1	Storage: LFS	Processing: Non-MR
Mode-2	Storage: LFS	Processing: Matlab's MR
Mode-3	Storage: HDFS (Hadoop PseudoMode)	Processing: Matlab's MR
Mode-4	Storage: HDFS (Hadoop PseudoMode)	Processing: Hadoop Pseudomode MR
Mode-5	Storage: HDFS (Hadoop FullMode)	Processing: Matlab's MR
Mode-6	Storage: LFS	Processing: Matlab's MR with n workers
Mode-7	Storage: HDFS (Hadoop FullMode)	Processing: Matlab's MR with n workers

Seven different modes are shown

Table 2 Performance measures for different CBIR methods on Corel-1K dataset

Corel-1K	APR	ARR	F-Measure	ANMRR	TMRE
ColorHist_RGB [1998]	64.28	37.71	29.275	0.5334	8.08
Color_Autocorrelogram [2012]	52.54	28.32	22.423	0.6278	9.46
LBP [2002]	69.18	38.69	31.228	0.5208	8.04
ULBP [2002]	69.26	44.42	33.929	0.4574	7.50
IC_HSI + DS_GLCM [2019]	80.15	50.94	39.502	0.3880	7.35

The same results will be obtained by any of the seven modes given in Table 1

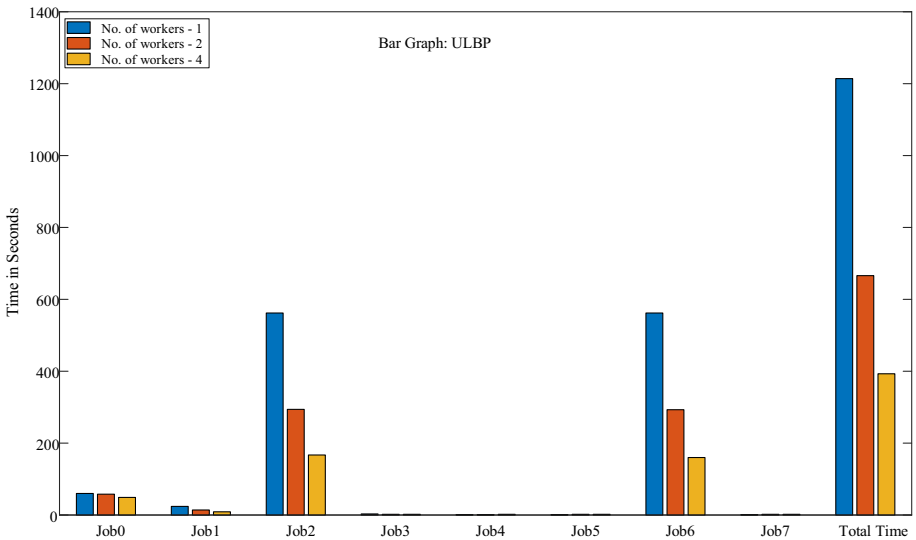


Fig. 19 Time graph for ULBP for Corel1K dataset. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also

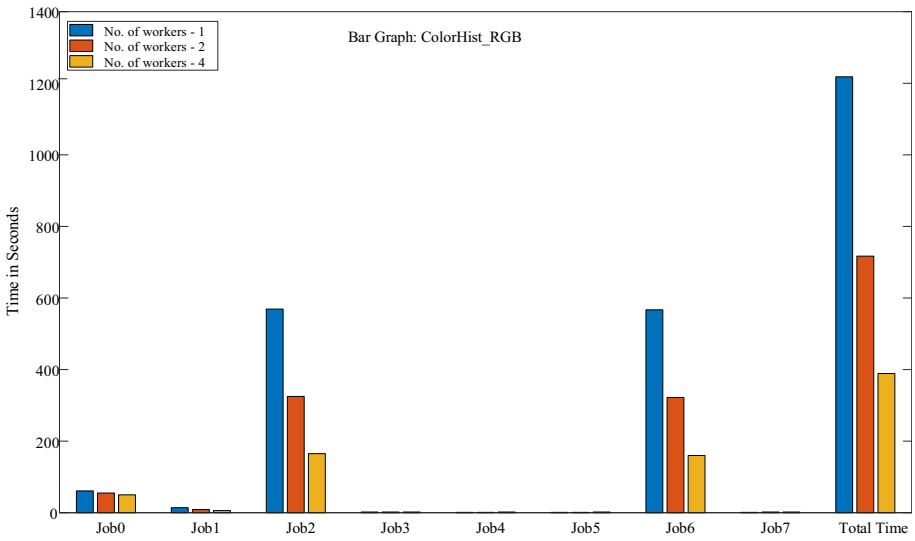


Fig. 20 Time graph for ColorHist_RGB for Corel1K. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also

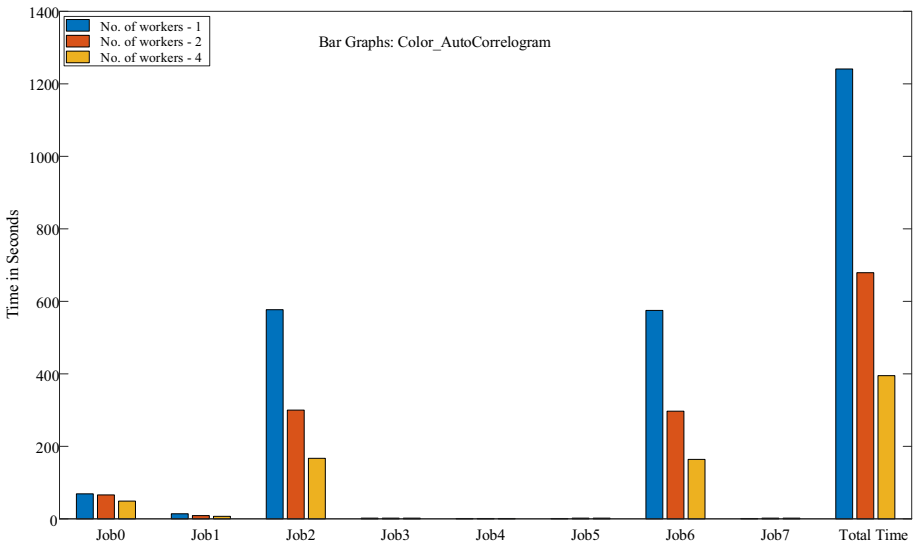


Fig. 21 Time graph for Color_Autocorrelogram for Corel1K dataset. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also

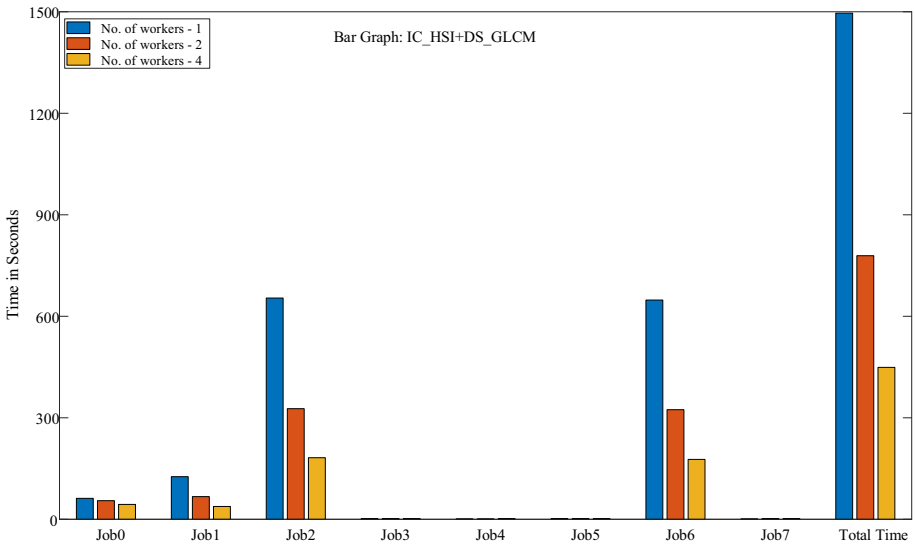


Fig. 22 Time graph for IC_HSI + DS_GLCM for Corel1K dataset. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also

Table 3 The time saved for Corel1K image datasets

Corel1K	LBP	ULBP	ColorHist_RGB	Color_AutoCorrelogram	IC_HSI_DS_GLCM	Total for 5 method
No. of workers: 1	1995	1214	1218	1241	1496	7164
No. of workers: 2	1098	666	717	679	779	3939
No. of workers: 4	636	393	389	395	449	2262
% of saved time 4 VS 2	42%	41%	46%	42%	42%	43%
% of saved time 2 VS 1	45%	45%	41%	45%	48%	45%
% of saved time 4 VS 1	68%	68%	68%	68%	70%	68%

Top part of the table shows the time in *seconds* and the lower part show the percentage of time saved with different number of workers

Fig. 23 VisTex sample images. First image of each of the 40 category is displayed

Table 4 Performance measures for different CBIR methods on VisTex dataset

VisTex	APR	ARR	F-Measure	ANMRR	TMRE
ColorHist_RGB [1998]	88.91	61.96	52.658	0.2866	13.44
Color_Autocorrelogram [2012]	86.64	57.28	49.576	0.3462	15.56
LBP [2002]	97.07	81.00	64.357	0.1216	4.38
ULBP [2002]	98.13	82.75	65.522	0.1081	4.00
IC_HSI + DS_GLCM [2019]	98.67	81.56	65.564	0.1280	5.25

The same results will be obtained by any of the seven modes given in Table 1

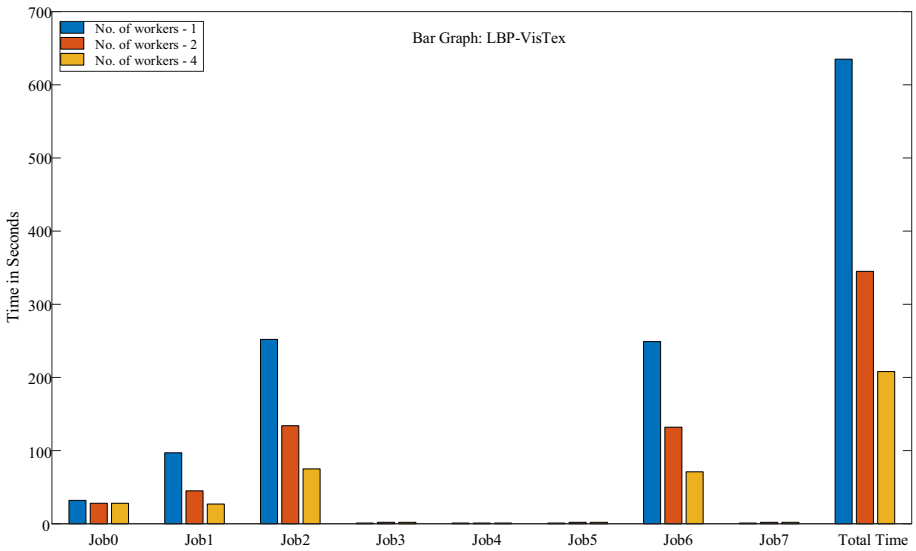
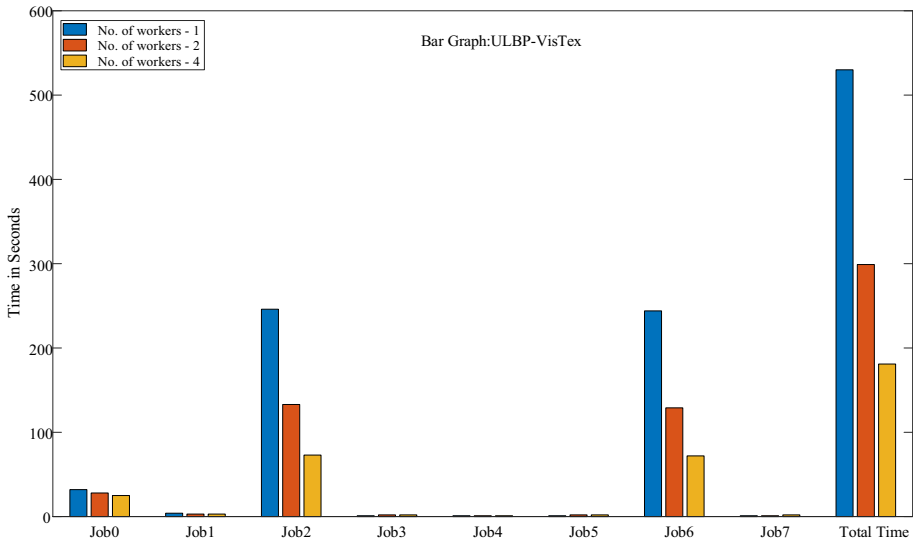


Fig. 24 Time graph for LBP for VisTex dataset. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also

Fig. 25 Time graph for ULBP for VisTex dataset. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also



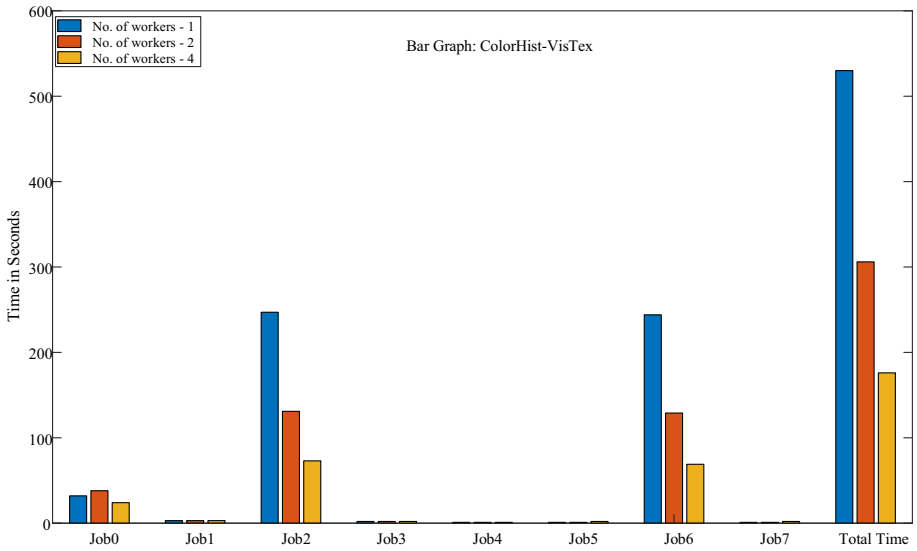


Fig. 26 Time graph for ColorHist_RGB for VisTex dataset. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also

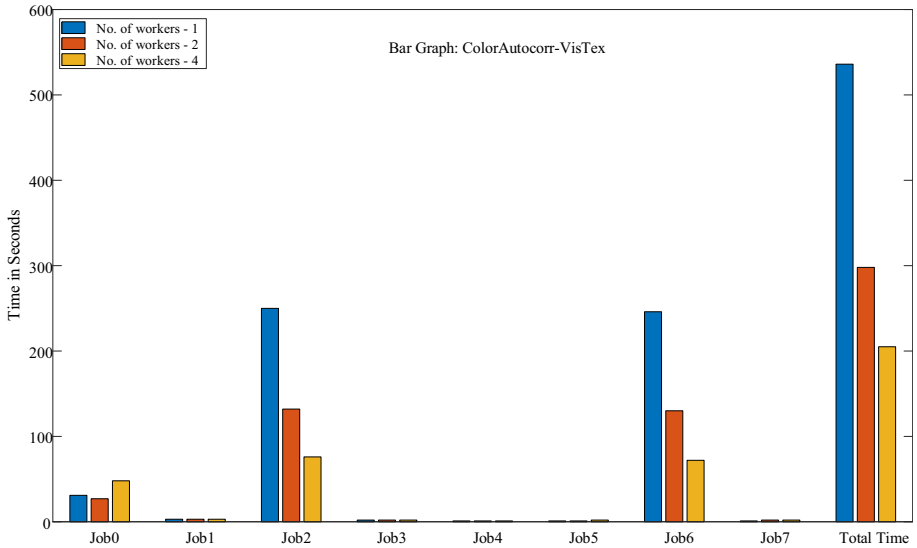


Fig. 27 Time graph for Color_Autocorrelogram for VisTex dataset. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also

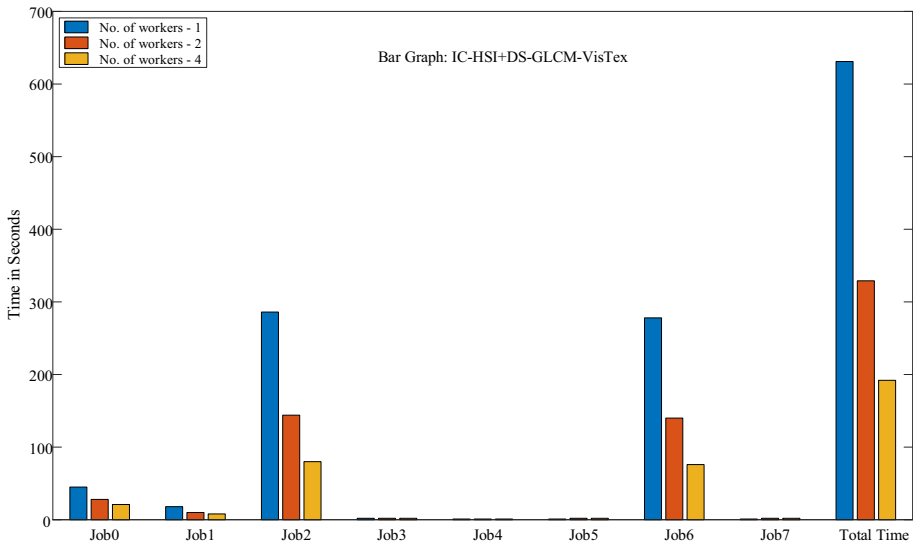


Fig. 28 Time graph for IC_HSI + DS_GLCM for VisTex dataset. For each job 'time to complete' is less if number of workers are increased. Same pattern is observed for total time also

Table 5 The time saved for VisTex image datasets

VisTex	LBP	ULBP	ColorHist_RGB	Color_AutoCorrelogram	IC_HSI_DS_GLCM	Total for 5 method
No. of workers: 1	897	1793	3337	6673	12,712	25,413
No. of workers: 2	487	972	1811	3620	6895	13,784
No. of workers: 4	283	564	1057	2112	4017	8033
% of saved time 4 VS 2	42%	42%	42%	42%	42%	42%
% of saved time 2 VS 1	46%	46%	46%	46%	46%	46%
% of saved time 4 VS 1	68%	69%	68%	68%	68%	68%

Top part of the table shows the time in *seconds* and the lower part show the percentage of time saved with different number of workers

five image retrieval methods on Corel1K, the time saved is 43%, 45% and 68% respectively for the number of workers as 4*vs*2, 2*vs*1 and 4*vs*1 workers. Similarly for VisTex it is 42% 46% and 68%. *Future Extensions:* In Future, more number of images with 96 parallel workers will be analyzed. The state-of-art technologies: Spark and HBase will be used.

References

1. Fadaei S, Amirfattahi R, Ahmadzadeh MR (2016) New content-based image retrieval system based on optimised integration of DCD, wavelet and curvelet features. IET Image Proc 11(2):89–98. https://doi.org/10.1049/iet-ipr.2016.0542
2. Singha M, Hemachandran K (2012) Content based image retrieval using color and texture. Signal Image Process 3(1):39–57. https://doi.org/10.5121/sipij.2012.3104
3. Huang J, Kumar, SR, Mitra M, Zhu WJ, Zabih R (1997) Image indexing using color correlograms. In: Proceedings of IEEE computer society conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, USA, p762–768. https://doi.org/10.1109/CVPR.1997.609412
4. Chun YD, Kim NC, Jang IH (2008) Content-based image retrieval using multiresolution color and texture features. IEEE Trans Multimed 10(6):1073–1084. https://doi.org/10.1109/TMM.2008.2001357
5. Bhunia AK, Bhattacharyya A, Banerjee P, Roy PP, Murala S (2018) A novel feature descriptor for image retrieval by combining modified color histogram and diagonally symmetric co-occurrence texture pattern. arXiv preprint. https://arXiv:1801.00879
6. Ojala T, Pietikainen M, Maenpaa T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans Pattern Anal Mach Intell 24(7):971–987. https://doi.org/10.1109/TPAMI.2002.1017623
7. Heikkilä M, Pietikäinen M, Schmid C (2006) Description of interest regions with center-symmetric local binary patterns. Comput Vision Graph Image Process 4338:58–69. https://doi.org/10.1007/11949619_6
8. Verma M, Raman B, Murala S (2015) Local extrema co-occurrence pattern for color and texture image retrieval. Neurocomputing 165:255–269. https://doi.org/10.1016/j.neucom.2015.03.015
9. Zhang B, Gao Y, Zhao S, Liu J (2009) Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor. IEEE Trans Image Process 19(2):533–544. https://doi.org/10.1109/TIP.2009.203588
10. Murala S, Maheshwari RP, Balasubramanian R (2012) Local tetra patterns: a new feature descriptor for content-based image retrieval. IEEE Trans Image Process 21(5):2874–2886. https://doi.org/10.1109/TIP.2012.2188809
11. Haralick RM, Shanmugam K, Dinstein IH (1973) Textural features for image classification. IEEE Trans Syst Man Cybern 6:610–621. https://doi.org/10.1109/TSMC.1973.4309314
12. Clausi DA (2002) An analysis of co-occurrence texture statistics as a function of grey level quantization. Can J Remote Sens 28(1):45–62. https://doi.org/10.5589/m02-004
13. Hu R, Barnard M, Collomosse J (2010) Gradient field descriptor for sketch based retrieval and localization. In: 2010 IEEE International Conference on Image Processing, Hong Kong, China, p 1025–1028. https://doi.org/10.1109/ICIP.2010.5649331
14. Hu RX, Jia W, Ling H, Zhao Y, Gui J (2013) Angular pattern and binary angular pattern for shape retrieval. IEEE Trans Image Process 23(3):1118–1127. https://doi.org/10.1109/TIP.2013.2286330
15. Osowski S (2002) Fourier and wavelet descriptors for shape recognition using neural networks-a comparative study. Pattern Recogn 35(9):1949–1957. https://doi.org/10.1016/S0031-3203(01)00153-4
16. Mathew SP, Balas VE, Zachariah KP (2015) A content-based image retrieval system based on convex hull geometry. Acta Polytech Hungarica 12(1):103–116. https://doi.org/10.12700/APH.12.1.2015.1.7
17. Rui Y, Huang TS, Chang SF (1999) Image retrieval: Current techniques, promising directions, and open issues. J Vis Commun Image Represent 10(1):39–62. https://doi.org/10.1006/jvci.1999.0413
18. Smeulders AW, Worring M, Santini S, Gupta A, Jain R (2000) Content-based image retrieval at the end of the early years. IEEE Trans Pattern Anal Mach Intell 22(12):1349–1380. https://doi.org/10.1109/34.895972
19. Kokare M, Chatterji BN, Biswas PK (2002) A survey on current content based image retrieval methods. IETE J Res 48(3–4):261–271. https://doi.org/10.1080/03772063.2002.11416285
20. Kanaparthi SK, Raju USN, Shanmukhi P, Aneesha GK, Rahman MEU (2019) Image retrieval by integrating global correlation of color and intensity histograms with local texture features. Multimed Tools Appl. https://doi.org/10.1007/s11042-019-08029-7
21. Cukier V-S (2013) Big data. John Murray Publishers, London
22. Uddin MF, Gupta N (2014) Seven V's of big data understanding big data to extract value. In: Proceedings of the 2014 zone 1

Conference of the American Society for Engineering Education, Bridgeport, CT, USA, p 1–5. https://doi.org/10.1109/ASEEZone1.2014.6820689
23. Tom Shafer (2017) The 42 V's of big data and data science. https://www.elderresearch.com/blog/42-v-of-big-data. Accessed 16 October 2020
24. Fritz Venter, Andrew Stein (2012) Analytics: driving better business decisions. http://analytics-magazine.org/images-a-videos-really-big-data/. Accessed 16 October 2020
25. Mark Sugrue (2015) CCTV—the challenge of sifting through Big Data. https://www.engineersireland.ie/Engineers-Journal/Technology/cctv-the-challenge-of-sifting-through-big-data. Accessed 16 October 2020
26. Wang W, Zhao W, Cai C, Huang J, Xu X, Li L (2015) An efficient image aesthetic analysis system using Hadoop. Signal Process: Image Commun 39:499–508. https://doi.org/10.1016/j.image.2015.07.006
27. Lin Y, Lv F, Zhu S, Yang M, Cour T, Yu K, Huang T (2011) Large-scale image classification: fast feature extraction and SVM training. CVPR 2011, Providence, RI, USA, pp 1689–1696. https://doi.org/10.1109/CVPR.2011.5995477
28. Zhang S, Yang M, Wang X, Lin Y, Tian Q (2013) Semantic-aware co-indexing for image retrieval. In: Proceedings of the IEEE international Conference on computer vision, Sydney, NSW, Australia, p 1673–1680. https://doi.org/10.1109/ICCV.2013.210
29. Dong L, Lin Z, Liang Y, He L, Zhang N, Chen Q, Izquierdo E (2016) A hierarchical distributed processing framework for big image data. IEEE Trans Big Data 2(4):297–309. https://doi.org/10.1109/TBDATA.2016.2613992
30. ProjectPro. Healthcare applications of Hadoop and Big data. https://www.dezyre.com/article/5-healthcare-applications-of-hadoop-and-big-data/85. Accessed 16 October 2020
31. Koppad SH, Kumar A (2016) Application of big data analytics in healthcare system to predict COPD. In: 2016 International Conference on Circuit, Power and Computing Technologies, Nagercoil, India, p 1–5. https://doi.org/10.1109/ICCPCT.2016.7530248
32. Chen M, Xugang Z, Guansen W, Jianxiao M (2015) A preliminary discussion on the application of big data in urban residents travel guidance. In: International Conference on Intelligent Transportation, Big Data and Smart City, Halong Bay, Vietnam, p 47–50. https://doi.org/10.1109/ICITBS.2015.18
33. Im H, Hong B, Jeon S, Hong J (2016) Bigdata analytics on CCTV images for collecting traffic information. In; International Conference on Big Data and Smart Computing (BigComp), Hong Kong, China, p 525–528. https://doi.org/10.1109/BIGCOMP.2016.7425985
34. Applications of Big Data Drive Industries. https://www.simplilearn.com/tutorials/big-data-tutorial/big-data-applications. Accessed 16 October 2020
35. Welcome to Apache™ Hadoop®!. http://hadoop.apache.org/. Accessed 15 October 2020
36. Raju USN, Chaitanya B, Kumar KP, Krishna, PN, Mishra P (2016) Video copy detection in distributed environment. In: IEEE Second International Conference on Multimedia Big Data (BigMM), Taipei, Taiwan, p 432–435. https://doi.org/10.1109/bigmm.2016.94
37. Turkington G (2013) Hadoop beginner's guide. Packt Publishing Ltd, Birmingham
38. Perera S, Gunarathne T (2013) Hadoop MapReduce cookbook. Packt Publishing Ltd, Birmingham
39. Istephan S, Siadat MR (2016) Unstructured medical image query using big data–an epilepsy case study. J Biomed Inform 59:218–226. https://doi.org/10.1016/j.jbi.2015.12.005
40. Raju USN, Suresh Kumar K, Haran P, Boppana RS, Kumar N (2020) Content-based image retrieval using local texture features in distributed environment. Int J Wavelets Multiresolut Inf Process 18(01):1941001. https://doi.org/10.1142/S0219691319410017
41. Zhang L, Jung T, Liu K, Li XY, Ding X, Gu J, Liu Y (2017) Pic: Enable large-scale privacy preserving content-based image search on cloud. IEEE Trans Parallel Distrib Syst 28(11):3258–3271. https://doi.org/10.1109/TPDS.2017.2712148
42. Netalkar Rohan Kishor, Hillol Barman, Raju USN, Suresh Kumar Kanaparthi and Harika Ala (2021) Content based image retrieval using frequency domain features: zigzag Scanning of DCT coefficients. In: Proceedings of the International Conference on Artificial Intelligence and Smart Systems ICAIS, p 1535–1540. https://doi.org/10.1109/ICAIS50930.2021.9396008
43. Cjames Z. Wang, Corel-1K image data set, modeling objects, concepts, aesthetics and emotions in big visual data, http://wang.ist.psu.edu/docs/home.shtml. Accessed 16 October 2020
44. MIT media lab: VisMod group. https://vismod.media.mit.edu/pub/VisTex/. Accessed 16 October 2020