A High-performance Computing Method for Photographic Mosaics upon the Hadoop Framework

Chin-Feng Lee[1], Jau-Ji Shen[2] , Kun-Liang Hou[2], Fang-Wei Hsu[2]

[1] Department of Information Management, Chaoyang University of Technology, Taiwan
[2] Department of Management Information Systems, National Chung Hsing University, Taiwan
lcf@cyut.edu.tw, jjshen@nchu.edu.tw, kunliang.hou@gmail.com, davidhsu1115@gmail.com

Abstract

Since digital images are non-structure data, in order to apply some image processing techniques on digital images, we need high computing power. Nowadays, digital images have become an issue of Big Data, so we decide to implement an adaptive K-Medoids based on a popular Big Data analysis tool, Hadoop, in our research. Hadoop can provide high computing power to do processes of high computational complexity required by the algorithm of mosaic images, especially, in a Big Data environment where there are tons of images to be dealt. Our research focuses on three main goals. First, we use an unsupervised clustering method, K-Medoids, to cluster the image dataset and build a codebook, and then we can use the codebook to generate the mosaic image to reduce the processing time. Second, we use two feature selection metrics to develop an adaptive K-Medoids method, called feature-based K-Medoids (FKM), which can cluster the image dataset faster by the feature selection mechanism. Third, our method surely reduces the processing time of mosaic images by the codebook. Though the image quality by our method is slightly lower compared with Szul et al.'s method, our method retains an acceptable image quality.

Keywords: Big data, Clustering, Digital image, Feature selection, K-Medoids, Mosaic image

1 Introduction

High-quality images can be rapidly generated by smartphones and cameras every minute. Image datasets have grown significantly. A computer perceives an image as non-structured, highly complex data, and thus, significant computer power is needed to process images. The distribution system called Hadoop [1] is a suitable solution to this issue. The users can develop their own distributed applications on Hadoop and processing big data even if they do not know the bottom-level details of the system. Many Hadoop applications are performed in the high-performed and high-availabilitycomputing and information systmes.

Processing a large volume of image datasets requires high computer power and powerful image processing techniques. Image features can be as contents for image retrieval [2], image detection [3], image hiding [4], etc. Therefore, effective and accurate image feature extraction [5] will be useful for many practical applications.

Our research focuses on the artwork of mosaic image.

The first step in creating an artwork mosaic image is to split the source image into several small cells. Then, the image in the image dataset is cropped or resized to match the small cells of the source image. Finally, the algorithm finds the best matches from the image dataset for each cell of the source image and then outputs the mosaic image.

In 2014, Szul and Bednarz proposed the use of Scalding to improve the productivity of mosaic images and the implementation [6] on Hadoop. Their experimental result achieved significant improvements. However, their proposed method is highly complex. Thus, we propose a method to reduce the computation complexity. This method preprocesses the original image dataset with the use of an unsupervised clustering algorithm called K-Medoids [7], which partitions data around their medoids. The preprocessing step takes time to cluster the image dataset. Thus we replace the Euclidean distance as the metric in the K-Medoids algorithm by two feature selection mechanisms, pixel difference (PD) and pixel coefficient of variation (PCV), to reduce the computing complexity. Our method further reduces the training cost.

Our study has two objectives. First, the method proposed by Szul et al. [6] improved the efficiency of constructing the mosaic image. We want to improve the productivity further by using the codebook generated by the K-Medoids algorithm. Second, we aim to reduce the high computational complexity of K-Medoids by using feature selection mechanisms.

*Corresponding Author: Jau-Ji Shen; E-mail: jjshen@nchu.edu.tw
DOI: 10.3966/160792642019092005003

2 Related Works

In 2014, Szul and Bednarz proposed a method [6] using Scalding to improve image processing algorithm, which avoids the time consuming process to write massive amount of images during the output of the map phase. The method not only conserves the processing time but also shortens the lines of code and makes the source code more readable. In image processing, there is still scope for improvement. In this study, we combined the K-Medoids algorithm [7] to increase the efficiency. A mosaic image is created based on the developed feature-based K-Medoids (FKM for short) method with feature selection mechanism using a Hadoop framework in a more efficient way.

2.1 MapReduce Framework

MapReduce framework is one of important part of Hadoop and also is the key role that managing the computing power and distribute the job properly. The first generation MapReduce framework came with Hadoop version 0.20 series [1] which is inherently parallel to put very large-scale data analysis into practice as demonstrated as Figure 1.

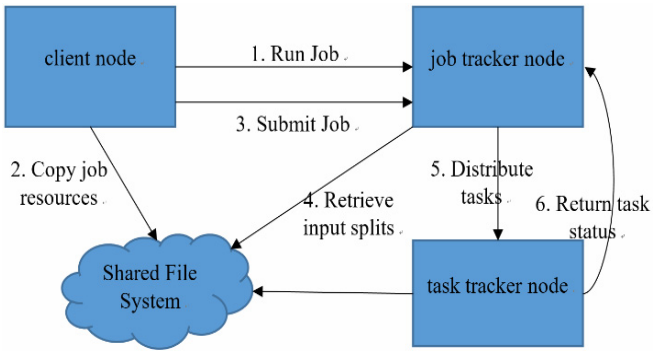


Figure 1. Workflow of Hadoop

Step 1. The MapReduce jobs start from the client node to the job tracker node. The client will perform a series check (input and output paths exist or not, for example) and compute the input split. Finally, job tracker node returns the ID for the job to the client node.

Step 2. The client nodes copy all the necessary resources to the shared file system (e.g., HDFS). The resources including job JAR file and input splits which the job tracker just computed.

Step 3. The client jobs submit the job to job tracker node which means the job is ready to perform.

Step 4. After receive the call of submit jobs, the job tracker creates equivalent map tasks to the numbers of input splits computed by client node and several reduce tasks. The number of reduce tasks is decided by mapred.reduce.tasks.

Step 5. The job tracker will distribute the job to the task tracker.

Step 6. The task tracker reports the task status to the job tracker by using heartbeat mechanism.

2.2 Scalding Mosaic Image

When the image dataset becomes increasingly larger or the resolution of the source image gets higher, the data will cause some problem during sharing, storing, analyzing, and capturing. Thus, the concept of big data is coming up. Hadoop is the most popular cloud computing technology with high scalability. MapReduce is a flexible framework [8-12] that can enhance the computing capability by scaling up the cluster. The MapReduce framework splits the job into several tasks, distributes those tasks to the work nodes of the cluster in the Map phase, collects the results of the tasks from the work nodes, and combines them in the Reduce phase.

It requires two MapReduce jobs to process mosaic image using Hadoop API. First, MapReduce job divides the source image into several non-overlap blocks with same size. Second, MapReduce is responsible for calculating the similarity between the blocks of the source image and all the images in the image dataset. In the map phase, it will generate a heat map after calculating the similarity between each image belongs to the image dataset and all blocks of the source image. The heat map shows the similarities and also indicates the best matches corresponding to the blocks. For example, let us assume there is a $W \times H$ source image, then divide it into several cell images, each size is $q \times q$. There are $W_q \times H_q$ cell images, for which the similarity with every image in the image dataset needs to be calculated. If there are $W_q \times H_q$ images with sizes $q \times q$ in the image dataset, we will get $W_q \times H_q$ heat maps after processing the calculations. Each heat map shows the best match image in the image dataset for the cell image. Next, we can construct the mosaic image by $W_q \times H_q$ heat maps.

However, the method we just described is not a very efficient one. Because of the processing of MapReduce framework, it needs to copy all blocks of source image and image to the local disk drive. It consumes much time to read and write on the local hard disk. To improve the productivity, Szul and Bednarz proposed a method using Scalding in 2014. By sending the flags of images instead of the images, the new method conserves a lot of time. The method not only improves the efficiency but also adds some constraints to filter the specific images. There are some benefits if the MapReduce job is developed by Scalding, such as conserving the executing time, adding constraints, readable, and streamlined source code. The algorithm proposed by Szul and Bednarz has two steps, BestMatchesSelection Step and Join Step as shown in Figure 2. In the BestMatchesSelection step, the grayscale images are first filtered out and then the

distance between the images in the image dataset and the blocks of the source image is calculated. After the calculation, the closest id of image in the image dataset of each block is output to the Join Step. In the Join Step, the images are searched according to their ids, which are just received in the image dataset, and the most similar image is outputted to the corresponding blocks.

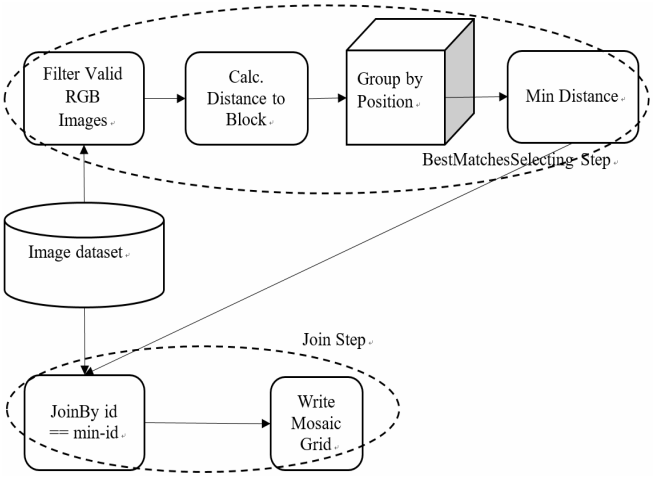


Figure 2. Method proposed by Szul et al. [6]

2.3 K-Medoids

K-Medoids is one kind of clustering algorithm. In each cluster, the algorithm will swap the non-medoid elements as medoid and calculate the summation of distance to each non-medoid element in the cluster to find which non-medoid element has the smallest summation of distance. Then re-distribute the all the elements to the cluster and perform the medoidshift algorithm again. Until the medoids are not changing which means the clusters are stable. The detail description of K-Medoids algorithm is shown as follows [7]:

Step 1. Initializing: Decide how many clusters the algorithm will generate.

Step 2. Clustering: Calculate the Euclidean distance of each data and medoid, and distribute the data to the closest cluster. And calculate the Total Cost by calculate the summation of distance each non-medoid element to the corresponding medoid of cluster.

$$Total\ Cost = \sum_{i=1}^{n} C_i \tag{1}$$

$$\begin{cases} O_i \in Non\ Medoids\ set \\ M_i \in Corresponding\ Medoids\ set \\ C_i = |\, O_i - M_i \,| \end{cases} \tag{2}$$

Step 3. Adjusting: Swap every medoid by non-medoid elements, and calculate its total cost for current configuration which refers to Step 2. Compare the total costs of current configuration and previous configuration to find out which better configuration which has smaller total cost.

Step 4. Inspecting: Keep repeating Steps 2 and 3 until the clustering configuration does not change anymore.

3 Proposed Method

This paper first presents an overview of the framework in the Figure 3, and then describes in detail how the FKM method and mosaic image processing are combined. The original K-Medoids algorithm has a high computational complexity because of two reasons: it uses Euclidean distance to judge which element in the dataset should be distributed to which cluster, and an image is an unstructured high-dimensional data. Two metrics for an image, namely, PD and PCV, which can measure the similarity of images, are used to replace the Euclidean distance and reduce the computational complexity. Second, we elaborate these two metrics. This paper also provides the pseudocode of feature-based K-Medoids and mosaic image algorithm.

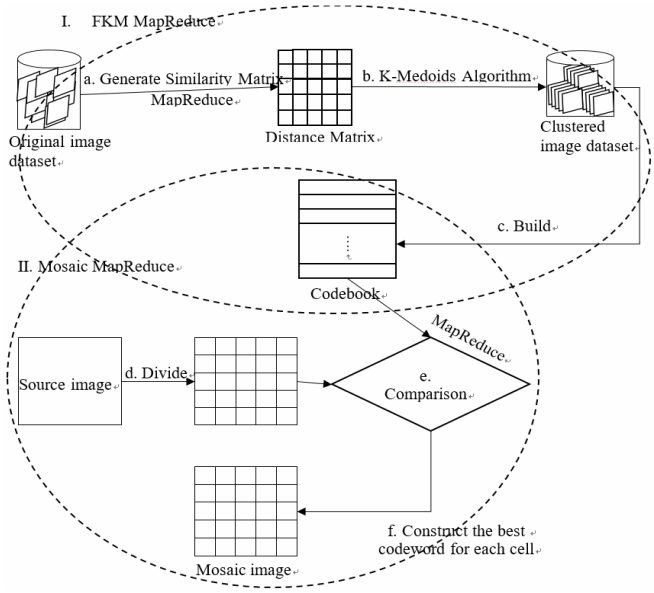


Figure 3. Framework of proposed method

3.1 Framework of Proposed Method

We propose a KFM method exploiting feature selection metrics on a Hadoop framework. Two MapReduce jobs exist in our proposed method, namely, FKM MapReduce and Mosaic MapReduce, which are performed sequentially. K-Medoids MapReduce is performed first to build the codebook, which can reduce the time consumed by Mosaic MapReduce job. Our proposed method focuses on the improvement of K-Medoids MapReduce in terms of reducing training cost and improving the codebook that can evaluate the image quality of the mosaic image.

In the proposed FKM algorithm, two different feature selection mechanisms are designed

with/without threshold parameter respectively. The first mechanism is named as Pixel Difference (PD). First step, PD will partition a given image into blocks. Next, PD calculates the pixel differences within the block and applies a predefined threshold to determine if the pixel differences can be a feature block. Only a feature block can be stored into a feature set. Each image associates with a feature set FS_{PD}. Afterwards, every two feature sets can be manipulated by an intersect operation to obtain a result set and the number of elements in the result set will be kept into a similarity matrix. A similarity matrix, *SM*, is an *N* by *N* matrix containing all the pairwise similarities between the objects being considered. PD approach is simple; however, it needs a proper threshold. So we derive another mechanism called pixel coefficient of variation, PCV for short. PCV is more generally than pixel difference for any case. PCV approach will calculate the coefficient of variation for each block. So each block has one coefficient of variation. In this approach, we will store the average pixel value of the block and the coefficient of variation in the feature set. Each image also corresponding to a feature set FS_{PCV}. Besides, PCV approach uses a range table $RT = \{R_r = [l_r, u_r] \mid r = 1, 2, ..., n\}$ to measure the similarity of two images by judging the coefficients of variation of two different sets of image blocks are in the same interval or not. The similarity is defined as the number of blocks from two images falling in the same interval.

Please refer to the K-Medoids MapReduce part in the Figure 3. K-Medoids MapReduce consists of three steps, as shown in the Figure 3. In the first step, **Step a**, the similarity matrix of each image in the image dataset to the other images in the image dataset is generated. **Step b**, MapReduce uses the distance table to perform the K-Medoids algorithm. Third, after the K-Medoids algorithm is completed, MapReduce builds the codebook to the Hadoop Distributed File System (HDFS) in **Step c**.

3.2 Feature-based K-Medoids (FKM) Process

The diagram shows the process of K-Medoids as shown in Figure 4. Initially, the original dataset is clustered. The proposed method preprocesses the original image dataset to downsize the original dataset to the codebook. Then, the codebook is used to compose the mosaic image; this approach is faster than Szul *et al.*'s method. To avoid recomputing the same distance repeatedly, the proposed method computes the similarity between each image and the others in the original dataset, and then it fills the similarity in the similarity matrix according to its corresponding index. The index is given when the entire dataset has been read during the initial process of the feature-based K-Medoids (FKM) process and then stored in the HashMap in the form of a key-value pair, and when the distances on the diagonal are zeros, which indicates the distance from the image to itself. After the distance table is generated, K-Medoids begins to cluster the image dataset by using the distance table. Finally, at the end of the process, a codebook is built. Next, we are going to introduce the FKM algorithm. For the sake of convenience, all the notations are listed in the Table 1.

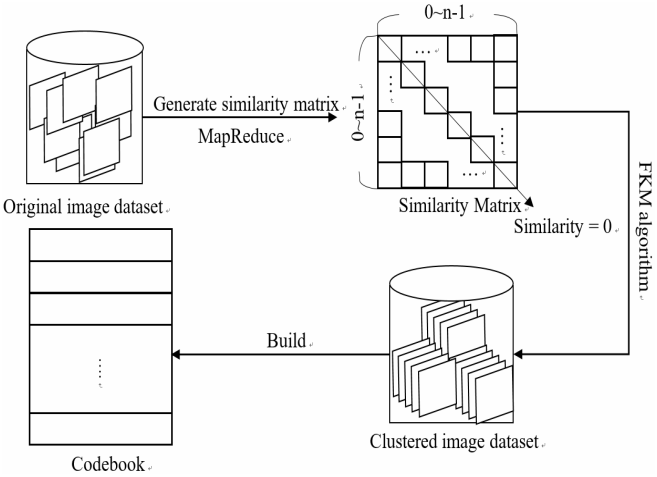


Figure 4. Process of FKM MapReduce

Table 1. Notations for FKM algorithm

Notations	Description
I	Image
SM	Similarity Matrix
M	Medoids
O	Non-Medoids
DT	Similarity matrix
CB	Built codebook
$TOTAL\_nCost_{prev}$	Total number of *Cost* of previous configuration
$TOTAL\_nCost_{cur}$	Total number of *Cost* of current configuration

3.2.1 Procedure Feature-based K-Medoids Algorithm

Input: similarity matrix SM

Initial: K medoids M where $M \in \{1, 2, ..., K\}$, *and the set of non-medoids* $O = DB_{FS} - M$. *Set* $TOTAL\_nCost_{prev}$ *and* $TOTAL\_nCost_{prev}$ *as zeros.*

Step 1: Calculate the Euclidean distance between each image in DB_{FS} *to the others in* DB_{FS} *and fill the result in the similarity matrix DT by the corresponding indices.*

Step 2: Distribute all the non-medoid images to the closest cluster by looking up the similarity matrix DT and calculate the summation of distance for each cluster. Let the summation denote as $TOTAL\_nCost_{prev}$.

Step 3: Swap each m in M with each non-medoids o_i *in O and calculate the* $TOTAL\_nCost_{prev}$.

Step 4: If $TOTAL\_nCost_{cur} - TOTAL\_nCost_{prev} > 0$ *then configure the medoids.*
Step 5: Repeat the Step 2 and Step 4 until all the $TOTAL\_nCost_{cur}$ *of each permutation combination of medoids and non-medoids.*
Step 6: Let codebook denote as CB. Set all the medoids to the CB then return CB.

The following K-Medoids MapReduce procedure contains tree classes to expound how the MapReduce job of K-Medoids works.

3.2.2 Procedure Feature-based K-Medoids MapReduce

• ***The Driver Phase:***

1. **Initialize similarity matrix *SM*.**
2. **Initialize HashMap *Medoids* for medoids.**
3. **Load the input image dataset *DS*.**
4. **Store each image in the *DS* into HashMaps *HM*. (Key: Index of image, Value: pixels of image)**
5.
6. **Perform feature selection mechanism by PD or PCV for each image *I* in the *HM*.**
7. **Replace the value of each key-value pair by the feature. (Key: Index of image, Value: features of image)**
8. **Input *HM* to the Map Phase to calculate the image to the others.**
9. **Collect the sequence file of result from the Reduce Phase and fill all similarities in the similarity matrix *SM* by the corresponding index.**
10. **Start K-Medoids algorithm to find the medoids for each cluster and store in the *Medoids* .**
11. **Return the *Medoids* as codebook *CB* to the Hadoop Distributed File System.**
12. **End procedure**

• ***The Map Phase:***

13. **Initialize HashMap HM_{input} for input data.**
14. **Load the input image dataset *DS*.**
15. **Initialize HashMap HM_{DS} for image dataset.**
16. **for each HM_{DS}**
17. **for each HM_{Input}**
18. **Calculate the similarity for the element in HM_{DS} and HM_{input} .**
19. **Emit the result to the Reduce Phase.**
20. **end for**
21. **end for**

• ***The Reduce Phase:***

1. **Initialize the HashMap HM for the results collecting from the Map Phase.**
2. **Extract the result and write in the sequence file of result.**
3. **Output the sequence file to the Hadoop Distributed File System.**

In order to reduce the computing complexity of K-Medoids MapReduce job, we proposed two metrics, pixel difference (PD) and pixel coefficient of variation (PCV), to evaluate the similarity of two images with the image feature selection. The result of evaluation can be filled into the similarity matrix and instead of the Euclidean distance. The following two feature selection mechanisms are proposed using pixel difference (PD) and pixel coefficient of variation (PCV) metrics.

Pixel difference (PD) feature-based mechanism. This PD function performs in a block-wise manner. First, PD divides an image into a series of non-overlapping block with 1×3 pixels in a block. Then PD calculates two difference values d_{i1} and d_{i2} for each block *i* according to Formula 3.

$$\begin{cases} d_{i1} = | p_{i2} - p_{i1} | \\ d_{i2} = | p_{i2} - p_{i3} | \end{cases} \quad \textbf{(3)}$$

The differences are checked if they are both greater than a predetermined threshold value *TH*. If the condition was hold, PD stores the block index *i* into the image feature set *FS*. Repeat to deal with all blocks of a given image and the process of Image Feature Selection Mechanism based on PD metric is depicted in Figure 5.

The primary purpose of PD mechanism is to compute an effective similarity measure between any input pair of images, say I_i and I_j. Let image feature sets $FS(I_i)$ and $FS(I_j)$ correspond to the image pair I_i and I_j, respectively. According to and $FS(I_j)$, we calculate the similarity matrix *SM*. Each matrix element can be expressed as $SM(I_i, I_j)$ which can be calculated from the number of an intersection of two feature sets $FS(I_i)$ and $FS(I_j)$. With the similarity matrix *SM* we can apply the feature-based K-Medoids algorithm to train a good codebook for clustering images to reach a higher performance in big data photomosaic computation on Hadoop-based Framework.

The process of Image Feature Selection Mechanism based on PD metric is clearly depicted in Figure 5.

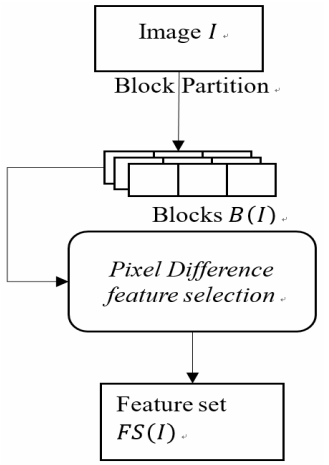


Figure 5. PD feature selection process

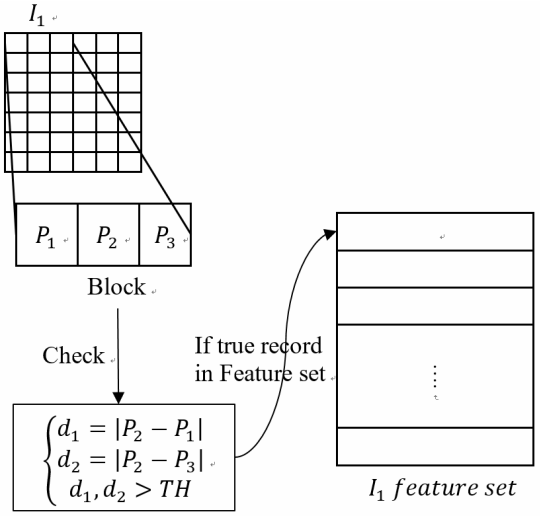


Figure 6. PD feature selection

Refer to Figure 6, The first scanned block in image I_1 contains three pixels p_1, p_2 and p_3, respectively.

Example 3.1:

Please refer to Figure 6 and Figure 7. A simple example is illustrated for this Pixel difference (PD) feature selection mechanism. Each image contains 3 blocks. In this case, we set the threshold to 70. Each block differences are calculated as follows, where

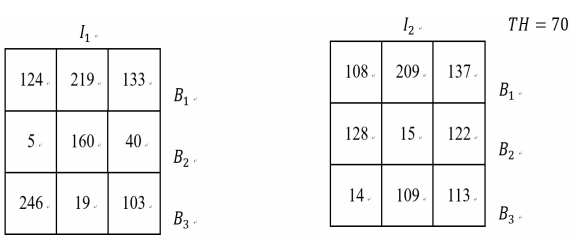


Figure 7. PD example

$$(d_{11}, d_{12}) = (95, 96), (d_{11}, d_{12}) = (155, 120),$$
$$\text{and } (d_{11}, d_{12}) = (227, 84).$$

Since every pair of block differences are larger than *TH*=70, the indices of blocks B_1, B_2, and B_3 are put into the image feature set $FS(I_1)$ of image I_1, i.e., $FS(I_1) = \{1, 2, 3\}$. Similarly, $FS(I_2) = \{1, 2\}$. We perform a set intersection on two feature sets and get a similarity value which is 2 because the number of $FS(I_1) \cap FS(I_2) = \{1, 2\}$ is two.

Pixel coefficient of variation (PCV) feature-based mechanism. The following we proposed another feature selection metric called pixel coefficient of variation, also called PCV for short. PCV is proposed for parameter-less purpose. Generally, it is hard to predetermine a proper threshold. Unlike PD mechanism, the advantage of PCV does not need to set a threshold for the image. First, PCV calculates the average and the coefficient of variation for every block of a given image. Coefficient of variation (CV) is a statistical measure of the dispersion of data points in a data series around the mean. It is a measure of relative variability and is the ratio of the standard deviation to the mean (average).

For each block k with three pixels of a grayscale image I_i. Let $\bar{p}_{ik}$ and θ_{ik} denote for the average of pixels and the coefficient of variation of block k. $\bar{p}_{ik}$ and θ_{ik} will be stored in a feature set $FS(I_i)$ corresponding to image I_i. $FS(I_i)$ is an ordered set, and is defined as follows. $FS(I_i) = [(\bar{p}_{ik}, \theta_{ik}) \mid 1 \le k \le \frac{W \times H}{m}]$, W and H stand for image width and height, and m denotes the number pixels in each block.

According to the research, human visual system is hard to tell the pixel differences which is smaller than 8. That means if the difference $d_{ijk} = \bar{p}_{ik} - \bar{p}_{ik}$, $1 \le k \le m$, is not greater than 8, the difference was hard to tell by human visual system.

Besides, PCV approach predetermined a range table $RT = \{R_r = [l_r, u_r] \mid l_r, < u_r, r = 1, 2, ..., n\}$, where $w_r = u_r - l_r$ and w_r is defined as follows. $w_r = \frac{\sqrt{m-1}}{n}$, here m denotes the number pixels in each block. In our example, there are three pixels in each block, thus $m = 3$. So, the value of coefficient is within the range of [0, 1.414]. We choose n=10 and have $w_r = 0.14$ in the experiments. The first condition is checking the average of three pixels, and the second condition needs satisfy the coefficient of variation θ_{ik} and θ_{jk} fall within the same R_r, i.e., $w_r \le |\theta_{jk} - \theta_{jk}|$.

Thus, the similarity of the two images is measured by whether the coefficient of variation of two different blocks falls within the same interval. The similarity is determined from the total number of blocks in which both images fall in the same interval. That means the similarity is the total number of blocks from two images which are falling in the same interval. The similarity value between images I_1 and I_j is set as one,

i.e., $SM(i,j)=1$ otherwise, $SM(i,j)=0$. Repeat to deal with all the blocks and the process of Image Feature Selection Mechanism based on PCV metric is depicted in Figure 8 and Figure 9.

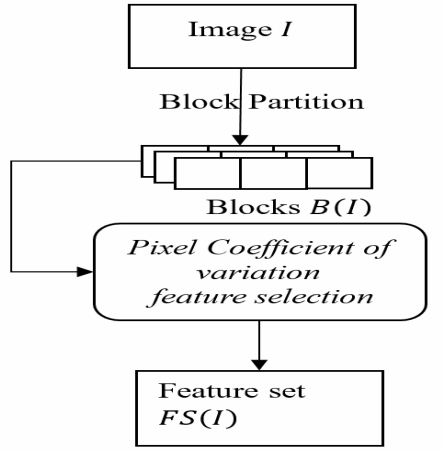


Figure 8. PVC feature selection process

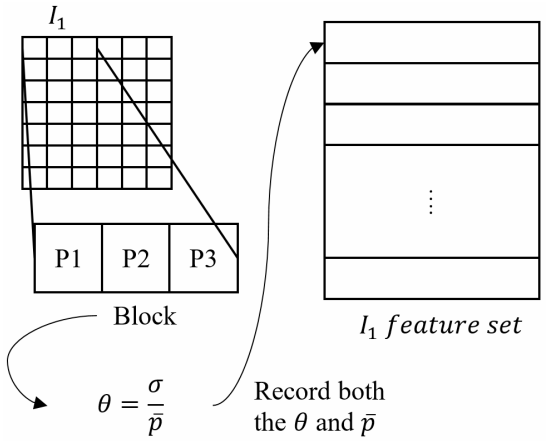


Figure 9. PVC feature selection

Example 3.2:

Two given grayscale images I_1 and I_2 are shown in Figure 10. According to the PCV feature selection mechanism, two feature sets $FS(I_1)$ and $FS(I_2)$ are as follows.

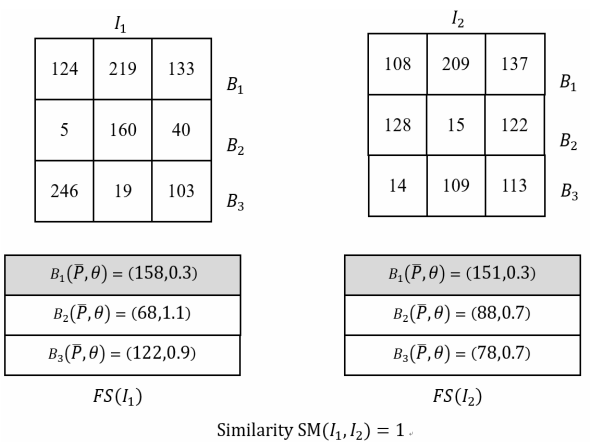


Figure 10. PCV example

$FS(I_1)=[(158,0.3),(68,1.1),(122,0.9),$ and
$FS(I_2)=[(151,0.3),(88,0.7),(78,0.7)]$.

Afterward, the differences of average pixels $d_{12k}=\bar{p}_{1k}-\bar{p}_{2k}$ are calculated. Only the difference 158-151=7 less than 8, so the mechanism of PCV further examines if both coefficients of variation falling within the same interval. Since $FS(I_1)\cdot\theta_1=FS(I_2)\cdot\theta_1=0.3$, the first blocks corresponding two images have high similarity. Repeat to examine the remainder two feature elements in $FS(I_1)$ and I_2, and we have the similarity value $SM(1, 2)=1$ corresponding to two images.

3.3 Mosaic MapReduce Job

In the process of mosaic image, Figure 11, it will divide the source image into several small pieces of cells and the sizes of all the cells are the same. Then the mosaic comparison MapReduce job will respectively find the best match image in the codebook for each cell. At the end of comparison MapReduce job, it will output the index of the best match cell for each cell of source image to compose the mosaic image. The following procedure will elaborate the Mosaic MapReduce job in detail.

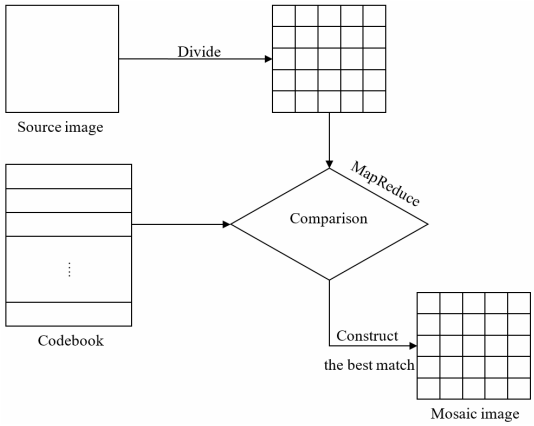


Figure 11. Process of mosaic image

3.3.1 Procedure Moasic MapReduce

- ***The Driver Phase:***

1. **Initialize codebook *CB*.**
2. **Load source image *SI*.**
3. **Split *SI* into several cell images and store each cell image into HashMap *CI*.**
4. **Start Map Phase and use *CI* as input.**
5. **Receive the distance between each codeword in the codebook and the *CI*.**
6. **Start Reduce Phase.**
7. **Receive the mosaic image and output to the Hadoop Distributed File System.**

• ***The Map Phase:***

1. **Load codebook CB.**
2. **for each HM_{input}**
3. **Calculate the distance between each codeword in the *CB* for each element in HM_{input}.**
4. **Emit the result of distance.**
5. **end for**

• ***The Reduce Phase:***

1. **Load codebook *CB*.**
2. **Find the best match codeword for each cell image by the id of cell image.**
3. **Compose the mosaic image by the codebook *CB*.**
4. **Emit the mosaic image to the Hadoop Distributed File System.**

4 Experimental Results and Discussions

In this chapter, we first compare the training cost of feature-based K-Medoids with different sizes of dataset. In our experiment, we prepare four different sizes of image datasets collected from Flickr. The size of each color image in our datasets is 75×75. In the second part of our experiment, the image quality of the mosaic image made up of six different source images is presented. Three of the source images were obtained from the research of Szul *et al.*, and the others were selected from Flickr which is an image- and video-hosting website and web services suite. The other three source images were selected to test the method under different conditions. Few articles have been written about this matter. Thus, we compare the training time and mosaic image quality with the results obtained by Szul *et al.* and with the two methods that use PD and PCV.

The environment of our experiment is based on VMware, and virtual machine is available on Cloudera's official website. The version of our virtual machine is CDH 5.5.0, and the version of Hadoop is 2.6.0. The hardware specifications of our virtual machine are shown as Table 2.

Table 2. Hardware specification

Component	Specification
Memory	8GB
Processors	4
Hard Disk	WD Black label 1TB

The first section of this chapter reveals the consumed training time and the time consumed to construct the mosaic image. The main contribution of this research is the reduction of the time consumed when constructing a mosaic image. The second section compares the image quality. The method proposed by Szul *et al.* composes the best-quality mosaic image by using an exhaustive search of the image dataset. Our research aims to reduce the time of constructing the mosaic image and to retain the good quality of the mosaic image.

4.1 Time Comparison

Three datasets, with different quantity of images, are "Tiny," "Small," and "Medium." The quantity of images of each dataset and the size are clearly note in the Table 3.

Table 3. Dataset information

	Dataset Information		
	Tiny	Small	Medium
Image quantity	5679	27921	111645
Dataset size	11MB	319MB	1.3GB

The Figure 12 shows the training cost (Time) of different codebook sizes with different image datasets. In this experiment, we use Euclidean Distance as a metric to measure the image similarity. Each line with different color stands for different training time to generate a codebook corresponding to different image dataset. Blue line represents "Tiny" image dataset. Orange line stands for "Small" image dataset. Gray line indicates the experiment result of "Medium" dataset. In the Figure 12, we can found that codebook training time with "Medium" image dataset takes most time. In contrast, training codebook with "Tiny" dataset is the fastest. We can also find that all the curves for the training time of codeword sized from 64 to 1024 are not dramatically increases. For convenience, we will denote the codebooks with size of 1024, 512, 256 and 64 as codebook_1024, codebook_512, codebook_256 and codebook_64, respectively in the following discussion.

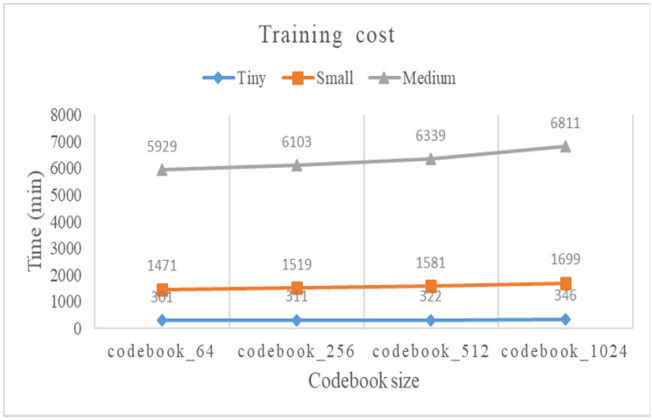


Figure 12. Training cost

Figure 13 shows the correlation between the training time and three metrics, which are Euclidean, PD, and PCV. We would like to observe which metric will reduce the most time of training cost. We pick the "Small" dataset as an experiment test set to see which method has the best improvement of training cost. Different color of lines in the Figure 13 stands for

different feature selection metrics, Euclidean Distance, PD and PVC, we used in the experiment. The blue line shows the performance of PD and result shows it has the best performance with smallest training time. The orange line and gray line demonstrate the result of PCV and Euclidean distance respectively.

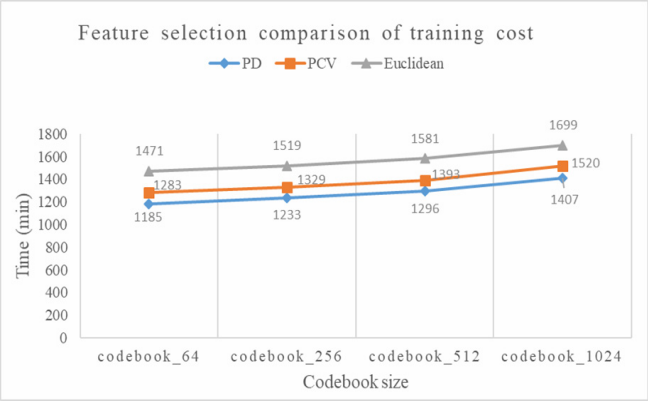


Figure 13. Feature selection metrics comparison of training cost

Figure 14 displays the time of consuming time of a mosaic image creation in different methods. Despite the image dataset will become bigger day by day, our method will cluster the image dataset into several groups. Thus the even the dataset gets bigger; however, our consuming time in creating a mosaic image does not change. Although the method proposed by the Szul et al. [6] can construct the best, but along with the image dataset getting bigger and bigger, the computing complexity becomes a serious problem.

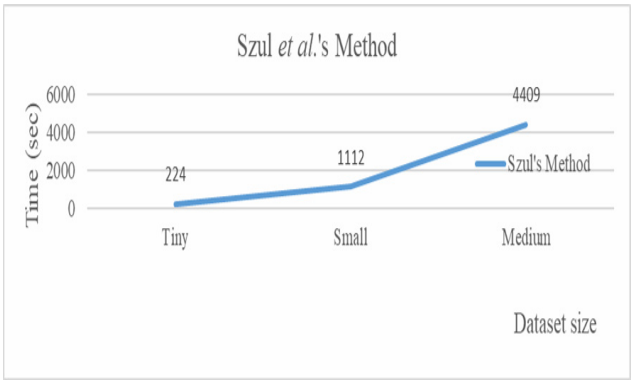


Figure 14. Szul et al.'s method

Figure 15 is the experiment result of constructing a mosaic image by using the proposed method in different codebook sizes. The consumed time in constructing a mosaic image increases with the size of the codebook. The most important advantage of the proposed method is that most of the time is transferred to the preprocessing step of the K-Medoids algorithm. Preprocessing can take place whenever the user does not require computing power or whenever the server does not need to deal with a heavy load from many users. Therefore, constructing a mosaic image does not require a considerable amount of time. Comparing the result to that of Szul et al.'s method [6] with small dataset, it takes 46 seconds to create a mosaic image by using the 1024-sized codebook.

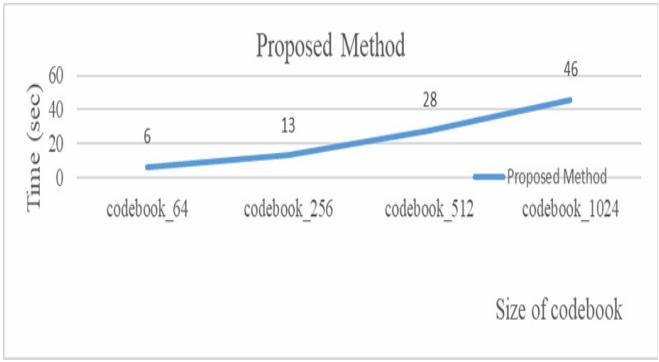


Figure 15. Performance of mosaic MapReduce in proposed method

It is easy to find out by the charts of the consuming time proposed by Szul et al.'s method and our proposed method in Figure 14 and Figure 15 respectively, In the Szul et al. method, it takes approximately 1112/27921 = 0.04 seconds to check whether each image in the dataset is similar to each cell image of the source image; However, our method takes only 46/27921=0.0016 second. Obviously, the time for constructing a mosaic image of our method is almost one twenty-fourth of Szul et al.'s result. It means our proposed method has a great progress in computing time on big data mosaic images.

This paper is inspired from the literatures with parallel K-Medoids algorithms and designs a feature-based K-Medoids (FKM) algorithm and integrates the algorithm into a framework for creating photo mosaics on Hadoop-based computing infrastructure.

Unlike the exhaustive search to find the best match for photographic mosaics, the feature-based K-Medoids (FKM) algorithm works well in reducing the time of constructing mosaic images and retains good quality of mosaic images.

4.2 Image Quality of Mosaic Image

Our experiment picks 6 images from Flickr website as source images and it is shown as follow. Firstly, 3 source images, Figure 16(a), Figure 20(b) and Figure 23(c), which are chosen as source images by Szul *et al.* and we also have these three images as test images in our experiment. In addition, we also choose another three images which are Figures 27(a), Figure 31(a) and Figure 35(a) in our experiment. These images all have significant image characteristics; for example, complex details, pixel distribution, brightness. Next we are going to observe the mosaic photos and give brief analysis for each image.

According to Figure 16(b), Green pixels are spread widely. Blue pixels are belongs to the dark tone. The amount of red pixels is not greater than blue pixels and green pixels. The next three butterfly mosaic images.

Figure 17 is generated by the Szul et al.'s method with different size of dataset. From left to right, there are Tiny, Small and Medium. In Figure 18, mosaic images are generated by the proposed method using Euclidean distance with different size of codebook. From left to right, there are Tiny, Small and Medium. Figures 19(a) and Figure 19(b) are the butterfly mosaic images generated by proposed method using PD and PCV mechanisms, respectively with Codebook_512 and the codebook is trained by the small-sized dataset.

(a) Source image - butterfly (b) Histogram of butterfly

Figure 16. Source image - butterfly

(a) Tiny - butterfly (b) Small - butterfly

(c) Medium - butterfly

Figure 17. Mosaic images created by Szul et al.'s method with different size of dataset

(a) codebook_256 -butterfly (b) codebook_512 - butterfly

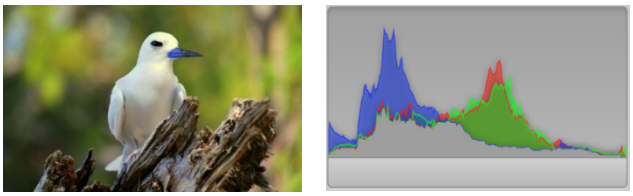
(c) codebook_1024 - butterfly

Figure 18. Mosaic images created by original K-Medoids with different codebook size

(a) PD - butterfly (b) PCV - butterfly

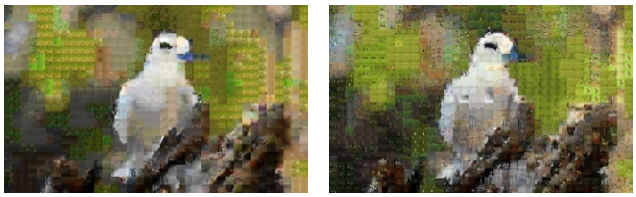
Figure 19. Mosaic images created by feature-based K-Medoids with two feature selection mechanisms

The next source image is dove, Figure 20. The blue pixels of the dove image are located in the dark tone which is similar to the image of butterfly. But the biggest difference between dove and butterfly is that the peaks of green pixels and red pixels are located in the midtones. Figure 21 demonstrates the mosaic image created by Szul et al.'s method with different size of dataset. In Figure 22, there are three mosaic image created by different codebook size which is training by original K-Medoids algorithm. In Figure 23 are two Mosaic images created by feature-based K-Medoids with two feature selection mechanisms.



(a) Source image - dove (b) Histogram of dove

Figure 20. Source image - dove

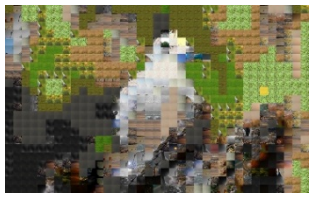


(a) Tiny - dove (b) Small - dove

(c) Medium - dove

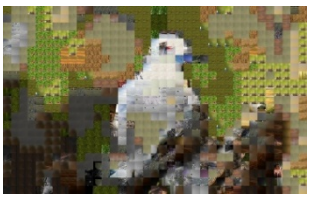
Figure 21. Mosaic image created by Szul et al.'s method with different size of dataset


(a) codebook_256 - dove


(b) codebook_512 - dove


(c) codebook_1024 - dove

Figure 22. Mosaic image created by original K-Medoids with different codebook size


(a) PD - dove

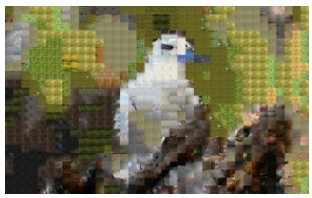

(b) PCV - dove

Figure 23. Mosaic images created by feature-based K-Medoids with two feature selection mechanisms

Figure 24 is face image which is the smoothest image of 6 source images. This image is pretty easy to judge the quality of codebook. The blue, green and red pixels are located in the highlight tone. The red pixels are the brightest pixels to the others. Figure 25 and Figure 26 show that the image qualities of mosaic images composed by smaller dataset and smaller codebook worse than the bigger one. Figure 27 are mosaic images created by feature-based K-Medoids with two feature selection mechanisms.


(a) Source image - face

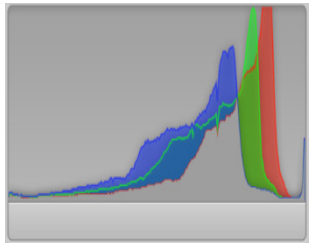
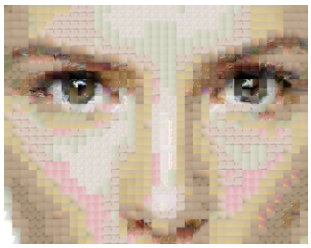
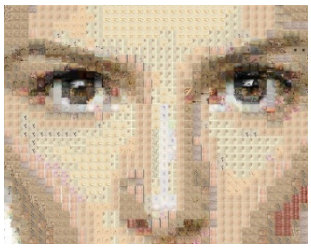

(b) PCV - dove

Figure 24. Source image - face


(a) Tiny - face


(b) Small - face

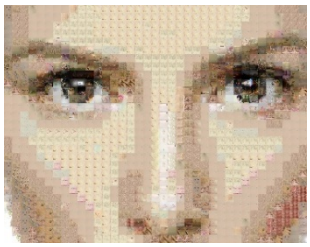
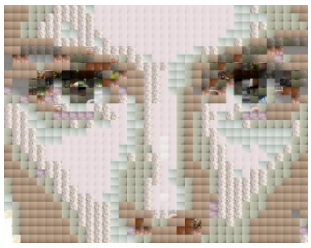
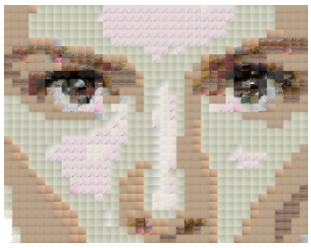

(c) Medium - face

Figure 25. Mosaic images created by Szul *et al.*'s method with different size of dataset


(a) codebook_256 - face


(b) codebook_512 - face

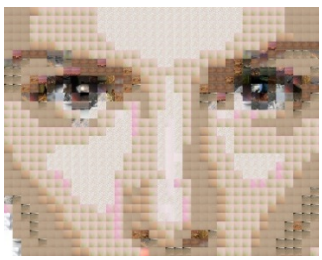
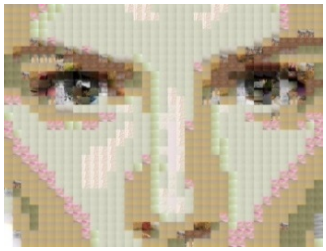

(c) codebook_1024 - face

Figure 26. Mosaic images created by original K-Medoids with different codebook size


(a) PD - face

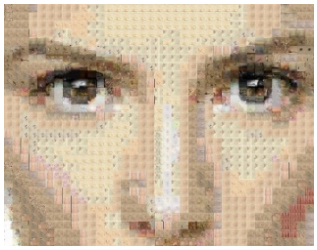
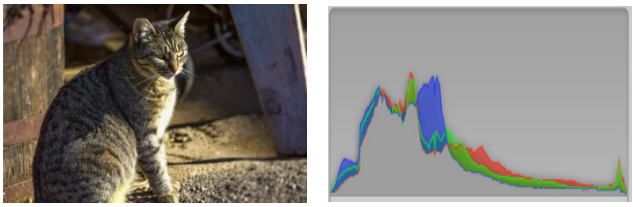

(b) PCV - face

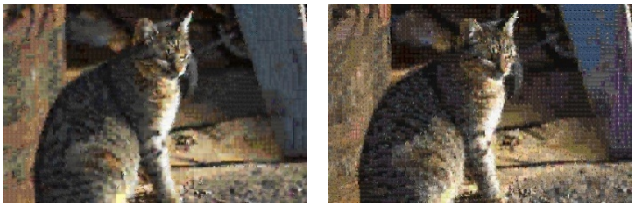
Figure 27. Mosaic images created by feature-based K-Medoids with two feature selection mechanisms

In the Figure 28(b), we found that the most of blue, green and red pixels are located in the zone of shadows. This source image is used to test the codebook whether can deal with dim image or not. There are also some complex details on the body of the cat. In Figure 29 and Figure 30, the cat image has really high complexity and each mosaic image show clear silhouette of cat. Bigger dataset or codebook size can deal with the detail of the cat's fur well. Figure 31 are Mosaic images created by feature-based K-Medoids with two feature selection mechanisms.



(a) Source image - cat (b) Histogram of cat

Figure 28. Source image – cat

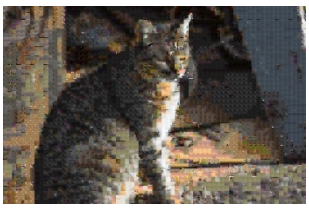


(a) Tiny - cat (b) Small - cat

(c) Medium - cat

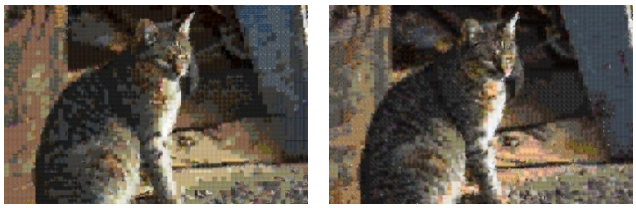
Figure 29. Mosaic images created by Szul et al.'s method with different size of dataset

(a) codebook_256 - cat (b) codebook_512 - cat



(c) codebook_1024 - cat

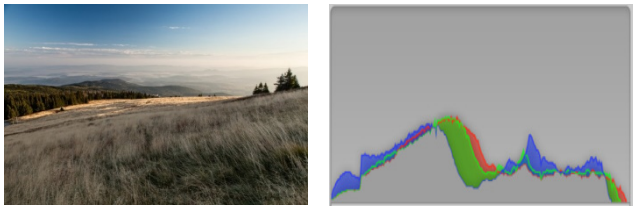
Figure 30. Mosaic images created by original K-Medoids with different codebook size



(a) PD - cat (b) PCV - cat

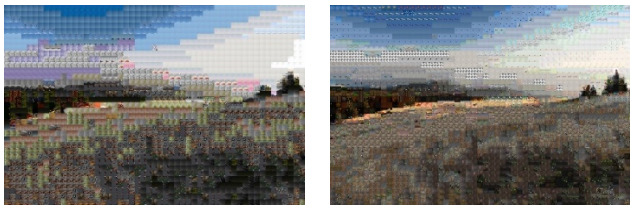
Figure 31. Mosaic images created by feature-based K-Medoids with two feature selection mechanisms

In the Figure 32(b), the pixels of each color, red, green and blue are averagely spreading. Each aspect is balance. The part of grass reveals abundant complicated details. Otherwise, the part of sky is really smooth. In Figure 33, Figure 34 and Figure 35. None of the method could handle the detail of grass well.



(a) Source image - grass (b) Histogram of grass

Figure 32. Source image – grass

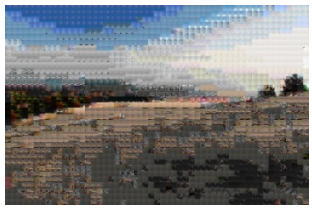


(a) Tiny - grass (b) Small - grass

(c) Medium - grass

Figure 33. Mosaic images created by Szul et al.'s method with different size of dataset

(a) codebook_256 - grass



(b) codebook_512 - grass

(c) codebook_1024 - grass

Figure 34. Mosaic images created by feature-based K-Medoids with two feature selection mechanisms

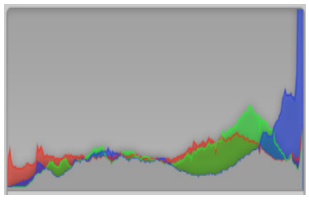
(a) PD - grass

(b) codebook_512 - grass

Figure 35. Mosaic images created by feature-based K-Medoids with two feature selection mechanisms

Here's where it gets interesting that in our brain along the three channels, Red, Green and Blue, seeing blue is what you experience when low-wavelength light excites the blue cones more than the green and red. In our algorithm, blue pixels are ignored due to the lower sensitivity of the human eye to the blue light [14], therefore making great reduction in the computing complexity. Apparently, when we choose Figure 36(a) as a source image which is a lake comprising more blue pixels as shown in Figure 36(b), the mosaic images created by feature-based K-Medoids with two feature selection mechanisms as seen in Figure 38 and Figure 39 are quite similar to that of Figure 37 created by Szul et al.'s method.

(a) Source image - lake



(b) Histogram of lake

Figure 36. Source image lake

(a) Tiny - lake

(b) Small - lake

(c) Medium - lake

Figure 37. Lake mosaic images created by Szul *et al.*'s method with different size of dataset

(a) codebook_256 - lake

(b) codebook_512 - lake

(c) codebook_1024 - lake

Figure 38. Lake mosaic images created by original K-Medoids with different codebook size

(a) PD - lake

(b) PCV - lake

Figure 39. Mosaic images created by feature-based K-Medoids with two feature selection mechanisms

The first experiment compares the mosaic image quality between Szul et al.'s method and the proposed method. It also aims to prove that the proposed method considerably reduced the time consumed in

constructing the mosaic image but is still able to retain a good mosaic image quality. This experiment uses the medium-sized image dataset for both K-Medoids and Szul *et al.*'s method. In this experiment, we aim to determine whether K-Medoids with Euclidean distance could preserve the image quality.

We use structural similarity, SSIM [13], to evaluate the quality of our proposed mosaic image. SSIM is a perception-based model to simulate human visual system to assess the similarity between two images. The SSIM value is calculated by the following Equation 4:

$$SSIM(x, y) = [l(x, y)]^{\alpha}[c(x, y)]^{\beta}[s(x, y)]^{\gamma} \quad \textbf{(4)}$$

$$where\ l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2\mu_y^2 + c_1}, c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2\sigma_y^2 + c_2} \text{ and}$$

$$s(x, y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

According to Figure 40, the mosaic image application has a great performance on smooth image. The following experiment is going to find out the best trade-off the training cost and quality of mosaic image. Larger size of codebook gives you better quality of mosaic image, but it cost more time to perform the K-Medoids algorithm. In Figure 40, the lines in yellow colors stand for the SSIM of Szul *et al.*'s method. Figure 40 also has the lines in gray color, orange color, and blue color stand for the proposed methods with codebook_1024, codebook_512, and codebook_256, respectively. Comparing the result of mosaic image quality (SSIM) of codebook_1024 to codebook_512, the largest difference in SSIM isn't greater than 0.09. It is obvious that the mosaic image quality generated by 512-sized codebook is pretty closed to 1024-sized codebook and it also has ideal training cost.

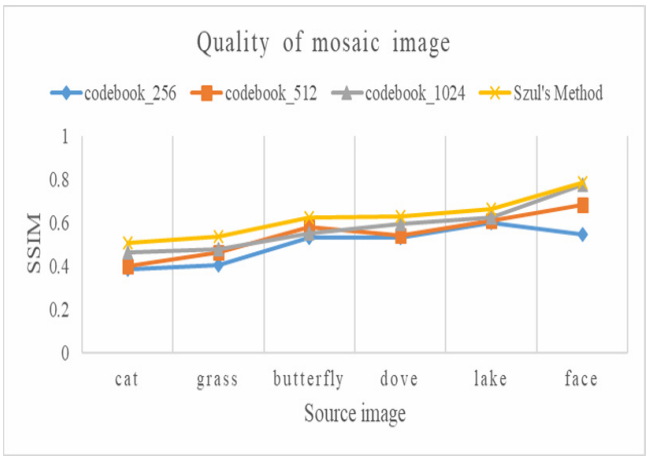


Figure 40. Quality of mosaic image

In the next experiment, Figure 41 shows the quality comparison between two new metrics, PD and PCV, adopt in K-Medoids. In our research, these two metrics are used to reduce the computing complexity of K-Medoids algorithm. In the comparison of consuming time between using different metrics, Euclidean distance, PD and PCV, respectively, we could find that PD and PCV absolutely conserve the training time. Let's compare the tradeoff between image quality and the codebook training cost. In Figure 41, we demonstrate the mosaic image quality by the proposed PCV feature selection mechanism since PCV feature selection mechanism has the best mosaic image quality in four source images, "dove," "face", "grass," and "lake." Their SSIM values are higher.

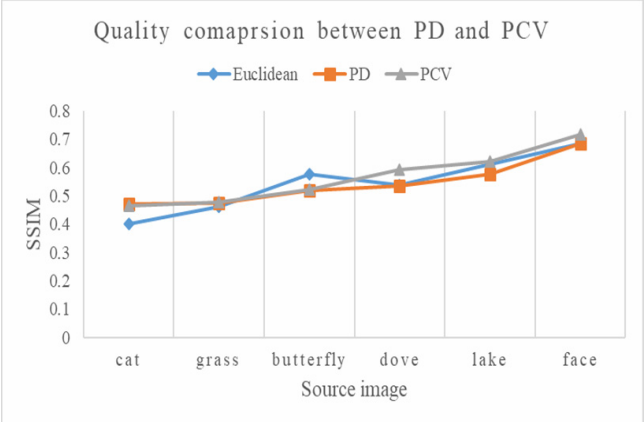


Figure 41. Quality comparison among various feature selection by Euclidean, PD and PCV

First refer to Figure 42, mosaic image quality and the codebook training cost is shown in the figure with line charts. The SSIM values of image "face" with the 256-sized codebook and 512-sized codebook are 0.5437 and 0.6843, respectively. The different of mosaic image quality of two different codebooks is 0.1406 in SSIM. We can use extra 62 minutes to train a 512-sized codebook for a better mosaic image quality. Let's compare the performance of 512-sized codebook and 1024-sized codebook. The training cost increases 118 minutes but the image quality only increases 0.0928.

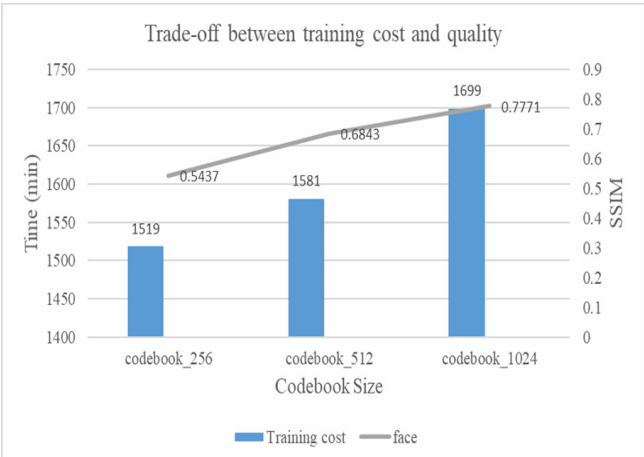


Figure 42. Trade-off between training cost and image quality for "face"

Figure 43 shows the charts of mosaic image quality with different databases. In this figure, we can see a bigger dataset like the medium dataset in our experiment does not imply it exactly deserve better mosaic image quality. According to Figure 43, the mosaic image quality of small dataset is better than that of the medium dataset. Both the mosaic images of tiny dataset and medium dataset could be seen the silhouette easily.

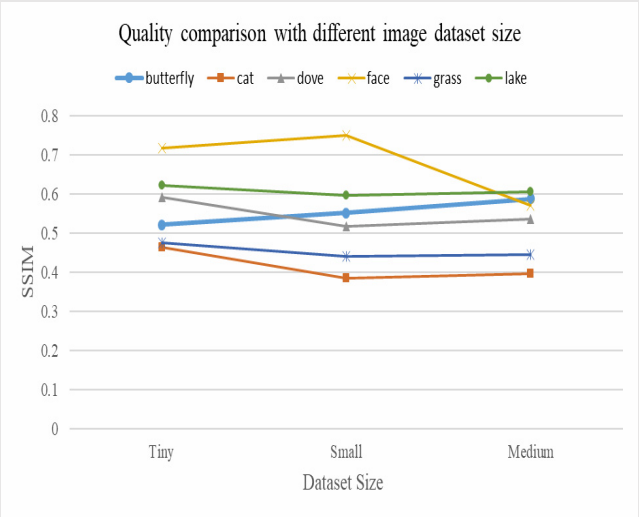


Figure 43. Quality comparison using different codebooks trained by different dataset sizes

In Figure 44, the charts demonstrate the mosaic image quality comparison of different thresholds of PD feature selection mechanism. The cases of smooth source images, face and lake, have better mosaic image quality with the codebook using smaller threshold. The source images with more details fit well with high threshold. If the threshold greater than 180, some clustering algorithm will end up with some errors on the configuration of clustering algorithm.

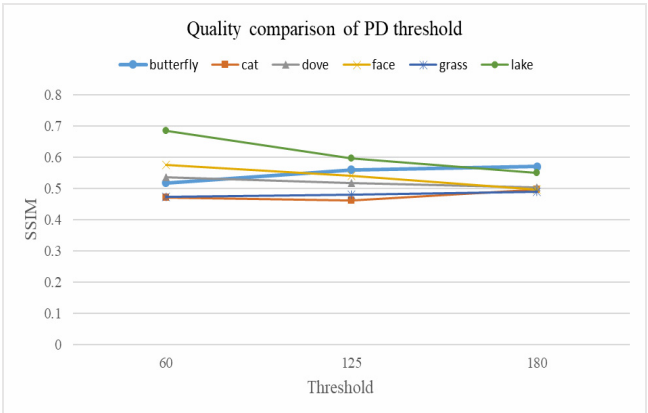


Figure 44. Quality comparison of PD threshold

5 Conclusion and Future Work

A photographic mosaics computing method upon a hadoop framework is proposed in this work. The mosaic technique has seldom been discovered in a hadoop environment except for the research of Szul *et al.* Based on the hadoop framework. The experimental results show that the method using the feature-based K-Medoids (FKM) and the mosaic image algorithms performs well in terms of image quality or processing performance. FKM requires considerable time to cluster an image dataset. Mosaic MapReduce can improve efficiency, thereby enhancing the user experience.

A smooth source image is needed to ensure a recognizable mosaic image. Also a smooth source image could be used to judge the quality of a codebook. A source image with high complexity does not provide a great effect on the mosaic image. The main contribution of this research is its use of the FKM algorithm to cluster an image dataset into several groups to reduce the time for constructing the mosaic image. Moreover, an acceptable mosaic image quality is achieved. Another key contribution is the use of PD and PCV, as well as the distance table, to reduce the training cost of the FKM algorithm. The mosaic image quality differed minimally, but both metrics reduce the computational complexity considerably. The experiment on the tradeoff between training cost and mosaic image quality indicates that the codebook with a size of 512 achieves the best balance of these two aspects.

This study has several future research directions. First, the FKM algorithm is implemented by Java and performed on a single PC. A significant improvement can be achieved if Mahout is used for implementation because it can fully upgrade the proposed framework in the Hadoop environment. Moreover, this approach can deal with more images and compose a larger mosaic image. Second, the proposed method aims to reduce the training cost and preserve and achieve an acceptable mosaic image quality. The use of different image processing technologies may eliminate the tradeoff between the training cost and the mosaic image quality. Third, the two metrics, PD and PCV, are still not simple enough to be implemented on different domains; this subject is another future research direction.

Acknowledgments

This research was partially supported by the Ministry of Science and Technology of the Republic of China under the Grants MOST106-2221-E-324-006 - MY2.

References

[1] T. White, *Hadoop: The Definitive Guide*, O'Reilly Media,

Inc., 4th ed., 2015.

[2] C. C. Chang, C. F. Lee, Relative Coordinates Oriented Symbolic String for Spatial Relationship Retrieval, *Pattern Recognition*, Vol. 28, No. 4, pp. 563-570, April, 1995.

[3] M. H. Aghdam, P. Kabiri, Feature Selection for Intrusion Detection System Using Ant Colony Optimization, *International Journal of Network Security*, Vol. 18, No. 3, pp. 420-432, May, 2016.

[4] J. J. Li, Y. H. Wu, C. F. Lee, C. C. Chang, Generalized PVO-K Embedding Technique for Reversible Data Hiding, *International Journal of Network Security*, Vol. 20, No. 1, pp. 65-77, January, 2018.

[5] S. P. Luttrell, An Adaptive Bayesian Network for Low-Level Image Processing, *1993 Third International Conference on Artificial Neural Networks*, Brighton, UK, 1993, pp. 61-65.

[6] P. Szul, T. Bednarz, Productivity Frameworks in Big Data Image Processing Computations - Creating Photographic Mosaics with Hadoop and Scalding, *Procedia Computer Science*, Vol. 29, pp. 2306-2314, June, 2014.

[7] L. Kaufman, P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley Series in Probability and Statistics, John Wiley & Sons, Inc., 2008.

[8] J. Wei, S. Wang, L. Zhang, A. Zhou, F. Yang, Virtual Machine Placement to Minimize Data Transmission Latency in MapReduce, *Journal of Internet Technology*, Vol. 18, No. 6, pp. 1379-1391, November, 2017.

[9] H. S. Kim, D. I. Shin, Y. J. Yu, H. Eom, H. Y. Yeom, Systematic Approach of Using Power Save Mode for Cloud Data Processing Service, *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 10, No. 2, pp. 63-73, July, 2012.

[10] K. Ma, B. Yang, Introducing Extreme Data Storage Middleware of Schema-Free Document Stores Using MapReduce, *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 20, No. 4, pp. 274-284, December, 2015.

[11] S. E. P. Hernandez, J. E. Ramirez, L. A. M. Rosales, G. R. Gomez, An Intermedia Synchronisation Mechanism for Multimedia Distributed Systems, *International Journal of Internet Protocol Technology*, Vol. 4, No. 3, pp. 207-218, January, 2009.

[12] V. G. Nguyen, A. Brunstrom, K. J. Grinnemo, J. Taheri, SDN Helps Velocity in Big Data, in: J. Taheri (Ed.), *Big Data and Software Defined Networks*, IET, 2018, pp. 207-227.

[13] Z. Wang, L. Lu, A. Bovik, Video Quality Assessment Based on Structural Distortion Measurement, *Signal Processing: Image Communication*, Vol. 19, No. 2, pp.121-132, February, 2004.

[14] T. Lamb, J. Bourriau, *Colour: Art and Science*, Cambridge University Press, Cambridge, UK, 1995.

Biographies

Chin-Feng Lee received her Ph.D. in Computer Science and Information Engineering from National Chung Cheng University, Taiwan in 1998. She is currently a professor of Information Management at Chaoyang University of Technology, Taiwan. Her research interests include steganography, image processing, information retrieval and data mining.

Jau-Ji Shen received his Ph.D. in Computer Science and Information Engineering from National Taiwan University, Taiwan in 1988. He is currently a professor of Information Management at Chung Hsing University, Taiwan. His research interests include image techniques, data techniques and software engineering.

Kun-Liang Hou received his master degree of Management Information Systems from National Chun Hsing University, Taiwan in 2016. He is currently a java developer who focus on developing backend software. He is specialized in Object-Oriented analysis & design, software architecture and pattern design. He has hands-on experience of software implementation.

Fang-Wei Hsu received his Bachelor of Business Administration (B.B.A), Department of Information Management from Tamkang University, Taiwan in 2017. He is currently a graduate student of Information Management at Chung Hsing University, Taiwan. His research interests include image techniques, and image authentication.