Journal of Engineering and Applied Sciences 13 (Special Issue 5): 4720-4727, 2018
ISSN: 1816-949X

Comparative Between Hash Function and MapReduce to Constructing Self-Similarity Matrix Based on Fractal Features

Israa Hadi and Firas Sabar Miften
Faculty of Information Technology, University of Babylon, Hillah, 00964 Babylon, Iraq

Abstract: Self-similarity is that property of being invariant with a different scale. The most of the researches used fractal dimension to calculate the self-similarity. In this study, we present a new algorithm, based on matching rang and domain fractal to find self-similarity properties of the data sets which can be used in data mining such as clustering and classification. This research focuses on two main points. Firstly, ranges and domains matching technique is used to extract self-similarity features from the images. Secondly, comparative between hash function and MapReduce to reducing time complexity. The experimental results show that the images from the same class are grouped to gather. The proposed methods of reducing time complexity results are presented and compared with traditional methods. The hash function reduced the complexity 0 (m×n) to 0 (mlogn) while MapReduce technique reduce the complexity 0(m×n) to 0(m×n/t) for the time where t is a number a of map task.

Key words: Fractal, self-similarity, fractal dimension, PIFS, experimental results, comparative, MapReduce

INTRODUCTION

In image retrieval, similarity metrics are widely active. Several image processing approaches have been proposed lately to find solutions to the dilemma of content based image retrieval. The fractal which stands for a modern theory suggested through last century, enriches a novel method to the problem.

The mathematical description of fractal has been clarified by authority of Mandelbrot (1983) which realizes a fractal is a collection that the dimension of Hausdorff Besicovich severely outreaches this topological dimension. Though, it represents a very abstract explanation. In general, it is possible to identify the fractal as a fragmentized geometric form which may be distributed into segments where each part is (at the very least nearly) a reduced-size version of that whole. Fractals may be seen as in general self-identical as well as independent of scale (Leodkin *et al*., 2009). The conception of self-similarity in which an entity seems to be similar to various scales or objects whose small pieces resemble the whole as an example this is the Sierpinski triangle in Fig. 1 illustrate self-similarity. In this mathematical object, each little piece is an exact smaller copy of the whole object (Mandelbrot, 1983; Kaye, 2008; Mon, 1986).

Image similarity measurement is a main process to distinguish the place through measuring the image

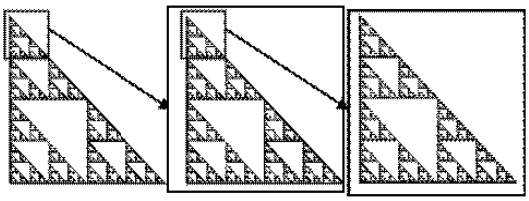


Fig. 1: Sierpinski triangle illustrate self-similarity

similarity. The basic to measure the image similarity is to construct a vector or matrix which can label the individual characteristic of image and recognize it from others. Most researchers developed techniques to mining images based on fractal theory which is used the fractal dimension as the features. The extracted features are used as input to mining algorithms.

Shih (2014) extracted fractal dimension as features to cluster images. He suggested that using dimension of fractal for image sequencing can develop the sufficiency in content-based image retrieval. Liu and He (2010) used the dimension of local fractal in the form of features to recognized the dissimilarity between on the one hand the tree leaf and on the other hand natural background. Liu *et al*. (2014) introduced fractal theory as image properties extraction. Fuzzy sequencing is used with neural network for processing characteristics. Tasoulis *et al*. (2014) present a classification tool for image of computer-assisted based on fractal as well as

Corresponding Author: Israa Hadi, Faculty of Information Technology, University of Babylon, Hillah, 00964 Babylon, Iraq

fuzzy sequencing for the quantification of rate of the IPF (Idiopathic Pulmonary Fibrosis) in images. Wein and Blake (1996) used the technique of clustering operation on image domain mass with the clusters constituted by the use of k-d trees.

Partitioned Iterated Function Systems (PIFS): Fractal image compression used PIFS which is applied to a single image. Suppose, we are dealing with a grayscale image 1 that has size w×w. The image is partitioned into n non-overlapping block size b×b is named as range blocks be represented by $R = \{R_1, R_2, ..., R_n\}$ note that $1 = \bigcup_{i=1}^{n} R_i$. Let D be the set of all possible blocks in the same image 1 which overlapping block of size 2×2b is named as domain blocks be represented by $D = \{D_1, D_2, ..., D_m\}$. Now for each range block R_i: $i \in \{1, 2, 3, ..., n\}$ must find a suitably matched domain block D_i: $j \in \{1, 2, 3, ..., m\}$ then save transformation parameters thus, obtained is called the fractal codes of the image 1.

Hash function: A hash function depends on a hash-key value to yields a bucket number. The hash-key value is an argument. The bucket number takes an integer number which lay between 0 and P-1, P is the number of obtained buckets (He and Petoukhov, 2011). Consider hash-keys are positive numbers. The simple form of the hash function is defined as h(x) = x mod B (Leskovee *et al*., 2014). Figure 2 demonstrates a hash function.

MapReduce technique: One of the effective ways to manipulate and process a large data is a MapReduce technique in which a parallel approach is used. It was developed to reduce the execution ty using parallel processing. It was used by Google to process the Web data in a parallel method. The general methodology of the MapReduce is described as:

- The map stage received the input data from a distributed file to produce a sequence of key and list (values) as pairs
- The second stage, it collects the mapper output of the reduce function. The output of map is sorted by the key values
- The reduce function are paired all the values related to one key. The method of combining values is determined by a user program (Dean and Ghemawat, 2008). Figure 3 shows the methodology of the MapReduce framework

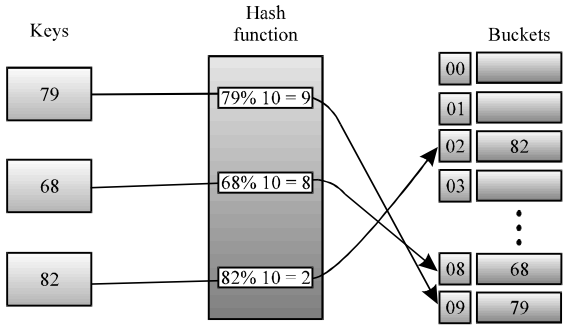


Fig. 2: The hash function where B = 10

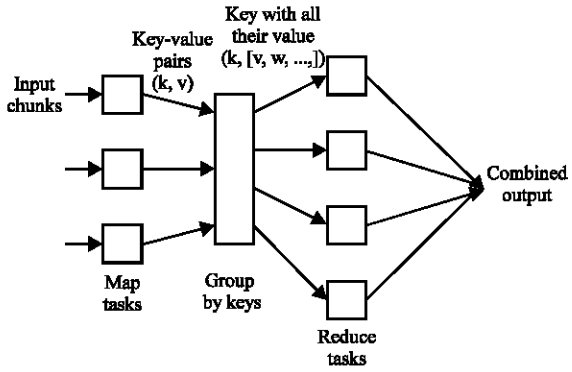


Fig. 3: Architecture of MapReduce programming model

MATERIALS AND METHODS

Our contribution: Upgrade PIFS to use in recognition patterns where most of the researchers used a fractal dimension. PIFS applied on a single image while upgrade PIFS applied on multi images where range and domain blocks extract from all images and each range block machining with all domain blocks which extract from all images. The output of upgrade PIFS is not transformation parameters but the index of the image that has range block and the index of the image that has domain block. Finally, count the number of occurs indexes to gather.

All images pass through 2 stages. The 1st known domain pool which is constituted by partitioning images into overlapping block fixed size and the second is ranges is formed from partitioning images into non-overlapping block fixed size. As far as the domain is concerned, the overlapping square results from image partitioning, the group D of domains involves all square degrees enriched by sides size 16. The domain amount double the range size. Consequently, it subsamples of 2×2 pixels to arrive a decreased domain by the identical sum average of pixels like the range.

Regarding the range, the image gets partitioning to the form of squares of a non-overlapping that has sides

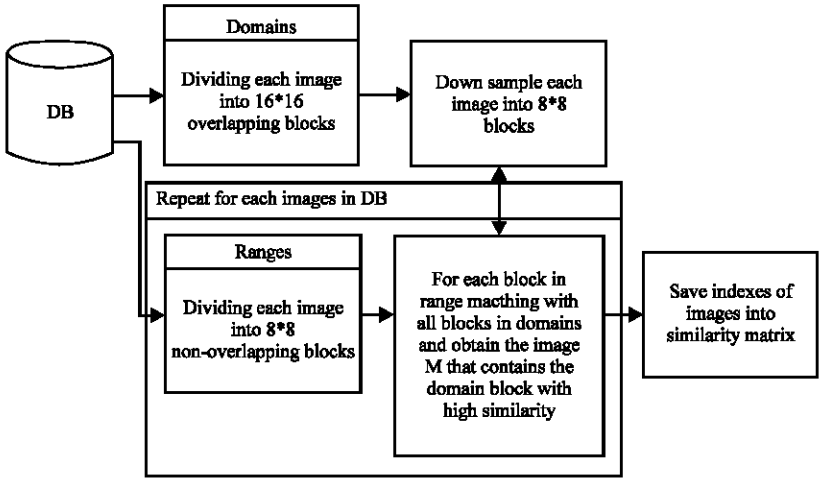


Fig. 4: Search operation to find the best matching between range and domains

size 8. Therefore, concerning every range, the algorithm attempts to detect a domain which provides the minimum error smaller. With respect to all recently invented ranges, the procedure is re-played, i.e., becoming suitable domains have been looked for ranges. The encoding ends when there are no ranges that remain uncovered and save the name of image which contains the domain block that fitting the current range. RMSE is used to find the best fitting between range and domain (Fig. 4).

Self-similarity matrix is computed as result from matching operation where every entry represents the number of times matching between images. Algorithm 1 read k of images while the output is k×k matrix called similarity matrix M where an entity of $M(t_1. t_2)$ represent the number of matching range from image t_1 with the domain from image t_2.

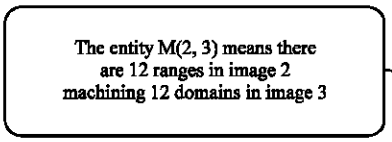
Algorithm 1; Calculate similarity matrix among images:

Input: k face images
Output: M similarity matrix of size k×k
Begin
/*Building domain Pool D*/
1: For each image I_i: I∈{1, 2, 3, ..., k}
2: Dividing I_i images into overlapping blocks D = {D_{i1}, D_{i2}, D_{im}}. The size of each domain is 2b×2b then down sample to b×b size. Where m is number of domains in image and b is chosen value
End for
/*Matching range with domain Pool D*/
3: For each image I_i: I∈{1, 2, 3, ..., k}
4: Dividing I_i image into non-overlapping range blocks R ={R_{i1}, R_{i2}, ..., R_{im}} that size b×b. Where n is number of ranges in image
End for
5: For each range R_{ij}∈R: I∈{1, 2, ..., k} and j∈{1, 2, ..., n}
6: For each domain D_{it}∈D: t∈{1, 2, ..., k} and l∈{1, 2, ..., m}
7: If RMSE (R_{ij}, D_{it}) is smallest
M (i.t) = M (i.t)+1
End for
End for
8: Return M

The algorithm read k images as input to the algorithm. To build domain Pool D in steps 1 and 2 all images are divided into overlapping blocks with size 2×2b because the size of the domain block must be double size of range block. Down sample must achieve on the size of the domain to facilitate the matching between ranges and domains with equal size. Average four pixels is used as down sample. Each domain indexed by image number which has this domain and domain number. Save all domains in pool domain D to using it later. With a new round in step 3 take images again to dividing into non-overlapping blocks with size b×b to get ranges R. Each range in R compare with all domains in domain pool to find high similarity domain then recorded i the index of the image that contains range and t index of the image that contain the domain. The output of the algorithm is M similarity matrix of size k×k where k is a number of input images. Rows in M represent i index of the image that contains range and column represent t index of the image that contains the domain. The entity M(i.t) is number how many ranges in image i machining with domains in image t (Fig. 5). The output matrix is not symmetric.

Reducing time complexity: The high calculating in the comparing step still exist time-consuming. In another word, consider n is a range blocks number and m is a domain blocks number. The finding of the best equal among a range and a domain blocks is 0(n×m). To reducing the complexity of time, hash function and MapReduce techniques are employed.

Hash function: We well employing a hash function to reducing the time of discovery the greatest matching among range and domains. Each domain block in domains



No	1	2	3	4	5	6	7	8	9	10	11	12
1	0	12	15	3	2	14	9	6	2	3	1	3
2	17	0	12	6	6	19	9	3	1	4	1	0
3	15	19	0	4	2	8	10	5	0	7	1	2
4	4	7	1	0	13	9	10	3	2	4	0	1
5	1	4	2	9	0	6	3	8	9	8	1	1
6	16	14	2	8	2	0	7	2	4	2	1	3
7	10	6	2	8	1	13	0	5	4	5	1	1
8	3	2	3	5	2	4	0	0	14	10	3	2
9	3	8	5	8	6	7	7	9	0	9	1	4
10	10	4	9	4	9	4	2	6	3	0	1	0
11	1	4	12	1	1	1	3	1	1	2	0	9
12	1	4	0	1	0	2	1	2	1	1	2	0

Fig. 5: What is entities of matrix M

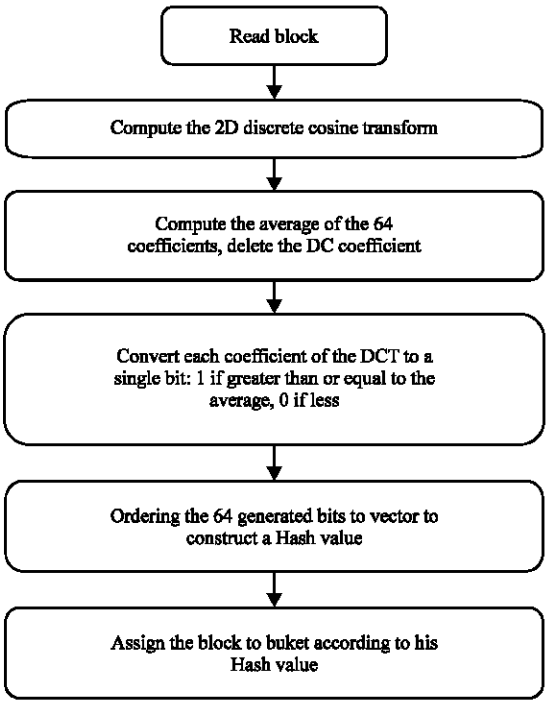


Fig. 6: Block diagram of hashing block to bucket

pool passes through many steps to generate hash-key which map domain block to the certain bucket. Rang block is hashed to the bucket and compare with only domains in the same bucket. The hash function reduces the time complexity 0(n×m) to sequential complexity 0(nlogm) and n is range blocks number, m is domain blocks number. Figure 6 illustrates the hashing block to bucket.

The process of hashing the block to certain bucket begins with a reading block then applying 2D (DCT) Discrete Cosine Transform. The DCT coefficients are classified into tow coefficients. The coefficient which has an index (0, 0) is DC coefficient while others coefficients are AC coefficients. All coefficients will convert to binary form by using threshold. The threshold is the average of AC coefficients. Because of DC coefficient has a high value that effect on computing threshold. Therefore, DC coefficient will exclude form equation of computing threshold. Each coefficient of the DCT is converted to a single bit where one if greater than or equal to the threshold otherwise zero if less than the threshold.

Now, we have 64 bits block by ordering the generated bits to vector to construct a hash value. The 1st 16 bit is chosen to convert binary number to decimal number which represents the bucket number. From 16 bit can have 65536 buckets. Each block is mapped to (0-65535) buckets. Algorithm 2 mapping the block to the certain bucket.

Algorithm 2; Hashing block to bucket:

Input: B block
Output: T is 16 bits vector, N is bucket number
Begin
1. Read B block of size (8×8)
2. Compute DCT = Discrete Cosine Transform (B)
3. Compute Threshold = $\Sigma_{i=0}^{7}\Sigma_{j=0}^{7}$ /* DCT(i, j) DCT(0,0) Do not enter into account */
4. For each i:i∈{0, 1, 2, ..., 7}
5. For each j:j∈{0, 1, 2, ..., 7}
6. if DCT (i, j)≥Threshold
7. BT (i, j) = 1
else
9. BT (i, j) = 0
End if
End for
End for
10. Compute vector T = zig zag scan (BT)
11. For each i: i∈ {0, 1, 2, ..., 15}
12. Compute P(I) = T(i)
End for
13. Convert binary number P to decimal number N:N∈{1, 2, ..., 655367}
14. Return (T, N)

Domain-range comparison with hash function: To reduce the matching between range and domains, the hash function is employed. First, all domains in domains pool are hashed to bucket number in the hash table. Each range in ranges pool well hashed to a certain bucket. The ranges in range pool are matched with only domains in the same bucket.

Algorithm 3; Similarity matrix construction:

Input: k images
Output: M similarity matrix of size k×k
Begin
1. For each range R_{ij}∈R: i∈{1, 2, ..., k} and j∈{1, 2, ..., n}
2. Call algorithm 2 to hash R_{ij} to bucket B
3. For each domain in bucket B where D_{tl}∈D:t∈{1. 2, ..., k} and l∈{1, 2, ..., m}
4. If Hamming (R_{ij}.D_{tl}) is smallest
5. M(i, t) = M(i, t)+1
End If
End for
End for
6. Return (M)

Each domain is hashed to certain bucket number in the hash table. Hashing all domains in pool domain D to using it later. With a new round in step 2 take images again to call Algorithm 2 to construct ranges pool R. Each range in R well hashed to certain bucket number and compare only with all domains in same bucket to find high similarity domain by using hamming distance then recorded i the index of image that contain range and t index of image that contain domain. The output of the algorithm is M similarity matrix of size k×k where k is the number of input images.

MapReduce technique: In this study, we will explain how employing MapReduce technique to reducing the time of finding the best matching between range and domains. Matching had been employed to work with MapReduce technique. MapReduce technique is a programming tool that requires the users to put their codes for parallel implementation. MapReduce technique reduces the time complexity 0(n×m) to 0(m×n/t) and n is range blocks number, m is domain blocks number, t is a number a of the map task. The MapReduce algorithms are sketched in Fig. 7. Algorithm 4 illustrates the pseudo code to the employee the MapReduce technique by provides the parallel in a multicore environment.

Algorithm 4; MapReduce implementation:

Input: k images
Output: M similarity matrix of size k×k
Begin
1. Splitting domains pool D to t pools where t is number of map task
2. For each range R_{ij}∈R: i∈{1, 2, .., k} and j∈{1, 2, .., n}
3. Execute Map function for each R_{ij} with D_t: l∈{1, 2, .., t}
4. Execute the Reduce function for all map functions
5. End for
6. Return (M)

Algorithm 4 read K images and return M(k×k) where M is similarity matrix. In step 1 the domains pool D is split into t pools where t is a number of map task. Map functions read single range block from ranges pool R and a piece of splitting domains pool D (many of domains). Map function will produce (key, value) pairs where the key is the number of range block and value is [number of domain block, RMSE]. As an example (90, [40, 30.5]) the key represents the range 90 matching domain 40 with root mean square error 30.5.

Reduce function uses the output of the Map function and performs the calculations and produces (key, value) pairs. The output of the algorithm is M similarity matrix of size k×k where k is the number of input images. All outputs are written to file.

Algorithm 5; Map function:

Input: (R_i, D_l): R_i is single range block, D_l piece of domain pool.
Output: (key, [values]) (Map with pair values where key is range number and values is [number of domain block, RMSE])
Begin
1. For each domain D_{tl} where D_{tl}∈D_l: t∈{1. 2.. k} and j∈{1. 2.. m}
2. If RMSE (R_i, D_{tl}) is smallest
3. rmse = RMSE (R_{ij}, D_{tl})
End if
End for
4. Return Map_output (keys, [values]) : (R_{ij}., [D_{tl}, rmse])
5. End Algorithm

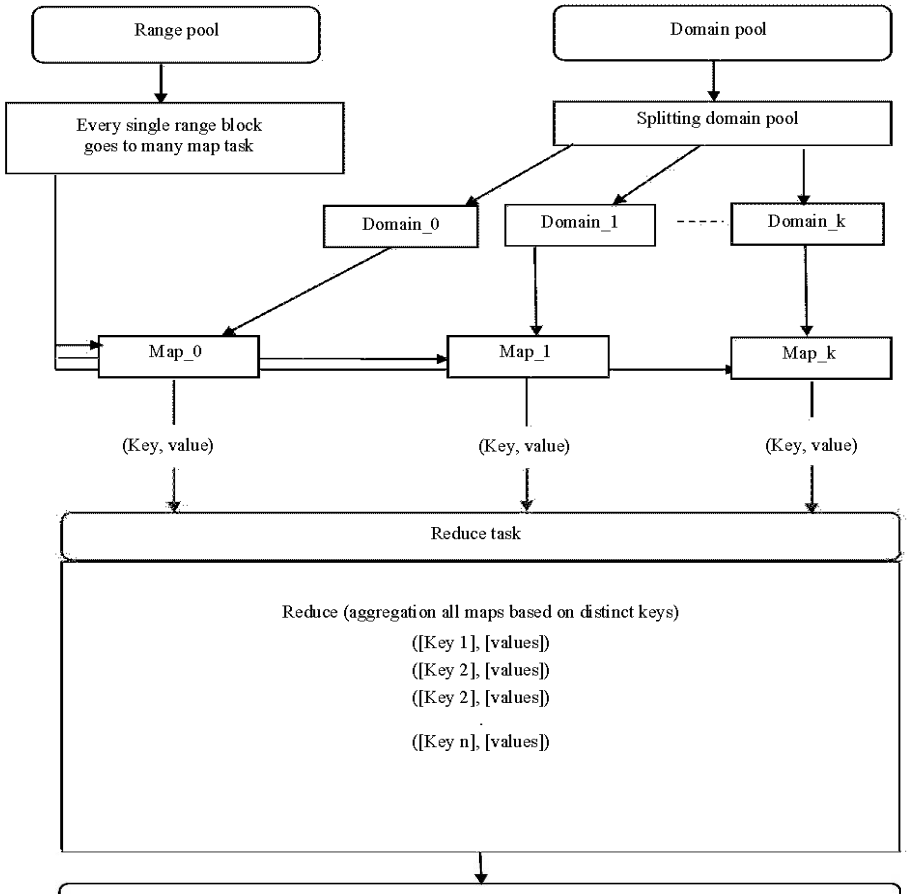
The range R_i compare with all domains in D_l piece of domain pool to find high similarity domain then records the range and t index of the image that contains the domain. Furthermore, it recorded the RMSE the best matching between range and domain blocks. Map function will produce (key, value) pairs where the key is the number of range block and value is [number of domain block, RMSE]. The reduce function is presented in an Algorithm 6.

Algorithm 6; Reduce function:

Input: map output (keys, [values])
Output: M similarity matrix of size k×k
Begin
1. For each (R_{ij}, [D_{tl}, RMSE]) in map
2. If RMSE is smallest
3. M (i, t) = M (i, t)+1
End if
End for
4. Return (M)

The intermediate results are read and a final result is produced by reduce function. Thus, to find the greatest equivalent among blocks of range and blocks of domain in a data set, the map function can find the best matching between range and domain blocks in each split of input data and then the reduce function can find the single the greatest equivalent among blocks of range and blocks of domain among all of the intermediate dataset. The output of the algorithm is M similarity matrix of size k×k where k is the number of input images.

The experimental data: The two database is used in this study. The first is the FERET database of images. This



No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	12	15	3	2	14	9	6	2	3	1	3	0	0	0	1	0	0	2	0	4	0	2	2	0	0	0	0	0	1
2	17	0	12	6	6	19	9	3	1	4	1	0	1	0	3	0	0	1	0	2	0	0	0	1	1	2	1	0	0	2
3	15	19	0	4	2	8	10	5	0	7	1	2	2	0	2	1	1	0	0	2	0	0	1	0	1	0	1	0	0	0
4	4	7	2	0	13	9	10	3	2	4	0	1	1	0	1	0	2	0	1	0	1	3	2	1	1	2	0	0	0	1
5	1	4	3	9	0	6	3	8	9	8	1	1	0	1	1	2	1	1	1	1	2	2	2	0	2	0	0	0	0	0
6	16	14	12	8	2	0	7	2	4	2	1	3	1	0	0	2	0	1	1	2	2	0	1	1	0	0	0	0	1	2
7	10	6	9	8	1	13	0	5	4	5	1	1	0	0	1	2	0	1	1	1	1	0	2	1	2	0	1	1	1	0
8	3	2	5	5	2	4	0	0	14	10	3	2	3	0	1	1	1	0	1	0	0	0	1	3	0	1	2	0	0	0
9	3	8	1	8	6	7	7	9	0	9	1	4	0	3	2	0	2	0	0	2	0	2	2	1	0	0	1	1	0	2
10	10	4	2	4	9	4	2	6	3	0	1	0	2	1	0	1	3	0	3	1	5	2	6	1	0	0	2	2	0	1
11	1	4	2	1	1	1	3	1	1	2	0	9	18	4	5	2	5	7	2	12	1	1	1	1	4	0	1	1	1	1
12	1	4	0	1	0	2	1	2	1	1	2	0	5	22	2	19	7	10	10	3	1	2	2	0	0	0	0	0	1	0

Fig. 7: Employing algorithm with MapReduce technique

database contains a total of 11338 facial images. These have been collected via. photographing 994 samples at different angles during the course of 15 meetings that took place 1993 and 1996. The color FERET is mostly a color copy of the authentic Facial Recognition Technology (FERET) database that was discharged in 20001 and encompassed 14051 grayscale images. The database aimed at developing, examine and appreciate face recognition algorithms. The total images during the color FERET database amount 512 in the form of 768 pixels. Besides, the files located through PPM-format.

Second, faces of the ORL database involves a group of face images gathered from April 1992-1994 by help of the lab. Ten dissimilar images for every of 40 discrete subjects are identified. Concerning particular subjects, these images have gathered from various times, ranging from lighting, the expressions of the face (open/closed eyes, smiling/not smiling) to the information of faces (no glasses/glasses). All of the images have considered in opposite to a dark analogous basis associating the subjects in a vertical, ahead position (with endurance for certain side action). Additionally, the image size is 92×112 pixels with 256 gray ranks per pixel (CLUC., 2002).

RESULTS AND DISCUSSION

In this study, we involved two databases: ORL and FERET as mentioned in this study. Fractal self-similarity technique was used to extract features from the databases. For ORL database choosing ten different images of each of 40 subjects and total images are 400. While the technique is implemented on FERET database on the two rounds. First choosing different images for 739 subjects and total images are 7762. In Second round, choosing different images 994 subjects and total images are 11338.

The results of the proposed method showed the images from the same class are grouped. In Fig. 8, the images from 1-10 which belong to one object have high similarity.

The hash function which proposed in study (6.1). The hash function reduces the time complexity 0(n×m) to sequential complexity 0(n log m) and n is range blocks number, m is domain blocks number. The execution time is reduced significantly for different implementations. Table 1 presents the results of execution time comparisons of the hash function and traditional algorithm with deferent images datasets.

Table 2 shows a significant difference in implementation time between traditional algorithm and hash function. Similarity, matrix was extracted by the hash function must be tested and compared.

The precision and recall are not as good as the original but the computing time is largely reduced, i.e., from few minutes to few seconds.

The MapReduce Model consists of two functions: a map function to specify the (key, [values]) (Map with pair values where the key is range number and values is [number of domain block, RMSE]. The implementation is done using CORE i5 computer in C++ that provides the parallel in a multicore environment. Since, our computer has only four cores. Therefore, we can run just four map function with one reduce function (one core to one map function). The map Algorithm 5 produces pairs of (key, [values]) and Algorithm 6 reads the intermediate results and produces a final result as Reduce function. MapReduce technique reduces the time complexity 0(n×m) to 0(n×m/t) and n is range blocks number, m is domain blocks number, t is a number a of map task.

The precision and recall are exactly of the original, but the computing time is reduced depending on a number of maps/reduce function. Compare three proposed methods based on the number of images and execution time (Table 3). Figure 9 shows the results of execution time and compare them together.

Table 1: Results comparisons of the hash function and traditional algorithm

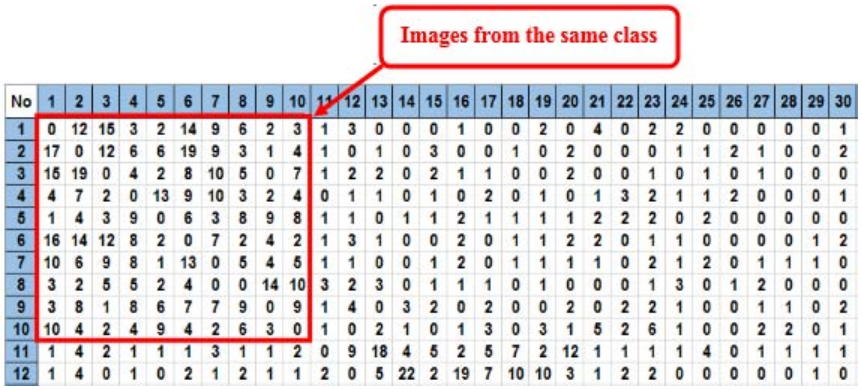
Algorithm	Dataset	No. of images	Encoding time (sec)	Encoding time (min.)
Traditional	orl	100	1251.210	20.854
Traditional	FERET	100	1491.990	24.867
Hash function	orl	100	0.345	0.006
Hash function	orl	400	3.147	0.052
Hash function	FERET	1000	19.486	0.325
Hash function	FERET	2000	129.825	2.164

Table 2: Comparison of the accuracy

Methods	Precision (%)	Recall (%)
k-mean with hash function	75	70
k-mean with traditional	99	99

Table 3: Results of MapReduce execution time

Algorithm	Dataset	No. of images	Encoding time (sec)	Encoding time (min.)
MapReduce	orl	100	350.350	5.839
MapReduce	FERET	100	380.990	6.350
MapReduce	orl	400	1704.310	28.405



No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	12	15	3	2	14	9	6	2	3	1	3	0	0	0	1	0	0	2	0	4	0	2	2	0	0	0	0	0	1
2	17	0	12	6	6	19	9	3	1	4	1	0	1	0	3	0	0	1	0	2	0	0	0	1	1	2	1	0	0	2
3	15	19	0	4	2	8	10	5	0	7	1	2	2	0	2	1	1	0	0	2	0	0	1	0	1	0	1	0	0	0
4	4	7	2	0	13	9	10	3	2	4	0	1	1	0	1	0	2	0	1	0	1	3	2	1	1	2	0	0	0	1
5	1	4	3	9	0	6	3	8	9	8	1	1	0	1	1	2	1	1	1	1	2	2	2	0	2	0	0	0	0	0
6	16	14	12	8	2	0	7	2	4	2	1	3	1	0	0	2	0	1	1	2	2	0	1	1	0	0	0	0	1	2
7	10	6	9	8	1	13	0	5	4	5	1	1	0	0	1	2	0	1	1	1	1	0	2	1	2	0	1	1	1	0
8	3	2	5	5	2	4	0	0	14	10	3	2	3	0	1	1	1	0	1	0	0	0	1	3	0	1	2	0	0	0
9	3	8	1	8	6	7	7	9	0	9	1	4	0	3	2	0	2	0	0	2	0	2	2	1	0	0	1	1	0	2
10	10	4	2	4	9	4	2	6	3	0	1	0	2	1	0	1	3	0	3	1	5	2	6	1	0	0	2	2	0	1
11	1	4	2	1	1	1	3	1	1	2	0	9	18	4	5	2	5	7	2	12	1	1	1	1	4	0	1	1	1	1
12	1	4	0	1	0	2	1	2	1	1	2	0	5	22	2	19	7	10	10	3	1	2	2	0	0	0	0	0	1	0

Fig. 8: Similarity matrix represent the similarity between images

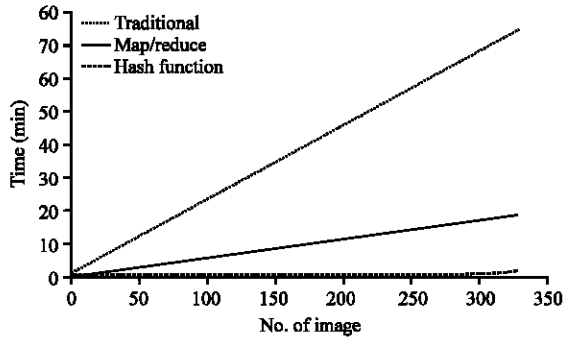


Fig. 9: Performance evaluation for traditional, MapReduce and hash function

CONCLUSION

The study presents a fractal method to extract self-similarity features for clustering. This method yields the results of 95 and 96% for clustering precision and recall, respectively. In addition, the proposed method proved the possibility of the use fractal self-similarity features in image clustering without the use of fractal dimension. To sum up, the proposed method is very efficient for image clustering. Hash will be also useful for the clustering of other data. It has been shown high reducing time but accuracy has been reduced by employing a hash function to reducing the time of finding the best matching between range and domains. Each domain block in domains pool passes through many steps to generate hash-key which map domain block to the certain bucket. Rang block is hashed to the bucket and compare with only domains in the same bucket. The hash function reduces 0(n× m) to sequential complexity 0(n log m), n is range blocks number and m is domain blocks number. MapReduce has been shown high reducing time with the same accuracy by employing MapReduce technique to reducing the time of finding the best matching between range and domains. Matching had been employed to work with MapReduce technique. MapReduce technique reduces the time complexity 0(n×m) to (n×n/t), n is range blocks number and m is domain blocks number and t is a number an of map task.

REFERENCES

CLUC., 2002. The ORL database. University of Cambridge, Cambridge, England http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Dean, J. and S. Ghemawat, 2008. MapReduce: Simplified data processing on large clusters. Commun. ACM, 51: 107-113.

He, M. and S. Petoukhov, 2011. Mathematics of Bioinformatics: Theory, Methods and Applications. Vol. 19, John Wiley & Sons, Hoboken, New Jersey, USA., ISBN:978-0-470-40443-0, Pages: 301.

Kaye, B.H., 2008. A Random Walk Through Fractal Dimensions. 2nd Edn., John Wiley & Sons, Hoboken, New Jersey, USA., ISBN:3-527-29078-8, pp: 136-426.

Leodkin, M.A., T.A. Lebedkina and A. Jacques, 2009. Multifractal Analysis of Unstable Plastic Flow. Nova Science Publishers, New York, Hauppauge, ISBN9781606924686,.

Leskovec, J., A. Rajaraman and J.D. Ullman, 2014. Mining of Massive Datasets. Cambridge University Press, Cambridge, England, ISBN:978-1-107-07723-2, Pages: 467.

Liu, G., N. Chen, C. Ou, Y. Liao and Y. Yu, 2014. Image feature extraction based on multifractal theory. Proceeding of the International 2014 IEEE Workshop on Electronics Computer and Applications, May 8-9, 2014, IEEE , New Taipei, Taiwan, ISBN:978-1-4799-4565-8, pp: 1023-1026.

Liu, W. and Y. He, 2010. Fractal features and fuzzy clustering based tree leaf segmentation from images with the complex background. Proceeding of International Conference on Computer, Mechatronics, Control and Electronic Engineering, August 24-26, 2010, IEEE, New Taipei City, Taiwan, ISBN:978-1-4244-7957-3, pp: 317-321.

Mandelbrot, B.B., 1983. The Fractal Geometry of Nature. Vol. 173, W.H. Freeman Publisher, New York, USA., Pages: 468.

Mon, K.K., 1986. Self-similarity and fractal dimension of a roughening interface by Monte Carlo simulations. Phys. Rev. Lett., 57: 866-1986.

Shih, A.Z., 2014. An examination of color image clustering by using fractal signatures. Proceedings of the 2014 International Conference on Machine Learning and Cybernetics, July 13-16, 2014, IEEE, New Taipei City, Taiwan, ISBN:978-1-4799-4216-9, pp: 410-413.

Tasoulis, S.K., I. Maglogiannis and V.P. Plagianakos, 2014. Fractal analysis and fuzzy c-means clustering for quantification of fibrotic microscopy images. Artif. Intell. Rev., 42: 313-329.

Wein, C.J. and I.F. Blake, 1996. On the performance of fractal compression with clustering. IEEE Trans. Image Process., 5: 522-526.