Fingerprint Classification using KMCG Algorithm under Varying Window and Codebook Sizes

Winnie Gift Odongo
School of Computer Science and Information Technology
Jomo Kenyatta University of Agriculture and Technology (JKUAT)
Nairobi, Kenya

Waweru Mwangi
School of Computer Science and Information Technology
Jomo Kenyatta University of Agriculture and Technology (JKUAT)
Nairobi, Kenya

Richard Rimiru
School of Computer Science and Information Technology
Jomo Kenyatta University of Agriculture and Technology (JKUAT)
Nairobi, Kenya

ABSTRACT
Fingerprints are the most widely used form of biometric identification. Fingerprint identification has become time-consuming because of growing size of fingerprint databases. Fingerprint classification can be one of the significant preprocessing steps to improve the accuracy of fingerprint identification systems and is done to put a given fingerprint to one of the existing classes. Classifying fingerprint images is a very difficult pattern recognition problem, due to the possible problem with accuracy which is a measure of how well the system is able to correctly match the biometric information from the same person and avoid falsely matching biometric information from different people. In this research an experiment was conducted and a comparative analysis based on vector quantization for fingerprint classification using Kekre's Median Codebook Generation (KMCG) was done using codebook sizes 2, 4, 8 and window sizes 2*2, 4*4, 8*8, 16*16, 32*32, 64*64. KMCG is one of the better and faster vector quantization codebook generation methods. Fingerprint images were obtained from the National Institute of Standards and Technology (NIST) special database 4 for this study. It was observed that the method effectively improves the computation speed and provides accuracy of A (Arch) by 99%, TA (Tented Arch) by 98%, LL (Left Loop) by100%, RL (Right Loop) by 100% for codebook size 4 and LL (Left Loop) by 99% accuracy for codebook size 8 and window size 8*8. Codebook size 2, 4 exhibited overall better percentage accuracy of classification than codebook size 8.

Keywords
Vector Quantization, KMCG, NIST, Fingerprint Classification.

1. INTRODUCTION
Biometrics refers to the unique identification of an individual using biological and behavioral characteristics. The different human traits that can be used by a biometric system include the face, fingerprint, Iris, voice, speech, hand geometry and retina [1]. Today, because of the progress in computer processing, automatic biometric systems, based on concepts developed from time immemorial, have become available [2]. There are numerous benefits of the biometric system and how it affects various work sectors globally. Most biometric technology users face challenges in defining right and accurate biometric technology systems that are cost effective in solving a particular problem in a given environment [3].

A fingerprint biometric system remains a popular choice because a fingerprint is an individual's unique characteristic that remains unchanged during his or her lifetime [4]. The process of assigning a fingerprint to its predefined class/group is known as Fingerprint classification. Fingerprint classification involves extraction of features from a fingerprint image. These extracted features are then compared with other existing features from images in a database. The fingerprint image with features that exactly resemble it is retrieved [4]. Classification of Fingerprint has also received considerable attention as pattern recognition problem for its difficulty, due to the small inter-class variability and the large intra-class variability[2]. Large Intra- class variability happens when there is a broad selection of possible patterns with each class whereas small inter- class variability occur when fingerprint images from one class emerge very related to prints from another class [5].

The extremely increasing size of fingerprint samples for identification systems has really developed into an issue these days. Fingerprint classification for the grouping of fingerprints which may additionally play as the pre-processing of identification system has gained research drive. The job of assigning the fingerprint to one of the considered classes is tricky due to the likely problem with precision which is a gauge of how well the system is able to appropriately match the biometric information from the same person and avoid wrongly corresponding biometric information from dissimilar people. It is in light of these issues that the following study was conducted to investigate more on how to classify fingerprints and test for their accuracy with various codebook sizes [4].

One of the VQ codebook generation techniques is The KMCG which makes use of classification and median technique for generating codebook where image is divided into blocks and blocks are converted to the vectors of size k as given in this paper. The paper is prearranged as follows: Part II briefly describes a range of fingerprint classes, Part III gives the methodology that was used for fingerprint classification, Part IV explains regarding the KMCG algorithm, Part V consists of results and discussions and Part VI gives conclusion of the paper.

2. FINGERPRINT CLASSES
A fingerprint consists of ridges (raised skin) and a core point (the northernmost point of the inmost ridge). A fingerprint can therefore be defined as the pattern of ridges found on the exterior of one's fingertips. When analyzed a significant trait of the fingerprint known as the minutiae can be found. The minutiae include features such as ridge bifurcation (a ridge that further separates two ridges), curvature (arc) and termination. These regions are collectively refered to as

singularities and are further divided into three topologies: loop, delta, and whorl [6]. The universal form of ridges and furrows, further extended by their counts form the basis for fingerprint classification. The ridge patterns of the fingerprints can be systematically classified as loops, whorls and arches. Majority of fingerprint images fall into the loop category about 60-65% on the other hand arch and whorl comprise of 30-35% and 5% respectively [7]. These three classes can further be divided into the subsequent five classes as shown below:

a) Arch (A) b) Tented Arch (TA)

c) Left Loop (LL) d) Right Loop (RL)

e) Whorl (W)

Figure 2.0: Fingerprint Classes a)Arch(A), b)Tented Arch(TA), c)Left Loop(LL), d)Right Loop(RL), e) Whorl (W)

2.1 Loops

60-65% of fingerprint patterns encountered are Loops. One or more of the ridges enters on whichever side of the impression, re-curves, touches or crosses the line running from the delta to the core and terminates on or in the direction of the side where the ridge or ridges entered. Loops may specifically be classified as right loop and left loop by observing the left hand. If the ridges run in the direction of the thumb, it can be classified as right loop and if it runs in the direction of the small finger then it can be classified as left loop.

2.2 Arches

The arch pattern comes as aresult of ridges lying one over the other. The ridges come in on single side run or appear to flow out from the other side. The Tented Arch consists of at least one protruding ridge which tends to cut in half of other ridges at right angles. Plain Arch has a wave resembling structure as compared to tented arch with a sharp rise at the center.

2.3 Whorl

The Plain Whorl has at least one ridge and two deltas creating a complete circuit, which may be coiled, elliptical, spherical, or any variation of a circle, which may be spiral, oval, circular, or any variant of a circle. The double loop is made up of two detached loop formations, with two deltas and two separate and distinct sets of shoulders.

2.4 Applied VQ approaches in Fingerprint Classification

A technique of compressing data based on grouping blocks having comparable data is referred to as Vector Quantization. These blocks are called Code Vectors and all the code vectors grouped together is called a Codebook. The Key to VQ data compression is a good codebook. There is a loss of quality while using VQ, but this is duly compensated by the important savings achieved by this compression method. VQ leads to creation of codebooks. These codebooks are a division of the blocks resulting from the data. It is an iterative method of grouping data, where iteration involves increasing the number of groups twofold and re-grouping the data till a finite desired number of clusters is reached. It is a three phase procedure involving Codebook Generation, Encoding and Decoding [8].

2.5 Algorithms for codebook generation

Linde-Buzo-Gray (LBG) Algorithm: In this algorithm centroid is computed as the first code vector for the training set. Two vectors V_1 and V_2 are then generated by adding constant error to the code vector. Euclidian distances of all the training vectors are computed with vectors V_1 and V_2 and two clusters are formed based on nearest of V_1 and V_2. This procedure is repeated for every cluster [7], [8].

Kekre's Proportionate Error (KPE) Algorithm: In this algorithm a proportionate error is added to the centroid to generate two vectors V_1 and V_2. The error ration is decided by the magnitude of coordinates of the centroid. Thereafter, the procedure is same as that of LBG [8].

Kekre's Fast Codebook Generation (KFCG) Algorithm: This algorithm reduces the time for codebook generation. It does not use Euclidian distance for codebook generation. In this algorithm image is divided into blocks and blocks are converted to the vectors of size k. Initially only one cluster with the entire training vectors and the code vector C_1 which is centroid. In the first iteration of the algorithm, the clusters are formed by comparing the first element of training vector with first element of code vector C_1. The vector X_i is grouped into the cluster 1 if $X_{i1} < C_{11}$ otherwise vector X_i is grouped into cluster 2. In second iteration, the cluster 1 is split into two by comparing second element X_{i2} of vector X_i belonging to cluster 1 with that of the second element of the code vector which is centroid of cluster 1. Cluster 2 is split into two by comparing the element X_{i2} of vector X_i belonging to cluster 2 with that of the second element of the code vector which is centroid of cluster. This procedure is repeated till the codebook size is reached to the size specified by user [8], [9].

Kekre's Error Vector Rotation (KEVR) Algorithm: This algorithm creates a VQ by using an error vector. This error vector sequences are created by taking the binary representation of numbers from 0 to k-1 where k is the number of iterations. The 0s in the binary representations are replaced by 1 and the 1s are replaced by -1. This algorithm takes time to compute as it uses mean square distances to compare vectors [4].

Kekre's Fast Codebook Generation (KFCG) Algorithm: This algorithm reduces the time for codebook generation. It makes use of sorting and median technique for generating codebook. In this algorithm image is divided into blocks and blocks are converted to the vectors of size k. Once the codebook is generated, the feature vector for each of the fingerprint classes is computed. The feature vector is calculated by taking the mean of all images in each cluster e.g. in cluster 1 the feature vector is the mean of all images in cluster 1. In this way the feature vector is calculated for all fingerprint classes and stored separately. This feature vector now becomes the identity of each of the fingerprint classes and the images are tested by checking the Mean Squared Error (MSE) with the feature vectors of each of the classes [8], [10] .

3. METHODOLOGY

3.1 Introduction

In order to investigate the use of code book size 2, 4, 8 and window sizes 2*2, 4*4, 8*8, 16*16, 32*32, 64*64 to classify fingerprints, the NIST special database 4 was selected as the tool for data collection.

NIST Special Database 4, contains 8-bit gray scale images of randomly selected fingerprints. The database is being distributed for use in the development and testing of automated fingerprint classification systems on a common set of images.

The CD-ROM contains 4000 (2000 pairs) fingerprints stored in NIST's IHead raster data format and compressed using a modified JPEG lossless compression algorithm. Each print is 512 X 512 pixels with 32 rows of white space at the bottom of the print [11]. Approximately 636 Megabytes of storage are needed when the prints are compressed whereas 1.1Gigabytes are needed when uncompressed (1.6: 1 average compression ratio).The fingerprints are classified into one of five categories (L = left loop, W = whorl, R = right loop, T = tented arch, and A = arch) with an equal number of prints from each class (400). Each filename contains a reference to the hand and digit number so the classes can be converted to other classification techniques (i.e. radial and ulnar). All classes are stored in the NIST IHead id field of each file, allowing for comparison with hypothesized classes. An advantage of this database is that the fingerprints have already been processed so all that is remaining is to test them with this model [12] .

In order to perform the experiment, two test cases were analyzed, and their results compared. For the test cases of code book size 2, 4 and 8, 105 random grey fingerprint images from NIST special database 4 were taken as a training set of input images per class e. g Figure 3.1

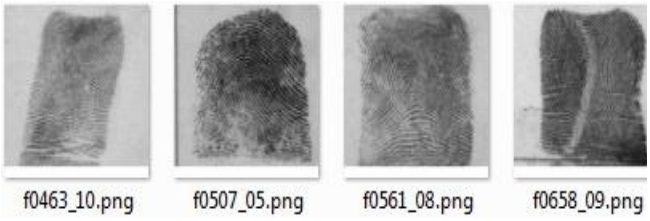

Figure 3.1: Sample of training images extracted from NIST special database 4

Source: http://www.nist.gov>srd>nist-special...

3.2 Procedure for collecting fingerprints

The procedure for collecting these fingerprints focused on the following aspects. Collecting 105 fingerprint images from the NIST special database 4 per class e.g. 105 Arch images and putting them in a folder named Arch. Training at least 5 images belonging to the same class in the classifier to improve the number of images that will correctly be classified. Testing the images and calculating the number of images classified correctly where,

$$\%\ accuracy\ of\ classification = \frac{No.of\ images\ successfully\ classified}{No.of\ classification\ attempt}\ x\ 100$$

3.3 The design of the classifier

The classifier is a supervised learner. It is first trained with the representative vectors for each class, and only after this can it be used for classification.

Note: There are five fingerprint classes, which are, Arch, Tented Arch, Right Loop, Left Loop and Whorl.

The classifier stores its knowledge in a knowledge-base. This knowledge-base is a hash map of names of each class, against their representative feature vectors.

An example knowledge-base, with a widow size of three = Map (

"Arch": [[98,100,145],...],

"Tented Arch": [[100,145,100],...],

"Right Loop": [[140,141,45],...],

"Left Loop": [[100,101,107],...],

"Whorl": [[99,47,50],...]

)

The representative feature vectors are determined in the training process. Training is done by determining a 'perfect' representative fingerprint sample for each class, and then computing the codebook for this sample, and setting this codebook against the respective class name in the knowledge-base.

To classify a test fingerprint sample whose class is initially unknown, we first compute its codebook, and then we compute its mean square error against all the five representative feature vectors in the knowledge-base, and the class against with the smallest mean square error is recorded is approximated to be the class of that test fingerprint sample.

This is a diagram of how the classifier works:

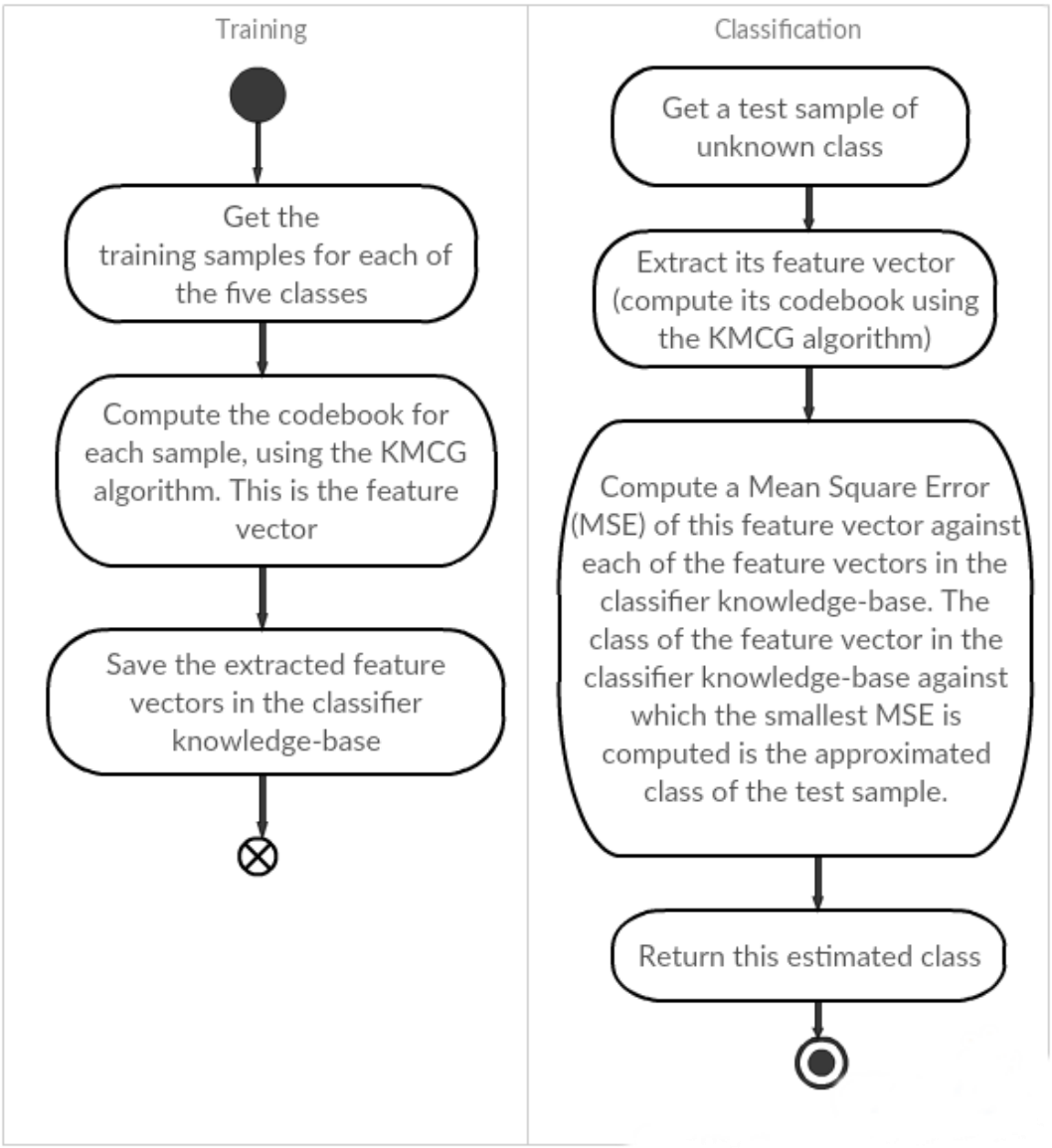

Fig 3.2: Diagram explaining how the classifier works

4. KEKRES MEDIAN CODEBOOK GENERATION ALGORITHM

4.1 A description of the codebook generation algorithm KMCG

It starts with a digital representation of the image, in the RGB color model.

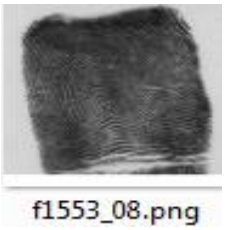

Figure 4.1: Sample of a grey scale fingerprint image extracted from NIST special database 4

Though the RGB color model is used, our fingerprints are grayscale, and only one color channel is used. Therefore, each pixel is represented by a single integer instead of a 3-tuple of integers. Each pixel value, p, is 0 <= p <= 255. The image is represented by a two-dimensional array, with each sub-array representing all the pixels in a particular image scan line.

Note: An image can be viewed as a collection of stacked up rows of pixels. Each of these rows is called a scan line. These scan lines are stacked up together like the way plates get stacked up on top of each other.

The image representation, IR, is:

IR = [

[98,100,234...],

[100, 106, 54...],

[35, 99,100...],

...

]

From IR, we derive the initial sequence I of integers, each representing a pixel.

IR is converted to a one-dimensional array, whose elements are all the elements of the sub-arrays.

That is, I = [98, 100, 234…].

I is then windowed/paged into sub-arrays of the size specified by the window size. This new set of sub-arrays of size <window size> forms the set of training vectors, T.

That is, assuming we have window size of 3, then

T = [

[98,100,234],

[100, 106, 54],

[35, 99,100],

…

]

T is then sorted with respect to the first element of each training vector, using the Quicksort sorting algorithm.

That is, after sorting with respect to the first element of each sub-array (vector),

T = [

[35, 99,100],

[54,100,106],

[98,100,234],

…

]

The median of the sorted T is then picked, and added to the codebook, and the codebook size is increased by one. T is then partitioned into two partitions of equal size. Each of these two partitions is then sorted with respect to their second member, using the Quicksort sorting algorithm.

Their medians are then picked, and added to the codebook, such that the codebook now has three vectors, and its size is increased by one to two. These two partitions are then partitioned into two partitions each, so that now we have four partitions. These partitions are then sorted by their third element, and this process is repeated until a codebook of the required size is achieved. Kekre's Median Codebook Generation algorithm is a codebook generation algorithm, and hence the final result is this codebook.

An example of a codebook size 4 generation process, with a 4x4 grayscale image:

```
Begin Program
 //a sample representation of the initial
 //array representation of the image
 let Ir =
  [[1   2   3   4]
   [5   6   7   8]
   [9   10  11  12]
   [13  14  15  16]]
 //block or window or page or chunk the array
 //into block of window size 2 x 2
 let Ib =
  [[[1 2]
    [5 6]]
   [[3 4]
    [7 8]]
   [[9 10]
    [13 14]]
   [[11 12]
    [15 16]]]
 //convert the blocks into 1-dimensional vectors
 let Il =
  [[1 2 5 6]
   [3 4 7 8]
   [9 10 13 14]
   [11 12 15 16]]
 Begin Proc get_codebook(Il, window_size, codebook_size):
  Begin Proc partition_clusters(clusters, window_size):
   Begin Proc partition_cluster(cluster):
    //sort the cluster based on the first element
    cluster.sort(axis=1)
    let median = median(cluster)
    //initialize the two new cluster
    let cluster_0 = []
    let cluster_1 = []
    for i=0l i < length(cluster); i++:
     let vector cluster[i]
     if (vector[i] < median[i]):
      cluster_0 += vector
     else:
      cluster_1 += vector
    return {
     median: median,
     clusters: [cluster_0, cluster_1]
    }
   End Proc
   //initialize storage for the new clusters
   let new_clusters = []
   //initialize storage for the codebook
   let cb = []
   for i = 0; i < length(clusters); i++:
    let result = partition_cluster(clusters[i])
    new_clusters += result['clusters']
    cb += result['median']
```

```
    return {
      codebook: cb,
      clusters: new_clusters
    }
  End Proc
  let next_cluster = partition_clusters(Il, window_size)
  Begin Proc iterate():
    let iterated_codebook = []
    for i=0; i < codebook_size; i++:
          //it_result => the result of the iteration
      it_result = partition_clusters(next_cluster, window_size)
      iterated_codebook += it_result['clusters']
      next_cluster = it_result['clusters']
    return iterated_codebook
  End Proc
  let codebook = iterate()
  return codebook
End Proc
End Program
```

4.2 The mean Square Error is Computed as follows

The mean square error is the mean of the sum of the squares of the errors/deviations of one value against another. That is, given values V_1 and V_2, then their MSE = E (|V1 - V2|2), where E is the Expectation (mean).

This case involves finding the difference between two 2-dimensional arrays (matrices), and squaring those differences, and then finding the mean of those squared errors/deviations element-wise. Assuming these matrices are A = a_{ij} and B = b_{ij}, then their difference,

$$d(a, b) = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} - b_{ij})^2}$$

The result is another 2-dimensional array (matrix). The mean value of this matrix is then calculated, by summing up all the elements of the matrix, and dividing this sum by the number of all the elements in the matrix. The result is a single non-negative integer value, which is the MSE of the matrix A against matrix B. How this MSE is applied this research is illustrated in fig3.2: The classifier design.

5. RESULTS AND DISCUSSIONS

5.1 Results

The results in this section were guided by the percentage accuracy of classification which is the number of fingerprints images successfully classified over the number of classification attempt times 100.

$$= \frac{No.of\ images\ successfully\ classified}{No.of\ classification\ attempt}\ x\ 100$$

The results were noted down as follows: codebook size 2, 4 and 8 with window sizes 2*2, 4*4, 8*8, 16*16, 32*32, 64*64 as shown in the tables and graphs below. The tables show the % accuracy of fingerprints images successfully classified over the number of classification attempts.

Table 5.1: Percentage Accuracy of Fingerprint Classification for Codebook size 2 with varying window sizes using KMCG

	codebook size = 2					
ws	A%	TA%	LL%	RL%	W%	Avg.
2	70	90	96	84	71	82.2
4	43	88	95	86	88	80
8	53	91	94	88	89	83
16	86	98	100	100	98	96.4
32	73	92	96	88	83	86.4
64	73	90	95	84	67	81.8

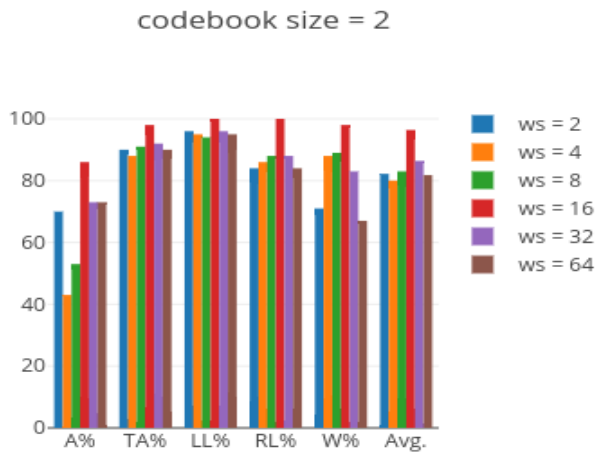

Figure 5.1: % Classification Accuracy for Codebook size 2 and respective pixel window sizes

Codebook size 2 and window size 16*16 for the LL and RL class classified all the fingerprints correctly i.e. LL=100% and RL=100%. Codebook size 2 and window size 16*16 recorded 98% for the TA and W class respectively which means that only 1 fingerprint was not correctly classified in these attempts and these were the best results for codebook size 2. Arch class recorded poor classification for codebook size 2 with Window size 4*4 computing 43%.

Table 5.2: Percentage Accuracy of Fingerprint Classification for Codebook size 4 with varying window sizes using KMCG

	codebook size = 4					
ws	A%	TA%	LL%	RL%	W%	Avg.
2	76	89	95	83	71	83.4
4	48	87	94	85	87	81
8	54	91	94	88	89	83.2

ws	codebook size = 4					
16	99	98	100	100	98	99
32	76	92	96	88	83	87
64	70	89	95	84	67	81.2

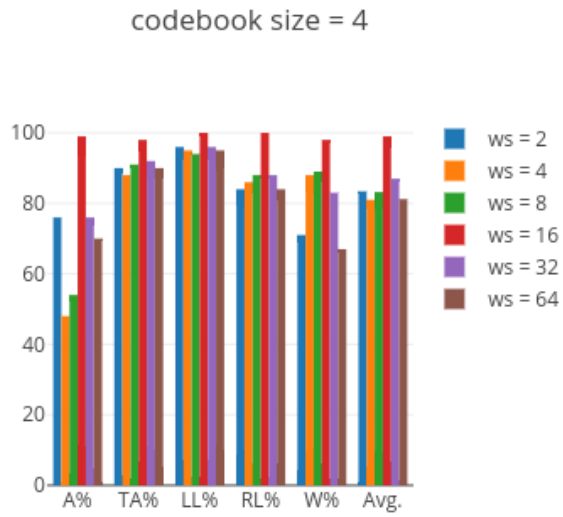

Figure 5.2: % Classification Accuracy for Codebook size 4 and respective pixel window sizes

Codebook size 4 and window size 16*16 recorded high percentage accuracy levels where A=99%, TA=98%, LL=100%, RL=100% and W=98% respectively. Codebook size 4 and Window size 4*4 for the A class recorded the lowest percentage accuracy of classification which was 48%. It was deduced that window size 16*16 for codebook size 4 yielded the best results for this experiment.

Table 5. 3: Percentage Accuracy of Fingerprint Classification for Codebook size 8 with varying window sizes using KMCG

ws	codebook size = 8					
	A%	TA%	LL%	RL%	W%	Avg.
2	62	90	90	82	63	77.4
4	41	86	89	79	73	73.6
8	46	82	95	78	71	74.4
16	64	84	99	72	69	77.6
32	68	86	98	74	67	78.6
64	53	85	97	70	63	73.6

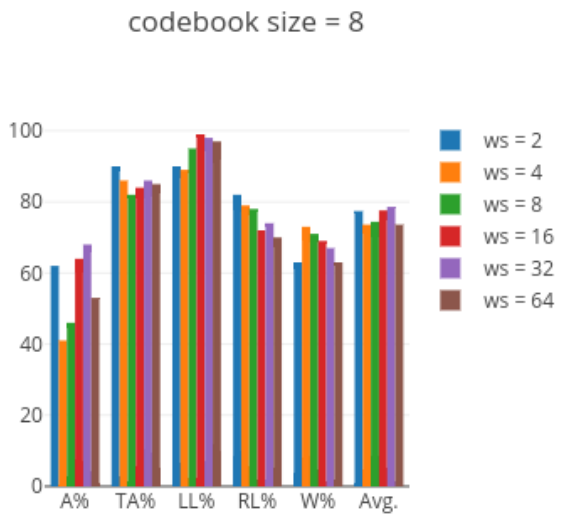

Figure 5.3: % Classification Accuracy for Codebook size 8 and respective pixel window sizes

Codebook size 8 and window size 16*16 for the LL class recorded 99%, percentage accuracy of classification which was the highest result in this category. Codebook size 8 and Window size 4*4 for the A class recorded the lowest percentage accuracy of classification which was 41%.

5.2 Discussions

Discussion of results for Fingerprint classification using KMCG with codebook size 2, 4, 8 and window sizes 2*2, 4*4, 8*8, 16*16, 32*32, and 64*64.

When interpreting and discussing proportion accuracy of classification, it should be noted that 100 percent proportion accuracy of classification means perfect while zero percent classification means entire classification failure. Fewer number of fingerprint images correctly classified over a high number of classification attempts will produce poor percentage accuracy of classification. Part 5.1 discussed the outcome illustrated in Figure 5.1, 5.2, 5.3. This study observed that the bars in place of percentage accuracy of classification extended longer for both codebook size 2 and 4.

This meant that the smaller codebook sizes i.e. 2, 4 exhibited overall better percentage accuracy of classification than codebook size 8. From the graphs we also inferred that, the larger the codebook size the lower the percentage accuracy of classification is achieved. The lowest percentage accuracy of classification recorded for codebook 2, 4, 8 and window size 4*4 could be due to the lower number of pixels in a 4*4 window size fingerprint image.

KMCG gave better performance for Codebook size 4 and window size 16*16 recording higher percentage accuracy levels where A=99%, TA=98%, LL=100%, RL=100% and W=98% respectively this is contrasting with conducted experiments when classifying fingerprints with KFCG which showed that codebook size 8 gave better performance with higher pixel window size .Still on the findings of KFCG, lower window sizes such as codebook size 4 generated using pixel window of size 7x7 gave the best performance [9]. From the graphs generated above, it is clearly observed that poor results are generated for window size 4*4 for codebook size 8, 4 and 2 which are 41%, 48% and 43 respectively for the A class. This indicated that smaller window sizes return poor results with the A class for all the codebook sizes. This research also showed that KMCG worked well with the LL class which recorded excellent performance for all codebook

size 2, 4 and 8.This concurs with previous findings on KMCG proven to work best for LL and gave poor results for the A class [10].

6. CONCLUSION

Classification is a significant task for the achievement of any computerized fingerprint Identification System. Fingerprint Classification Using KMCG Algorithm under Varying Window and Codebook Sizes provides for recording higher proportion accuracy levels for the LL and RL class for Codebook size 2 and 4 and window size 16*16. It is computationally fast since it does not need any distance calculation. Future work entails testing the proposed approach on a huge database and making it more competent by improving its level of accuracy further.

7. REFERENCES

[1] H. B. Kekre, U. Thapar, and N. Parmar, "Human Ear Identification using Vector Quantization Algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 2, no. 12, pp. 4542–4547, 2013.

[2] D. Michelsanti, Y. Guichi, A. Ene, R. Stef, K. Nasrollahi, and B. Moeslund, "Fast Fingerprint Classification with Deep Neural Network," in *International Conference on Computer Vision Theory and Applications*, 2017.

[3] A. S. Falohun, O. D. Fenwa, and F. A. Ajala, "A Fingerprint-based Age and Gender Detector System using Fingerprint Pattern Analysis," vol. 136, no. 4, pp. 43–48, 2016.

[4] S. Thepade, V. Murthi, and B. Shah, "Fingerprint Classification using KEVR Algorithm," *Int. J. Comput. Appl.*, vol. 45, no. 18, pp. 5–7, 2012.

[5] N. Yager and A. Amin, "Fingerprint classification : a review," *Springer-Verlag London Ltd. 2004*, pp. 77–93, 2004.

[6] K. S. Sim, Y. K. Tan, M. E. Nia, and G. D. Lee, "Rotation-invariant Reference Point Location Detection Using Complex Filtering for Fingerprint Matching," *Int. J. Futur. Comput. Commun.*, vol. 1, no. 3, pp. 321–322, 2012.

[7] S. Thepade, D. Parekh, U. Thapar, and V. Tiwari, "LBG ALGORITHM FOR FINGERPRINT CLASSIFICATION," *Int. J. Adv. Eng. Technol.*, vol. 5, no. 1, pp. 430–435, 2012.

[8] H. B. Kekre, "Performance Comparison of LBG , KPE , KFCG and KMCG for Global Codebook Technique," *Int. J. Comput. Appl.*, vol. 30, no. 10, pp. 42–50, 2011.

[9] H. B. Kekre, S. D. Thepade, and D. Parekh, "Comparison of Fingerprint Classification using KFCG Algorithm with Various Window Sizes and Codebook Sizes," *IJCSNS Int. J. Comput. Sci. Netw. Secur. VOL.13 No.3, March 2013*, vol. 13, no. 3, pp. 60–63, 2013.

[10] S. Thepade, D. Parekh, J. Shah, B. Shah, and P. Vora, "Classification of Fingerprint using KMCG Algorithm," *Int. J. Sci. Technol. Res.*, vol. 1, no. 6, pp. 105–107, 2012.

[11] R. Wang, C. Han;, and T. Guo, "In 2016 23rd International Conference on Pattern Recognition (ICPR).," in *A Novel Fingerprint Classification Method Based on Deep Learning.*, 2016, pp. 931–936.

[12] C. I. Watson and C. L. Wilson, "NIST Special Database 4," *NIST Spec. Database 4*, pp. 1–14, 1992.