



# Image Recognition Using Unsupervised Learning Based Automatic Fuzzy Clustering Algorithm

V. V. Tai<sup>1</sup> and L. T. K. Ngoc<sup>2,3,4(✉)</sup>

<sup>1</sup> College of Natural Science, Can Tho University, Can Tho City, Vietnam  
vvtai@ctu.edu.vn

<sup>2</sup> Faculty of Engineering, Van Lang University, Ho Chi Minh City, Vietnam  
ngoc.1tk@vlu.edu.vn

<sup>3</sup> University of Science, Ho Chi Minh City, Vietnam

<sup>4</sup> Vietnam National University, Ho Chi Minh City, Vietnam

**Abstract.** This article proposes a novel techniques for unsupervised learning in image recognition using automatic fuzzy clustering algorithm (AFCA) for discrete data. There are two main stages in order to recognize images in this study. First of all, new technique is shown to extract sixty four textural features from  $n$  images represented by a matrix  $n \times 64$ . Afterwards, we use the proposed method based on Hausdorff distance to simultaneously determine the appropriate number of clusters. At the end of the unsupervised clustering process, discrete data belonging to the same cluster converge to the same position, which represents the cluster's center. After determining number of cluster, we have probability of assigning objects to the established clusters. The simulation result built by Matlab program shows the effectiveness of the proposed method using the corrected rand, the partition entropy, and the partition coefficients index. The experimental outcomes illustrate that the proposed method is better than the existing ones as Fuzzy C-mean. As a result, we believe that the proposed method is filled with a potential possibility which can be applied in practical realization.

**Keywords:** Automatic algorithm · Hausdorff distance · Image recognition · Fuzzy · Unsupervised clustering

## 1 Introduction

In recent years, unsupervised learning techniques have made a great contribution to machine learning that is used to find the common natural cluster structure of an unlabeled data set. The conventional supervised learning methods utilized in many fields, containing as data analysis, require a supervisor to guide the machine, labeling inputs with the outputs we want the machine to learn from. Nevertheless, this labeling process is complex stage, especially when a number of data to label are being created up to the billions of items every day, such as data images on the Internet. As a result, unsupervised

learning techniques are required to solve this limitation. In unsupervised learning, unsupervised clustering is an important data mining technique, used to divide data points into groups. Such those objects in the same group will have a high degree of similarity while objects belonging to the other ones will have a high degree of dissimilarity. It has been widely used in many applications such as data analysis, pattern recognition and image processing [1, 2].

Unsupervised item recognition plays a vital role in many fields such as artificial intelligence, architecture, and engineering. Although it has been concerned by many scientists in many areas for a long time, it is quite challenging at present. To overcome these challenges, there are a great deal of approaches presented by [3, 4] to recognize items through some algorithms. However, these approaches are usually process compoundedly, create inaccurate outcomes and take a huge amount of time to perform. Therefore, this paper shows a new method to image recognition using an automatic fuzzy clustering algorithm. There are two primary phases in order to recognize images in this study.

To begin with a first phase, human eye is a discerning item like a combination of primary parts (color, texture, shape). Therefore, color extraction, texture extraction and shape extraction are three main approaches in image feature extraction technique. We can choose the method which is appropriate to our goal as each approach has its advantages and disadvantages. In this paper, we concentrate on the texture extraction method. In image recognition, there are two main approaches that attracted by many researchers: signal processing approach [5–7] and statistical approach [8–10]. To compare with the signal processing approach, statistical image analysis based on the grey level co-occurrence matrix (GLCM) is simple to carry out and takes less computational time. The experimental outcomes in Sect. 4 illustrate that the proposed algorithm is better than the existing ones about running time and accuracy. In fact, the proposed method gives a high clustering performance and is applied by many researchers. For example, Clausi [11] utilized GLCM and Fisher method to group natural textures, Bhogle and Patil [12] combined GLCM and Mahalanobis distance to detect oil spill. It is well-known that there are two major steps in texture-based image recognition. In the first step of recognition, we compute the GLCMs from each image. GLCM is a two-dimensional matrix of joint probabilities between pairs of pixels, separated by a distance  $d$  in a given direction  $\theta$ . For example, Ayala [13] presented  $d = 2$  and  $\theta = \{0, 90^\circ, 180^\circ, 270^\circ\}$  while Celebi and Alpkocak [10] claimed that using *contrast* with  $d = \{1, 2, 3, 4\}$  is the best. In the final step, we extract the features from GLCMs. Haralick [8] defined 14 statistical features from gray-level co-occurrence matrix for texture clustering, where contrast, correlation, homogeneity and energy are common features since they have strongly affects on clustering result. In GLCM, we will collect  $x \times y \times z$  typical variables for texture (TVT) if  $x$ ,  $y$  and  $z$  are cases of distance  $d$ , direction  $\theta$  and extracted features for each GLCM, respectively.

In the second phase, we use the extracted features for clustering by the proposed method, called Automatic fuzzy clustering algorithm (AFCA). AFCA is a novel technique in unsupervised learning since it is built based on combining two techniques: the determination of the suitable number of cluster, and items belonging to the same cluster converge to the same position, which represents the cluster's center in the end of the

unsupervised clustering process. In the next step, we have the probability for assigning the items into the established clusters. As a consequent, the proposed method shows an outstanding effect rather than other ones through applied experiments. In our knowledge, most technique used in unsupervised clustering are fuzzy c-mean clustering (FCM) [3, 4] to recognize the image. However, FCM technique requires prior  $k$  clusters so the choice of the number of cluster,  $k$ , is based on experience of researchers. This choice leads to no guarantee of the accuracy of the clustering results. To solve this drawback, we proposed AFCA which can be seen as a new method in unsupervised clustering. There are two main reasons why we believe that AFCA is the novel technique. First of all, we proposed the vector of weight  $\lambda_i$  where  $\prod \lambda_i = 1$ ,  $i = 1, \dots, k$  utilized to measure the difference of central clusters. In addition, we also build the objective function  $f(U)$  which is improved to optimize the initial matrix partition  $U$ .

According to kind of objects, we have some common objects such as discrete elements [14] probability density functions [15–18], and interval data [19, 20]. Particularly, cluster analysis for discrete data (CAD) was studied first with a lot of announced results, which are related to both theory and application [14]. Furthermore, discrete data are becoming more and more popular in storing of data. Hence, we propose a novel technique for unsupervised clustering in image recognition using automatic fuzzy clustering algorithm (AFCA) for discrete data. Determining the criteria for evaluating the similarity between two objects and between two clusters in CAD, the commonly used distances in measuring the similarity between the two clusters are the min distance, the max distance, the average distance, and the Ward distance while the Euclidean distance ( $d_E$ ) the  $L_p$  distance, and the city-block distance ( $d_C$ ). They are the main criteria for evaluating the similarity among discrete data. In this paper, we use the Hausdorff distance ( $d_H$ ) as primary measurement to evaluate the difference of data. The experiment results present that the proposed algorithm used  $d_H$  is better than the other ones, where the corrected rand (CR), the partition entropy (PE), and the partition coefficients (PC) are indexes utilized to check effect of outcome clustering out. As a result, we believe that the proposed method is filled with a potential possibility which can be applied in practical realization.

The remainder of the paper is organized as follows. In Sect. 2, some distance measurements are defined for multi-dimension cases. In addition, clustering evaluation criteria also defined in this section. The proposed technique contains image extraction method, the proposed algorithm and its convergence given in Sect. 3. Section 4 illustrates the proposed technique by two applications in image recognition and compares with existing methods based on PC and PE index. Section 5 is the conclusion of the paper.

## 2 Some Distance Measurements for Discrete Data

### 2.1 Measure Distance Between Points

The Hausdorff Distance is commonly used in machine. In that field, a typical problem is that you are given an image and a model of what you want to match to. The goal is to find all the locations in the image which match the model. This is similar to the problem of matching protein motifs within protein sequences. This distance is different from some of the previously discussed measures, instead of forming a one-to-one mapping between the two, we allow a many-to-many correspondence in this case. Often times, it is easier

to build a many-to-many correspondence, since if we wish to change one assignment, we no longer have a cascade of other assignments which now also need to be redone.

**Definition 2.1.1.** Given two set  $X = \{x_1, x_2, \dots, x_n\}$  and  $Y = \{y_1, y_2, \dots, y_n\}$  in  $E^2$ . The distance from  $X$  to  $Y$  is defined as.

The Hausdorff distance

$$d_H(X, Y) = \sum_{i=1}^n (h_H(x_i, y_i), h_H(y_i, x_i))$$

Where

$$h_H(X, Y) = \sum_{i=1}^n \max_{x_i \in X} \min_{y_i \in Y} \|x_i - y_i\|$$

The Euclidean distance

$$d_E(X, Y) = \left[ \sum_{i=1}^n (x_i - y_i)^2 \right]^{\frac{1}{2}}$$

The City – Block distance

$$d_C(X, Y) = \sum_{i=1}^n |x_i - y_i|$$

## 2.2 Clustering Evaluation Criteria

**Definition 2.2.1.** The corrected rand (CR) index [21] is utilized to evaluate the accuracy of the clustering result. CR index is defined as follows

$$CR = \frac{\sum_{i=1}^R \sum_{j=1}^C \binom{n_{ij}}{2} - \binom{n}{2}^{-1} \sum_{i=1}^R \binom{n_i}{2} \sum_{j=1}^C \binom{n_j}{2}}{\frac{1}{2} \left[ \sum_{i=1}^R \binom{n_i}{2} + \sum_{j=1}^C \binom{n_j}{2} \right] - \binom{n}{2}^{-1} \sum_{i=1}^R \binom{n_i}{2} \sum_{j=1}^C \binom{n_j}{2}}$$

where  $n_{ij}$  represents the number of objects that are in clusters  $u_i$  and  $v_j$ ;  $n_i$  indicates the number of objects in cluster  $u_i$ ;  $n_j$  indicates the number of objects in cluster  $v_j$ ; and  $n$  is the total number of objects. CR is an external measure that can make the comparison between the partition produced by a clustering algorithm and the actual partition where “ground-truth” labeling is known. The closer the CR is to 1, the better the clustering result is.

**Definition 2.2.2.** The partition coefficient and entropy are used to evaluate the quality of fuzzy clusters. They are given as follows

$$PC = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^k \mu_{ji}^2,$$

$$PE = -\frac{1}{N} \sum_{i=1}^N \sum_{h=1}^k \mu_{ji} \log(\mu_{ji}),$$

where  $k$  and  $N$  are the number of clusters and objects, respectively. The  $PC$  index has the value in  $[1/k, 1]$ . The closer to unity the index the “crisper” the clustering is.

### 3 The Proposed Technique

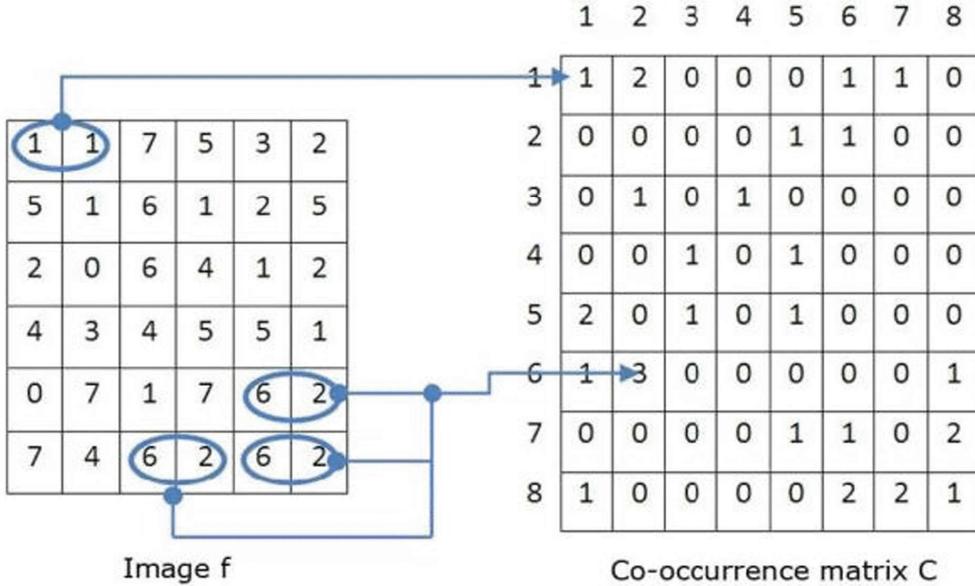
#### 3.1 Methods to Extract Image Data

Image recognition which has many applications is a very interesting and challenging problem. It is the foundation for many applications in the fields of medicine, environment, security [4, 15, 16, 22]. In this study, we performed cluster analysis for images by the proposed algorithm. To identify the image, we must first extract data from the grayscale image. There are many methods of doing this as presented in [23]. Furthermore, we extracted the feature of the image based on the grey level co-occurrence matrix (GLCM) which is a popular and effective method today [24]. The GLCM is a common technique in statistical image analysis that is used to estimate image properties related to second-order statistics. GLCM considers the relation between two neighbor pixels in one offset, as the second order texture, where the first pixel is called reference and the second one the neighbor pixel. The GLCM presents the information about intensities of pixels and their neighbors at fixed distance  $d$  and orientation  $\theta$ . If we have an image with the size of  $M \times N$  ( $M$  pixels in  $X$ -axis and  $N$  pixels in  $Y$ -axis) and  $G$  is the domain of grey level. Then, GLCM is a matrix  $P$  with the size of  $G \times G$ . Each element  $p(i,j)$  presents the probability of the occurrence of intensity  $i$  and intensity  $j$  at fixed distance  $d$  and orientation  $\theta$ . The formula to compute  $p(i,j)$  is presented by (1):

$$p_{d\theta}(i,j) = \{(x, y), (x', y') \in M \times N | d = \|(x, y), (x', y')\|, \\ \theta = \Theta((x, y), (x', y')), f(x, y) = i, f(x', y') = j\}$$

$$p_{d\theta}(i,j) = \left\{ (x, y), (x', y') \in M \times N \middle| \begin{array}{l} d = \|(x, y), (x', y')\|, \\ \theta = \Theta((x, y), (x', y')) \end{array} f(x, y) = i, f(x', y') = j \right\}. \quad (1)$$

From GLCM, Haralick [8] defined 14 statistical measures that can be extracted. However, there are only 4 features consisting of contrast, correlation, homogeneity, and energy having strong affects on classification result. These features are presented in Table 1 (Fig. 1).

**Fig. 1.** Illustrating how to calculate GLCM.**Table 1.** Formula to extract four features from GLCM.

| Features    | Formula  |
|-------------|--|
| Contrast    | $\sum_{i,j}  i - j ^k p^l(i, j)$                                     |
| Correlation | $\sum_{i,j} \frac{(i - \mu_i)(j - \mu_j)p(i, j)}{\delta_i \delta_j}$ |
| Entropy     | $\sum_{i,j} p(i, j)^2$   |
| Homogeneity | $\sum_{i,j} \frac{p(i, j)}{1 +  i - j }$                             |

### 3.2 The Proposed Algorithm

Suppose we have  $N$  images  $X = \{a_1, a_2, \dots, a_n\}$ . The Automatic Fuzzy Clustering Algorithm for discrete (AFCA) includes the two stages:

In the first phase of the proposed algorithm, there are four steps of finding the number of suitable groups. The second phase in the AFCA consists of three steps which can determine the probability to assign each image to clusters. The steps of the AFCA are shown as follows.

**Step 1.** Extract 64 features based on GLCM to data set  $I_{N \times 64}$  where  $N$  is the number of images.

**Step 2.** Initialize  $t = 0$  and  $V^{(0)} = X = \{v_1, v_2, \dots, v_n\}$

**Step 3.** Update the prototype images using (2):

$$v_i^{(t+1)} = \sum_{j=1}^N \frac{f(v_i^{(t)}, v_j^{(t)})}{\sum_{k=1}^N f(v_i^{(t)}, v_k^{(t)})} v_j^{(t)}, i = 1, \dots, N, \quad (2)$$

where

$$f(v_i^{(t)}, v_j^{(t)}) = \begin{cases} \exp\left[-d(v_i^{(t)}, v_j^{(t)})/\sigma\right] & \text{if } d(v_i^{(t)}, v_j^{(t)}) \leq \mu\alpha_{ij}(t), \\ 0 & \text{if } d(v_i^{(t)}, v_j^{(t)}) > \mu\alpha_{ij}(t), \end{cases}$$

with

$$\alpha_{ij}(0) = 1, \alpha_{ij}(t) = \frac{\alpha_{ij}(t-1)}{1 + \alpha_{ij}(t-1)f(v_i^{(t-1)}, v_j^{(t-1)})}, t \geq 1,$$

$$\mu = \frac{1}{N} \sum_{i < j} d(v_i^{(0)}, v_j^{(0)}), \sigma = \sqrt{\frac{1}{N} \sum_{i < j} [d(v_i^{(0)}, v_j^{(0)}) - \mu]^2}.$$

**Step 4.** Repeat Step 3 until

$$\|v^{(t)} - v^{(t-1)}\| < \varepsilon.$$

**Step 5.** Find the number of images in  $v^{(t)}$ . If it has  $c$  images in  $v_i^{(t)}$  then we have  $c$  clusters  $C_1, C_2, \dots, C_c$ . Establish the initial matrix partition with  $h = 0$

$$U^{(0)} = [\mu_{ik}^{(0)}]_{c \times N},$$

where  $\mu_{ik} = 1$  if the interval  $k$  belongs to the cluster  $i$ ,  $\mu_{ik} = 0$  for otherwise.

Find the representative image of the cluster  $C_i$  by (3):

$$w_i = \frac{\sum_{k=1}^N (\mu_{ik})^m a_k}{\sum_{k=1}^N (\mu_{ik})^m}, \quad (3)$$

Where  $\mu_{ik} \in U^{(h)}$ ,  $1 \leq i, j \leq c$ ,  $1 \leq k \leq N$  is fuzzy probability of  $c$  clusters.

The weighted exponent  $m$  of formula (3) has an effect on the fuzzy degree of result. When  $m = 1$ , fuzzy clustering becomes non-fuzzy clustering. In this article, we choose the value  $m = 2$  in our numerical examples and applications.

**Step 6.** Update the new partition matrix  $U^{(t)}$ , where each image of  $U^{(t)}$  is determined by the formula (4):

$$\mu_{ik}^{(t)} = \frac{\lambda_i d_H(w_i, a_k)^2}{\sum_{j=1}^c \lambda_j d_H(w_j, a_k)^2}, \quad 1 \leq i, k \leq c; 1 \leq k \leq N \quad (4)$$

with

$d_H(w_i, a_k)$  is the Hausdorff distance of  $w_i$  central cluster and original data  $a_k$ .

$\lambda_i = \frac{\left\{ \prod_{i=1}^c \left[ \sum_{k=1}^n (\mu_{ik}^{t-1})^2 d_H(w_i, a_k) \right] \right\}^{\frac{1}{c}}}{\sum_{k=1}^n (\mu_{ik}^{t-1})^2 d_H(w_i, a_k)}$  is the vector of weights and

$$\prod_{i=1}^k \lambda_i = 1.$$

**Step 7.** Repeat Step 6 until

$$\|G(U^{(t)}) - G(U^{(t-1)})\| < \varepsilon. \quad (5)$$

where  $w_i$  and  $\bar{w}$  are central cluster and average of central cluster, respectively,

$$G(U^{(t)}) = \frac{\sum_{i=1}^c \sum_{k=1}^N \mu_{ik}^2 d_H(w_i, a_k) + \frac{1}{c} \sum_{i=1}^c \|w_i - \bar{w}\|^2}{\min_{i \neq j} \|w_i - w_j\|^2} \quad (6)$$

$G$  is the objective function of the proposed algorithm. The closer the  $G$  function is small, the better the fuzzy result is.

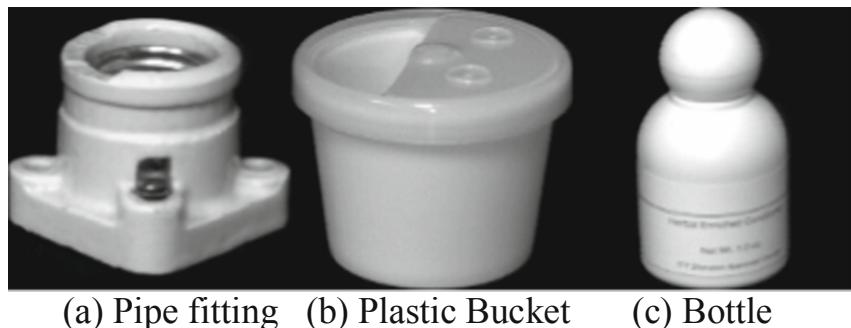
**Step 8.** Export the result of cluster analysis

$$Z = \arg \max_{1 \leq i \leq N} \{U_{ik}^{(t)}\}, k = 1, \dots, c.$$

## 4 Applying in Image Recognition

The proposed algorithm has been coded in Matlab. In order to show its effectiveness with the existing methods, two data sets are considered and taken from the database, called Columbia Object Image Library (COIL-20) (<http://deeplearning.net/datasets>). We gave two applications for annotation and retrieval. The first application creates data set from texture images, whereas the database of the second application is extracted from a color image set. A detailed description of each of the data sets is given below.

**Example 1.** The first data set utilized in this example is composed of three different objects, namely Pipe fitting, Plastic Bucket, and Bottle. There are four groups of equal sample size of 72 images. Some image samples are given in Fig. 2.



**Fig. 2.** Three image samples: (a) Pipe fitting, (b) Plastic Bucket and (c) Bottle.

To begin with, we use the proposed method and detailed presentation a step by step to recognize 216 images.

**Phase 1.** Finding a suitable number of cluster for 216 images.

**Step 1.** We use  $d = 1, 2, 3, 4$  and  $\theta = \{0, 90^\circ, 180^\circ, 270^\circ\}$  to create 16 GLCMs. In this step, we extract the contrast, the correlation, the energy and the homogeneity from each GLCM to establish 64 extracted features. As a result, we receive  $216 \times 64$  matrix which is data set extracted from 216 texture images to recognize (see Fig. 3).

| Variables - kq |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| kq             |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 216x64 double  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 1              | 2      | 3      | 4      | 5      | 6      | 7      | 8      | 9      | 10     | 11     | 12     | 13     | 14     | 15     | 16     | 17     | 18     |        |
| 2              | 0.1604 | 0.5108 | 0.4202 | 0.5046 | 0.3971 | 1.2523 | 1.0201 | 1.2558 | 0.6762 | 1.9494 | 1.5724 | 1.9748 | 0.9705 | 2.4992 | 1.9965 | 2.5792 | 0.9845 | 0.9503 |
| 3              | 0.1532 | 0.5015 | 0.4165 | 0.4918 | 0.3666 | 1.2507 | 1.0392 | 1.2337 | 0.6181 | 1.9622 | 1.6305 | 1.9722 | 0.8938 | 2.5305 | 2.0961 | 2.6147 | 0.9855 | 0.9522 |
| 4              | 0.1516 | 0.5029 | 0.4198 | 0.4874 | 0.3618 | 1.2921 | 1.0915 | 1.2709 | 0.6159 | 2.0859 | 1.7764 | 2.1030 | 0.8921 | 2.7294 | 2.3247 | 2.8340 | 0.9866 | 0.9554 |
| 5              | 0.1527 | 0.4902 | 0.4029 | 0.4694 | 0.3620 | 1.2516 | 1.0404 | 1.2173 | 0.6034 | 1.9950 | 1.6814 | 2.0012 | 0.8727 | 2.6591 | 2.2456 | 2.7387 | 0.9863 | 0.9560 |
| 6              | 0.1516 | 0.4943 | 0.4060 | 0.4709 | 0.3619 | 1.2563 | 1.0360 | 1.2092 | 0.6067 | 1.9972 | 1.6735 | 1.9779 | 0.8785 | 2.6444 | 2.2171 | 2.6894 | 0.9864 | 0.9557 |
| 7              | 0.1476 | 0.4874 | 0.4022 | 0.4608 | 0.3553 | 1.2304 | 1.0196 | 1.1883 | 0.6008 | 1.9503 | 1.6304 | 1.9301 | 0.8616 | 2.5764 | 2.1618 | 2.6037 | 0.9864 | 0.9551 |
| 8              | 0.1545 | 0.4929 | 0.3991 | 0.4595 | 0.3669 | 1.2435 | 1.0262 | 1.1931 | 0.6128 | 1.9513 | 1.6269 | 1.9367 | 0.8815 | 2.5633 | 2.1425 | 2.6091 | 0.9857 | 0.9544 |
| 9              | 0.1505 | 0.4968 | 0.4051 | 0.4615 | 0.3581 | 1.2487 | 1.0344 | 1.1836 | 0.6110 | 1.9520 | 1.6325 | 1.9272 | 0.8822 | 2.5492 | 2.1447 | 2.6103 | 0.9858 | 0.9529 |
| 10             | 0.1547 | 0.5009 | 0.4021 | 0.4623 | 0.3793 | 1.2719 | 1.0326 | 1.1970 | 0.6451 | 1.9969 | 1.6486 | 1.9600 | 0.9343 | 2.6249 | 2.1765 | 2.6686 | 0.9856 | 0.9532 |
| 11             | 0.1530 | 0.4897 | 0.3951 | 0.4550 | 0.3704 | 1.2346 | 1.0125 | 1.1854 | 0.6276 | 1.9553 | 1.6191 | 1.9385 | 0.9060 | 2.5865 | 2.1513 | 2.6410 | 0.9853 | 0.9530 |
| 12             | 0.1595 | 0.5081 | 0.4071 | 0.4713 | 0.3955 | 1.2896 | 1.0465 | 1.2353 | 0.6718 | 2.0450 | 1.6824 | 2.0264 | 0.9665 | 2.7155 | 2.2472 | 2.7681 | 0.9852 | 0.9529 |
| 13             | 0.1673 | 0.5216 | 0.4177 | 0.4884 | 0.4107 | 1.3507 | 1.0887 | 1.2855 | 0.6985 | 2.1683 | 1.7843 | 2.1391 | 1.0093 | 2.8842 | 2.3991 | 2.9266 | 0.9850 | 0.9530 |
| 14             | 0.1617 | 0.5156 | 0.4167 | 0.4809 | 0.4005 | 1.3160 | 1.0694 | 1.2555 | 0.6846 | 2.1342 | 1.7533 | 2.1002 | 0.9928 | 2.8710 | 2.3734 | 2.8957 | 0.9849 | 0.9518 |
| 15             | 0.1627 | 0.5201 | 0.4161 | 0.4825 | 0.4061 | 1.3411 | 1.0734 | 1.2630 | 0.6946 | 2.1731 | 1.7614 | 2.1053 | 1.0041 | 2.8936 | 2.3599 | 2.8749 | 0.9845 | 0.9500 |
| 16             | 0.1737 | 0.5570 | 0.4491 | 0.5204 | 0.4241 | 1.3959 | 1.1104 | 1.3093 | 0.7262 | 2.2152 | 1.7747 | 2.1292 | 1.0533 | 2.9227 | 2.3361 | 2.8810 | 0.9833 | 0.9462 |
| 17             | 0.1651 | 0.5263 | 0.4250 | 0.4876 | 0.4077 | 1.3154 | 1.0392 | 1.2288 | 0.7089 | 2.0783 | 1.6322 | 1.9944 | 1.0391 | 2.7576 | 2.1313 | 2.7078 | 0.9833 | 0.9462 |
| 18             | 0.1718 | 0.5191 | 0.4139 | 0.4872 | 0.4306 | 1.3085 | 1.0076 | 1.2312 | 0.7501 | 2.0972 | 1.5981 | 2.0191 | 1.1038 | 2.7729 | 2.0844 | 2.7605 | 0.9833 | 0.9493 |
| 19             | 0.1705 | 0.5287 | 0.4242 | 0.5034 | 0.4265 | 1.3250 | 1.0518 | 1.2873 | 0.7420 | 2.1199 | 1.6679 | 2.1049 | 1.0903 | 2.8091 | 2.1890 | 2.8662 | 0.9837 | 0.9490 |
| 20             | 0.1603 | 0.5296 | 0.4317 | 0.4959 | 0.3934 | 1.3188 | 1.0645 | 1.2583 | 0.6859 | 2.1228 | 1.7019 | 2.0529 | 1.0022 | 2.8173 | 2.2528 | 2.8168 | 0.9849 | 0.9497 |
| 21             | 0.1552 | 0.5007 | 0.4019 | 0.4583 | 0.3698 | 1.2477 | 1.0045 | 1.1560 | 0.6218 | 2.0054 | 1.6351 | 1.9102 | 0.9040 | 2.6692 | 2.1870 | 2.6387 | 0.9857 | 0.9536 |
| 22             | 0.1499 | 0.4968 | 0.4026 | 0.4558 | 0.3580 | 1.2522 | 1.0236 | 1.1692 | 0.6014 | 2.0453 | 1.6904 | 1.9450 | 0.8731 | 2.7380 | 2.2779 | 2.7026 | 0.9867 | 0.9558 |
| 23             | 0.1500 | 0.4842 | 0.3943 | 0.4489 | 0.3558 | 1.2206 | 0.9960 | 1.1493 | 0.5936 | 1.9818 | 1.6396 | 1.8982 | 0.8586 | 2.6742 | 2.2390 | 2.6591 | 0.9870 | 0.9580 |
| 24             | 0.1491 | 0.4750 | 0.3845 | 0.4370 | 0.3506 | 1.2022 | 0.9840 | 1.1201 | 0.5805 | 1.9446 | 1.6207 | 1.8502 | 0.8325 | 2.6265 | 2.2083 | 2.5771 | 0.9873 | 0.9594 |
| 25             | 0.1473 | 0.4838 | 0.3948 | 0.4422 | 0.3512 | 1.2062 | 0.9885 | 1.1212 | 0.5881 | 1.9478 | 1.6099 | 1.8348 | 0.8431 | 2.6184 | 2.1826 | 2.5364 | 0.9875 | 0.9587 |
| 26             | 0.1468 | 0.4894 | 0.4015 | 0.4453 | 0.3497 | 1.2029 | 0.9840 | 1.1071 | 0.5906 | 1.9386 | 1.5884 | 1.8122 | 0.8513 | 2.5792 | 2.1284 | 2.4960 | 0.9873 | 0.9576 |

Fig. 3. The data set is extracted from 216 texture images.

**Step 2.** We start initializing  $V^{(0)} = \{v_1^{(0)}, v_2^{(0)}, \dots, v_{216}^{(0)}\}$  and calculate distance among images by the Hausdorff distance before updating the prototype images (see Fig. 4).

| d              |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 216x216 double |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 1              | 2      | 3      | 4      | 5      | 6      | 7      | 8      | 9      | 10     | 11     | 12     | 13     | 14     | 15     | 16     | 17     | 18     |        |        |
| 1              | 0      | 0.2116 | 0.3501 | 0.7611 | 0.5751 | 0.5318 | 0.4389 | 0.4163 | 0.4162 | 0.4743 | 0.4197 | 0.6087 | 0.9357 | 0.8625 | 0.8757 | 0.9337 | 0.5264 | 0.5389 |        |
| 2              | 0.2116 | 0      | 0.1668 | 0.5681 | 0.3986 | 0.3619 | 0.3026 | 0.2774 | 0.2773 | 0.3188 | 0.2817 | 0.4355 | 0.7584 | 0.6887 | 0.7055 | 0.7665 | 0.3888 | 0.4250 |        |
| 3              | 0.3501 | 0.1668 | 0      | 0.4471 | 0.2594 | 0.2247 | 0.1814 | 0.1686 | 0.1645 | 0.2186 | 0.1799 | 0.3415 | 0.6710 | 0.5977 | 0.6232 | 0.7046 | 0.3671 | 0.4350 |        |
| 4              | 0.7611 | 0.5681 | 0.4471 | 0      | 0.2493 | 0.3069 | 0.4563 | 0.4612 | 0.4641 | 0.3735 | 0.4397 | 0.2283 | 0.2785 | 0.2239 | 0.2684 | 0.3819 | 0.3874 | 0.4472 |        |
| 5              | 0.5751 | 0.3986 | 0.2594 | 0.2493 | 0      | 0.0733 | 0.2169 | 0.2265 | 0.2317 | 0.1627 | 0.1627 | 0.2039 | 0.1638 | 0.4718 | 0.3993 | 0.4519 | 0.5702 | 0.3652 |        |
| 6              | 0.5318 | 0.3619 | 0.2247 | 0.3069 | 0.0733 | 0      | 0.1550 | 0.1631 | 0.1698 | 0.1060 | 0.1459 | 0.1844 | 0.5184 | 0.4486 | 0.4986 | 0.6115 | 0.3739 | 0.4402 |        |
| 7              | 0.4389 | 0.3026 | 0.1814 | 0.4563 | 0.2169 | 0.1550 | 0      | 0.0443 | 0.0525 | 0.1456 | 0.0769 | 0.3194 | 0.6687 | 0.5976 | 0.6431 | 0.7500 | 0.4504 | 0.5146 |        |
| 8              | 0.4163 | 0.2774 | 0.1686 | 0.4612 | 0.2265 | 0.1631 | 0.0443 | 0      | 0.0302 | 0.1303 | 0.0693 | 0.0976 | 0      | 0.2765 | 0.6345 | 0.5629 | 0.6091 | 0.7150 | 0.4072 |
| 9              | 0.4162 | 0.2773 | 0.1645 | 0.4641 | 0.2317 | 0.1698 | 0.0525 | 0.0302 | 0      | 0.1378 | 0.0693 | 0.3201 | 0.6731 | 0.6037 | 0.6476 | 0.7493 | 0.4414 | 0.5043 |        |
| 10             | 0.4743 | 0.3188 | 0.2186 | 0.3735 | 0.1627 | 0.1060 | 0.1456 | 0.1303 | 0.1378 | 0      | 0.0976 | 0.1928 | 0.5503 | 0.5950 | 0.4843 | 0.5327 | 0.6374 | 0.3643 |        |
| 11             | 0.4197 | 0.2817 | 0.1799 | 0.4397 | 0.2039 | 0.1459 | 0.0769 | 0.0600 | 0.0693 | 0.0976 | 0      | 0.2765 | 0.6345 | 0.5629 | 0.6091 | 0.7150 | 0.4072 | 0.4618 |        |
| 12             | 0.6087 | 0.4355 | 0.3415 | 0.2283 | 0.1638 | 0.1844 | 0.3194 | 0.3129 | 0.3201 | 0.1928 | 0.2765 | 0      | 0.3610 | 0.2964 | 0.3521 | 0.4639 | 0.3024 | 0.3448 |        |
| 13             | 0.9357 | 0.7584 | 0.6710 | 0.2785 | 0.4718 | 0.5184 | 0.6687 | 0.6674 | 0.6731 | 0.5503 | 0.6345 | 0.3610 | 0      | 0.1018 | 0.1459 | 0.2373 | 0.4784 | 0.4938 |        |
| 14             | 0.8625 | 0.6887 | 0.5977 | 0.2239 | 0.3993 | 0.4486 | 0.5976 | 0.5986 | 0.6037 | 0.4843 | 0.5629 | 0.2964 | 0.1018 | 0      | 0.1006 | 0.2401 | 0.4018 | 0.4270 |        |
| 15             | 0.8757 | 0.7055 | 0.6232 | 0.2684 | 0.4519 | 0.4986 | 0.6431 | 0.6434 | 0.6476 | 0.5327 | 0.6091 | 0.3521 | 0.1459 | 0.1006 | 0      | 0.1556 | 0.3818 | 0.4090 |        |
| 16             | 0.9337 | 0.7665 | 0.7046 | 0.3819 | 0.5702 | 0.6115 | 0.7500 | 0.7460 | 0.7493 | 0.6374 | 0.7150 | 0.4639 | 0.2373 | 0.2401 | 0.1556 | 0      | 0.4222 | 0.4374 |        |
| 17             | 0.5264 | 0.3888 | 0.3671 | 0.3874 | 0.3652 | 0.3739 | 0.4504 | 0.4396 | 0.4414 | 0.3643 | 0.4072 | 0.3024 | 0.4788 | 0.4018 | 0.3818 | 0.4222 | 0      | 0.1272 |        |
| 18             | 0.5389 | 0.4250 | 0.4350 | 0.4472 | 0.4315 | 0.4402 | 0.5146 | 0.4991 | 0.5043 | 0.4166 | 0.4618 | 0.3448 | 0.4938 | 0.4270 | 0.4090 | 0.4374 | 0.1272 | 0      |        |
| 19             | 0.7069 | 0.5545 | 0.5242 | 0.3438 | 0.4376 | 0.4713 | 0.5898 | 0.5783 | 0.5853 | 0.4742 | 0.5405 | 0.3246 | 0.3231 | 0.     |        |        |        |        |        |

|    | 216x64 double |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|----|---------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
|    | 1             | 2      | 3      | 4      | 5      | 6      | 7      | 8      | 9      | 10     | 11     | 12     | 13     | 14     | 15     | 16     | 17     | 18     |
| 1  | 0.1615        | 0.4895 | 0.3965 | 0.4802 | 0.4025 | 1.2224 | 0.9676 | 1.2015 | 0.6930 | 1.9240 | 1.4891 | 1.9093 | 1.0107 | 2.4681 | 1.8682 | 2.5104 | 0.9836 | 0.9499 |
| 2  | 0.1604        | 0.5108 | 0.4202 | 0.5046 | 0.3971 | 1.2523 | 1.0201 | 1.2558 | 0.6762 | 1.9494 | 1.5724 | 1.9748 | 0.9705 | 2.4992 | 1.9965 | 2.5792 | 0.9845 | 0.9503 |
| 3  | 0.1532        | 0.5015 | 0.4165 | 0.4918 | 0.3666 | 1.2507 | 1.0392 | 1.2337 | 0.6181 | 1.9622 | 1.6305 | 1.9722 | 0.8938 | 2.5305 | 2.0961 | 2.6147 | 0.9855 | 0.9522 |
| 4  | 0.1516        | 0.5029 | 0.4198 | 0.4874 | 0.3618 | 1.2921 | 1.0915 | 1.2709 | 0.6159 | 2.0859 | 1.7764 | 2.1030 | 0.8921 | 2.7294 | 2.3247 | 2.8340 | 0.9866 | 0.9554 |
| 5  | 0.1527        | 0.4902 | 0.4029 | 0.4694 | 0.3620 | 1.2516 | 1.0404 | 1.2173 | 0.6034 | 1.9950 | 1.6814 | 2.0012 | 0.8727 | 2.6591 | 2.2456 | 2.7387 | 0.9863 | 0.9560 |
| 6  | 0.1518        | 0.4943 | 0.4060 | 0.4709 | 0.3619 | 1.2563 | 1.0360 | 1.2092 | 0.6067 | 1.9972 | 1.6735 | 1.9779 | 0.8785 | 2.6444 | 2.2171 | 2.6894 | 0.9864 | 0.9557 |
| 7  | 0.1476        | 0.4874 | 0.4022 | 0.4608 | 0.3553 | 1.2304 | 1.0196 | 1.1883 | 0.6008 | 1.9503 | 1.6304 | 1.9301 | 0.8616 | 2.5764 | 2.1618 | 2.6037 | 0.9864 | 0.9551 |
| 8  | 0.1545        | 0.4929 | 0.3991 | 0.4595 | 0.3669 | 1.2435 | 1.0262 | 1.1931 | 0.6128 | 1.9513 | 1.6269 | 1.9367 | 0.8815 | 2.5633 | 2.1425 | 2.6091 | 0.9857 | 0.9544 |
| 9  | 0.1505        | 0.4968 | 0.4051 | 0.4615 | 0.3581 | 1.2487 | 1.0344 | 1.1836 | 0.6110 | 1.9520 | 1.6325 | 1.9272 | 0.8822 | 2.5492 | 2.1447 | 2.6103 | 0.9858 | 0.9529 |
| 10 | 0.1547        | 0.5009 | 0.4021 | 0.4623 | 0.3793 | 1.2719 | 1.0326 | 1.1970 | 0.6451 | 1.9969 | 1.6486 | 1.9600 | 0.9343 | 2.6249 | 2.1765 | 2.6686 | 0.9856 | 0.9532 |
| 11 | 0.1530        | 0.4897 | 0.3951 | 0.4550 | 0.3704 | 1.2346 | 1.0125 | 1.1854 | 0.6276 | 1.9553 | 1.6191 | 1.9385 | 0.9061 | 2.5865 | 2.1513 | 2.6410 | 0.9853 | 0.9530 |
| 12 | 0.1595        | 0.5081 | 0.4071 | 0.4713 | 0.3955 | 1.2896 | 1.0465 | 1.2353 | 0.6718 | 2.0450 | 1.6824 | 2.0264 | 0.9665 | 2.7155 | 2.2472 | 2.7681 | 0.9852 | 0.9529 |
| 13 | 0.1673        | 0.5216 | 0.4177 | 0.4884 | 0.4107 | 1.3507 | 1.0887 | 1.2855 | 0.6985 | 2.1683 | 1.7843 | 2.1391 | 1.0093 | 2.8842 | 2.3991 | 2.9266 | 0.9850 | 0.9530 |
| 14 | 0.1617        | 0.5156 | 0.4167 | 0.4809 | 0.4005 | 1.3160 | 1.0694 | 1.2555 | 0.6846 | 2.1342 | 1.7533 | 2.1002 | 0.9928 | 2.8710 | 2.3734 | 2.8957 | 0.9849 | 0.9518 |
| 15 | 0.1627        | 0.5201 | 0.4161 | 0.4825 | 0.4061 | 1.3411 | 1.0734 | 1.2630 | 0.6946 | 2.1731 | 1.7614 | 2.1053 | 1.0041 | 2.8936 | 2.3599 | 2.8749 | 0.9845 | 0.9500 |
| 16 | 0.1737        | 0.5570 | 0.4491 | 0.5204 | 0.4241 | 1.3959 | 1.1104 | 1.3093 | 0.7262 | 2.2152 | 1.7747 | 2.1292 | 1.0533 | 2.9227 | 2.3361 | 2.8810 | 0.9833 | 0.9462 |
| 17 | 0.1651        | 0.5263 | 0.4250 | 0.4876 | 0.4077 | 1.3154 | 1.0392 | 1.2288 | 0.7089 | 2.0783 | 1.6322 | 1.9944 | 1.0391 | 2.7576 | 2.1313 | 2.7078 | 0.9833 | 0.9462 |
| 18 | 0.1718        | 0.5191 | 0.4139 | 0.4872 | 0.4306 | 1.3085 | 1.0076 | 1.2312 | 0.7501 | 2.0972 | 1.5981 | 2.0191 | 1.1038 | 2.7729 | 2.0844 | 2.7605 | 0.9833 | 0.9493 |
| 19 | 0.1705        | 0.5287 | 0.4242 | 0.5034 | 0.4265 | 1.3250 | 1.0518 | 1.2873 | 0.7420 | 2.1199 | 1.6679 | 2.1049 | 1.0903 | 2.8091 | 2.1890 | 2.8662 | 0.9837 | 0.9490 |
| 20 | 0.1603        | 0.5296 | 0.4317 | 0.4959 | 0.3934 | 1.3188 | 1.0645 | 1.2583 | 0.6859 | 2.1228 | 1.7019 | 2.0529 | 1.0022 | 2.8173 | 2.2528 | 2.8168 | 0.9849 | 0.9497 |
| 21 | 0.1552        | 0.5007 | 0.4019 | 0.4583 | 0.3698 | 1.2477 | 1.0045 | 1.1560 | 0.6218 | 2.0054 | 1.6351 | 1.9102 | 0.9040 | 2.6692 | 2.1870 | 2.6387 | 0.9857 | 0.9536 |
| 22 | 0.1499        | 0.4968 | 0.4026 | 0.4558 | 0.3580 | 1.2522 | 1.0236 | 1.1692 | 0.6014 | 2.0453 | 1.6904 | 1.9450 | 0.8731 | 2.7380 | 2.2779 | 2.7026 | 0.9867 | 0.9558 |
| 23 | 0.1500        | 0.4842 | 0.3943 | 0.4489 | 0.3558 | 1.2206 | 0.9960 | 1.1493 | 0.5936 | 1.9818 | 1.6396 | 1.8982 | 0.8586 | 2.6742 | 2.2390 | 2.6591 | 0.9870 | 0.9580 |
| 24 | 0.1491        | 0.4750 | 0.3845 | 0.4370 | 0.3506 | 1.2022 | 0.9840 | 1.1201 | 0.5805 | 1.9446 | 1.6207 | 1.8502 | 0.8325 | 2.6265 | 2.2083 | 2.5771 | 0.9873 | 0.9594 |
| 25 | 0.1473        | 0.4838 | 0.3948 | 0.4422 | 0.3512 | 1.2062 | 0.9885 | 1.1212 | 0.5881 | 1.9478 | 1.6099 | 1.8348 | 0.8431 | 2.6184 | 2.1826 | 2.5364 | 0.9875 | 0.9587 |
| 26 | 0.1468        | 0.4894 | 0.4015 | 0.4453 | 0.3497 | 1.2029 | 0.9840 | 1.1071 | 0.5906 | 1.9386 | 1.5884 | 1.8122 | 0.8513 | 2.5792 | 2.1284 | 2.4960 | 0.9873 | 0.9576 |

**Fig. 5.** The new data set is updated at  $t = 1$ .

**Step 4.** We have deviation  $\|v^{(1)} - v^{(0)}\| = 0.5062$  which is greater than  $\varepsilon = 0.0001$ . Repeating the second step until statisfing condition which is smaller than  $\varepsilon$ . The clustering result collected is three groups ( $c = 3$ ) after 182 iterations.

The Table 2 compares and contrasts data on differences in the result of the proposed method at the first stage (AFCA) using three different distances to determine the number of cluster, namely the Euclidean, the City - Block and the Hausdorff distance through three criteria as running time, iteration and CR index.

**Table 2.** The result of the AFCA for 216 images

| Method       | Time (s) | Inter | CR index |
|--------------|----------|-------|----------|
| Proposed - E | 425.7543 | 744   | 1.000    |
| Proposed - C | 512.0382 | 897   | 1.000    |
| Proposed - H | 161.0938 | 182   | 1.000    |

Overall, all methods achieve a similar clustering result with  $CR = 1$ . However, the iteration of AFCA based on the Hausdorff distance is the smallest while that of AFCA based on City-Block is the highest. As a result, a computational time of AFCA using the Hausdorff distance is the least of nearly 161 s. It is meaningful to apply the proposed method in reality because of spending less computational cost.

**Phase 2.** Determining the probability to assign each image to cluster for 216 images after determining the number of clusters ( $c = 3$ ).

**Step 5.** Build the initial partition with  $h = 0$

$$U_{3,216}^{(0)} = \begin{bmatrix} 111 \dots 000 \dots 000 \\ 000 \dots 111 \dots 000 \\ 000 \dots 000 \dots 111 \end{bmatrix}$$

Find the representative image of the three cluster as follow

$$w = \begin{bmatrix} 0.1576 & 0.5078 & 0.4122 & \dots & 0.6856 & 0.7360 & 0.6851 \\ 0.1061 & 0.2375 & 0.1835 & \dots & 0.7861 & 0.8412 & 0.7827 \\ 0.1329 & 0.2368 & 0.1580 & \dots & 0.8290 & 0.8950 & 0.8192 \end{bmatrix}$$

**Step 6.** Update the new partition matrix  $U(1)$  with the vector of weights  $\lambda = [0.7542 \ 1.0747 \ 1.2338]$ .

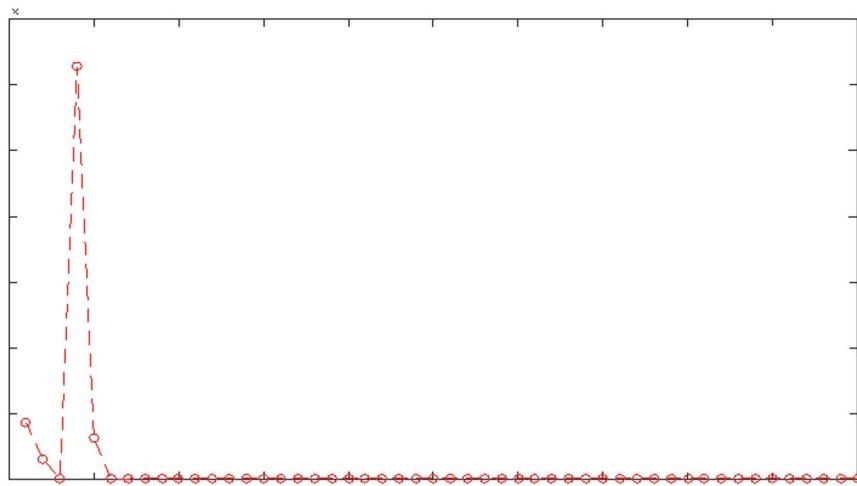
$$U_{3,216}^{(1)} = \begin{bmatrix} 0.9629 & 0.9862 & 0.9944 & \dots & 0.0025 & 0.0008 & 0.0015 & \dots & 0.0211 & 0.0217 & 0.0286 \\ 0.0154 & 0.0058 & 0.0024 & \dots & 0.9729 & 0.9909 & 0.9827 & \dots & 0.0780 & 0.0797 & 0.0945 \\ 0.0217 & 0.0080 & 0.0032 & \dots & 0.0246 & 0.0083 & 0.0157 & \dots & 0.9010 & 0.8986 & 0.8769 \end{bmatrix}$$

**Step 7.** Computer deviation  $\|G^{(1)} - G^{(0)}\| = 0.4075$  which is greater than  $\varepsilon = 0.00001$ .

**Step 8.** Determine the result of cluster analysis  $Z$  in the 7 iterations

$$\begin{aligned} Z_1 &= \{a_1, a_2, \dots, a_{72}\} \\ Z_2 &= \{a_{73}, a_{74}, \dots, a_{114}\} \\ Z_3 &= \{a_{115}, a_{116}, \dots, a_{216}\}. \end{aligned}$$

We find the probability to assign each image to cluster for 216 images based on fuzzy clustering algorithm using the objective function which is improved further than previous ones (see Fig. 3).



**Fig. 6.** The convergence of the AFCA after 50 iterations.

Figure 6 has shown that the proposed algorithm converge from 7 out of 1000 iterations with  $f^{1000} = 4838695.2187$ . It is effective that the objective function gave the AFCA an edge in terms of optimization the number of iterations.

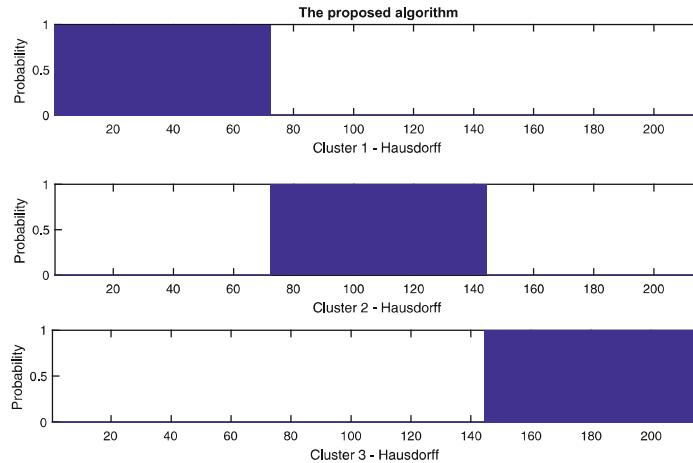
The Table 3 illustrates data on differences in the outcome of proposed method using Euclidean, City-Block and Hausdorff distance to compare with FCM method based on PC and PE index.

**Table 3.** The fuzzy clustering result of the proposed method for first data set

| Method   | No.cluster | PC index | PE index |
|----------|------------|----------|----------|
| FCM - E  | 3          | 0.8212   | 0.3410   |
| FCM - C  | 3          | 0.7901   | 0.3999   |
| FCM - H  | 3          | 0.8655   | 0.2056   |
| AFCA - E | 3          | 0.9565   | 0.1065   |
| AFCA - C | 3          | 0.9619   | 0.0980   |
| AFCA - H | 3          | 1.0000   | 0.0000   |

An overview of the Table 3 reveals that AFCA - H is the highest ( $PC = 1$ ) while the reverse is true for FCM - C with  $PC = 0.7901$ . With regard to the proposed method using three distances, PC index is over 0.95, which is slightly higher than FCM, under 0.85. As a result, PE index of proposed method using Hausdorff distance is the slowest, at 0.0000 whereas that of FCM is 0.2056. Hence, we believe that our method proposed is a novel technique to recognize the image in unsupervised learning. This method is completely viable to put in an application in real life.

From the Fig. 7, we can see that the probability of each image belonging to each clusters is an absolute perfection after 7 iterations. In terms of the first cluster, the probability of image from 1 to 72 stood at approximately 1. Therefore, the clustering effectiveness of the proposed algorithm which utilize fuzzy clustering technique is completely accurate. The second cluster and the third cluster are the same the cluster 1.

**Fig. 7.** The probability of each image belonging to three clusters.

**Example 2.** The second data set used in this example is made up of three different objects, namely Coca cola, Pottery pot, and Circle box. There are four clusters of equal sample size of 144 images. Some image samples are given in Fig. 8.

**Phase 1.** Finding a suitable number of cluster for 416 images.



**Fig. 8.** Three image samples: (a) Coca cola, (b) Pottery pot and (c) Circle box.

Unlike the first data set, the data set in this example is color image that required transformation to Gray scale. To similar performance the first example, we have data set with size  $416 \times 64$  after using GLCM technique to extract 64 features from image set including 416 images.

As we can see that the Table 4 shows the clustering result of the proposed method using three different distances. In terms of CR index, the proposed method products the best result ( $CR = 1$ ). After 117 iteration, the proposed method using Hausdorff distance take less time more other ones. It is meaningful to apply the proposed algorithm in reality because of saving computer cost as well as running time.

**Table 4.** The clustering result of the proposed method for 416 images

| Method       | Time (s) | Inter | CR index |
|--------------|----------|-------|----------|
| Proposed - E | 474.9752 | 427   | 1.000    |
| Proposed - C | 638.5893 | 929   | 1.000    |
| Proposed - H | 366.4745 | 117   | 1.000    |

## Phase 2. Determining the probability to assign each image to cluster for 416 images.

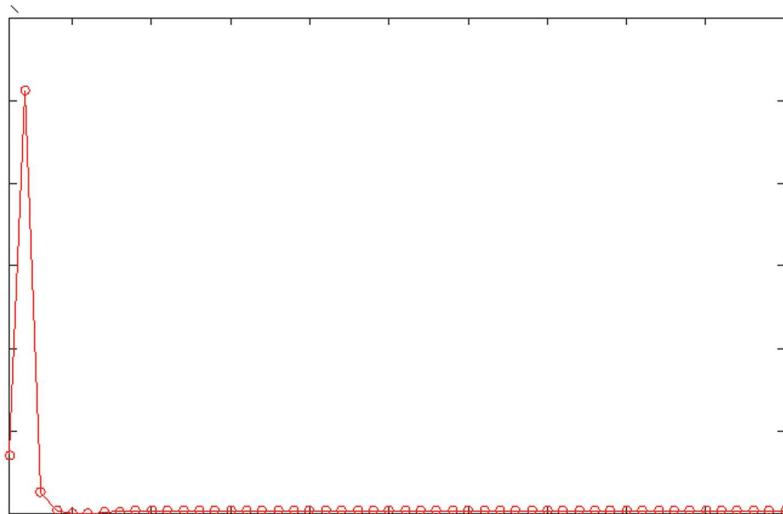
In this second Phase, we perform to find the probability to assign each image to cluster for 416 images through an improvement of vector of weight and objective function. We have Table 5 which presents data on differences in the outcome of the proposed method and FCM method.

Using Hausdorff distance, both AFCA and FCM method give better index than the remaining distances. In comparison with FCM method, the figure for the proposed method is higher, approximately  $PC = 1$  and  $PE = 0$ . These results indicate that the proposed method is a powerful in unsupervised clustering technique and highly viable for image recognition.

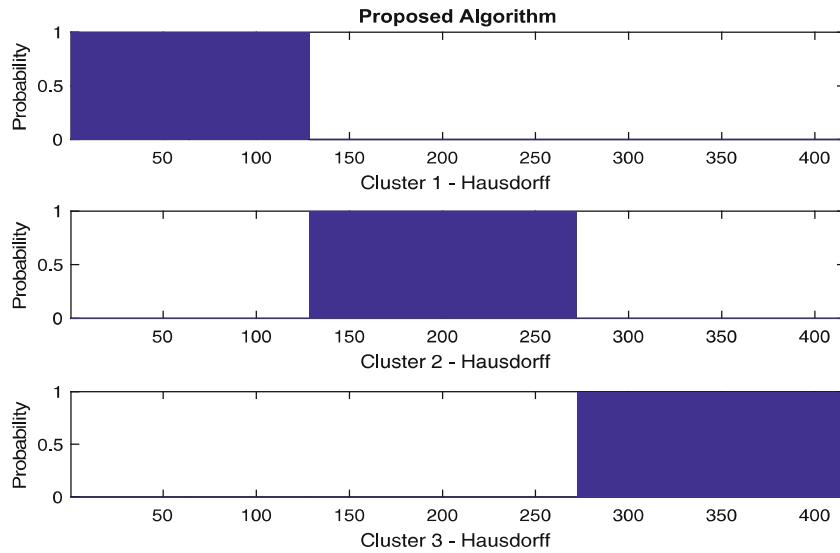
**Table 5.** The fuzzy clustering result of the proposed method for second data set

| Method   | No.cluster | PC index | PE index |
|----------|------------|----------|----------|
| FCM - E  | 3          | 0.9315   | 0.1478   |
| FCM - C  | 3          | 0.9471   | 0.1215   |
| FCM - H  | 3          | 0.9486   | 0.1122   |
| AFCA - E | 3          | 0.9839   | 0.0367   |
| AFCA - C | 3          | 0.9888   | 0.0282   |
| AFCA - H | 3          | 1.0000   | 0.0000   |

Figure 9 has shown that the proposed algorithm using the objective function converge from 5 out of 1000 iterations with  $f^{1000} = 395952663.722852$ .

**Fig. 9.** The convergence of the AFCA after 50 iterations.

The proposed method can determine not only the suitable number of clusters in the first phase but also the probability of assigning the image to the established clusters in the second phase. Like Example 1, the Fig. 10 shows that the fuzzy clustering outcome of AFCA based on the Hausdorff distance is completely accurate.



**Fig. 10.** The probability each image to belong to three clusters.

## 5 Conclusion

In this paper, we use Grey level Co-occurrence matrix to extract the image features, which is a crucial technique to recognize images in machine learning. Based on the Hausdorff distance, the proposed AFCA with the improvement about the vector of weight and objective function shows a clear effect with respect to the standard FCM technique, allowing for a substantial reduction in the running time as well as the quality of unsupervised clustering. The purpose of this method is to find the suitable number of cluster image based on center cluster to receive the accurate results instead of using experiment of researchers to choose the number of cluster. Moreover, the proposed technique also give the probability to assign each data set to cluster at the same time to ensure the quality of fuzzy clustering. The convergence of the proposed method is considered in theory and illustrated by the central clusters. The coming proposed approach will be continued studying to improve the computational cost and to apply the proposed algorithm in many practical problems associated with image recognition.

## References

1. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms, vol. 2981. Plenum Press, New York (2013)
2. Hoppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis: Methods for Classification Data Analysis and Image Recognition. Wiley, New York (1999)
3. Kabade, R.S., Gaikwad, M.S.: Segmentation of brain tumour and its area calculation in brain MR images using k-mean clustering and fuzzy c - mean algorithm. Int. J. Comput. Sci. Eng. Technol. **4**, 524–531 (2013)
4. Kavitha, P., Prabakaran, S.: A novel hybrid segmentation method with particle swarm optimization and fuzzy c-mean based on partitioning the image for detecting lung cancer. Int. J. Eng. Adv. Technol. **8**, 1223–1227 (2019)
5. Leung, T., Malik, J.: Representing and recognizing the visual appearance of materials using three-dimensional textons. Int. J. Comput. Vis. **43**, 29–44 (2001)

6. Schmid, C.: Constructing models for content-based image retrieval. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, vol. 2, pp. 11–39 (2001)
7. Goh, A.: Unsupervised riemannian clustering of probability density functions, pp. 377–392 (2008)
8. Haralick, R.M.: Statistical and structural approaches to texture. Proc. IEEE **67**, 786–804 (1979)
9. Shapiro, L., Haralick, R.: Computer and Robot Vision, vol. 2. Addison-Wesley, Reading (1992)
10. Celebi, E., Alpkocak, A.: Clustering of texture features for content-based image retrieval. In: Advances in Information Systems, pp. 216–225. Springer (2000)
11. Clausi, D.A.: An analysis of co-occurrence texture statistics as a function of grey level quantization. Can. J. Remote. Sens. **28**, 45–62 (2002)
12. Bhogle, P., Patil, S.: Oil spill detection in SAR images using texture entropy algorithm and mahalanobis classifier. Int. J. Eng. Sci **4**, 4823–4826 (2012)
13. Ayala-Ramirez, V., Obara-Kepowicz, M., Sinchez-Ygez, R.E., Jaime-Rivas, R.: Bayesian texture classification method using a random sampling scheme. In: IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 2065–2069 (2003)
14. Tai, V.V., Thao, N.T.: Similar coefficient of cluster for discrete elements. Sankhya B, Indian J. Stat. **80**, 19–36 (2019)
15. Chen, J.H., Hung, W.H.: An automatic clustering algorithm for probability density functions. J. Stat. Comput. Simul. **85**, 3047–3063 (2015)
16. Tai, V.V., Ha, C.N., Thao, N.T.: Clustering for probability density functions based on genetic algorithm. In: 1st International Conference on Applied Mathematics in Engineering and Reliability, pp. 51–57. CRC Press (2016)
17. Tai, V.V., Thao, N.T.: Cluster similar of cluster for probability density functions. Commun. Stat. - Theory Methods **47**, 1792–1811 (2017)
18. Thao, N.T., Tai, V.V.: Fuzzy clustering of probability density functions. J. Appl. Stat. **44**, 583–601 (2017)
19. Souza, R.M., Carvalho, F.D.A.: Clustering of interval data based on city-block distances. Pattern Recogn. Lett. **25**, 353–365 (2004)
20. Masson, M.H., Denoeux, T.: Clustering interval-valued proximity data using belief functions. Pattern Recogn. Lett. **25**, 163–171 (2004)
21. Hubert, L., Arabie, P.: Comparing partitions. J. classif. **2**, 193–218 (1985)
22. Tai, V.V.:  $L^1$  - distance and classification problem by Bayesian method. J. Appl. Stat. **44**, 385–401 (2017)
23. Setia, L., Teynor, A., Halawani, A., Burkhardt, H.: Image classification using cluster co-occurrence matrices of local relational features. Int. Work. Multimed. Inf. Retr. MIR **6**, 173–182 (2006)
24. Eleyan, A., Demirel, H.: Co-occurrence matrix and its statistical features as a new approach for face recognition. Turk. J. Electr. Eng. Comp. Sci **19**, 97–107 (2011)