Efficient K-Nearest Neighbor Searches for Multiple-Face Recognition in the Classroom based on Three Levels DWT-PCA

Hadi Santoso
Department of Computer Science and Electronics,
Faculty of Mathematics and Natural Science
Universitas Gadjah Mada
Yogyakarta, Indonesia
Information System Study Program
STMIK Atma Luhur
Pangkalpinang, Indonesia

Agus Harjoko, Agfianto Eko Putra
Department of Computer Science and Electronics,
Faculty of Mathematics and Natural Science
Universitas Gadjah Mada
Yogyakarta,
Indonesia

***Abstract*—The main weakness of the k-Nearest Neighbor algorithm in face recognition is calculating the distance and sort all training data on each prediction which can be slow if there are a large number of training instances. This problem can be solved by utilizing the priority k-d tree search to speed up the process of k-NN classification. This paper proposes a method for student attendance systems in the classroom using facial recognition techniques by combining three levels of Discrete Wavelet Transforms (DWT) and Principal Component Analysis (PCA) to extract facial features followed by applying the priority of k-d tree search to speed up the process of facial classification using k-Nearest Neighbor. The proposed algorithm is tested on two datasets that are Honda/UCSD video dataset and our dataset (AtmafaceDB dataset). This research looks for the best value of k to get the right facial recognition using k-fold cross-validation. 10-fold cross-validation at level 3 DWT-PCA shows that face recognition using k-Nearest Neighbor on our dataset is 95.56% with k = 5, whereas in the Honda / UCSD dataset it is only 82% with k = 3. The proposed method gives computational recognition time on our dataset 40 milliseconds.**

***Keywords*—Multiple-face recognition; DWT; PCA; priority k-d tree; k-Nearest Neighbor**

I. Introduction

Facial recognition performance is influenced by several variables, including pose, expression, lighting, and occlusion (namely, glasses, mustaches, beards, headgear, etc.) [1].

Many educational institutions in Indonesia continue to use attendance sheets, allowing students to cheat by asking friends to sign their names. The use of attendance sheets has proven to be time-consuming, unreliable, inaccurate, and inefficient. Based on this issue, we propose a combination of face recognition methods using three levels of Discrete Wavelet Transforms (DWT) and Principal Component Analysis (PCA) to extract facial features followed by applying k-d tree to accelerate the process of facial classification using k-NN. The k-NN process with Euclidean distance is used to identify students' presence through pictures or video of a student's face. By using this approach, it is possible that automatically integrated systems between IP camera and pc or notebooks will determine whether each student is present or absent and will be recognized in the group of courses followed.

The contibutions of this paper include: 1) combining three levels of 2D DWT-PCA for feature extraction; 2) we adapted approximate nearest neighbors search algorithm using the priority of k-d tree to find some (k) nearest neighbors from a certain query point (q) efficiently [2]

The paper is structured as follows: Section 2 consists of related research discussions, Section 3 consists of the proposed research method, Section 4 presents the results of research and analysis, and Section 5 offers some conclusions.

II. Related Works

Human facial recognition plays an important role in biometrics. The eigenvector-based method of facial recognition was first introduced by [3], and then expanded upon by [4] and [5]. The eigenvector-based method extracts the low-dimensional subspace, which tends to simplify the classification process. Chitaliya and Trivedi [6] developed a facial recognition model that used wavelet-PCA for feature extraction, then used Euclidean distance and neural networks for classification. In this study, Level 1 discrete wavelet transform (DWT) was used. They were able to attain up to 93.3% accuracy.

In 2012, Rao [7] proposed a facial recognition system using discrete wavelet transform (DWT) and eigenvectors, showing an average of 3.25% improvement in recognition performance.

Research on attendance management in the classroom proposed by [8], using the haar cascade method for face detection. While the face recognition using eigenface method. This face recognition approach achieves an accuracy of 85%.

Research on improving facial recognition was proposed by [9], who used a wavelet-PCA decomposition method with mahalanobis classification. The results of this study improved recognition by 95.7%. Recognition results using Euclidean classification reached 93.8% accuracy, with a computing speed of 8.501 milliseconds.

In 2014, Mao [10] conducted research into multiple face detection, tracking, and recognition in the classroom using a Honda/UCSD dataset of videos. In the experiment, the student dataset consisted of 16 and 39 people in the classroom. Fifty-nine videos were used: 20 for training and 39 for testing. The study used the Haar cascade facial detection method. For facial recognition, this study used eigenfaces, LBP+K-mean. The average precision and recall values in this study were more than 90%.

The combination of methods for student attendance systems in the classroom was proposed by [11] using facial recognition techniques by incorporating Discrete Wavelet Transforms (DWT) and Discrete Cosine Transform (DCT) to extract the facial features of students followed by applying Radial Basis Function (RBF) to classify face objects. The success rate of the proposed system in recognizing facial images of students who sit in the classroom is about 82%.

In 2017, Sayeed [12] presented an automated attendance monitoring system with face recognition in a real-time background world for with a database of student's information by using Principal Component Analysis (PCA) algorithm. The testing results have been tested and taken from few different environment backgrounds. Basically is during the day and night time with lights either on or off. Average successful rate of the proposed system are about 2.43 to 2.81.

K-nearest neighbor (k-NN) is a simple and effective classification method. Samet [13] proposed a k-nearest neighbor algorithm using MaxNearestDist, while the k-NN repair method had been proposed by [14]-[16]. The weakness of k-NN algorithm, namely the process of calculating the similarity to be done on all existing training data [17]. If training data increases in number, the time for classification will also increase proportionately. The problem can be solved if using k-d tree data structure [18]-[21]. It can be used with a k-nearest neighbor (k-NN) approach to match facial features efficiently and search for the location of the nearest neighbors. K-d tree data structure has been used as a data structure for overcoming increased processing times caused by the addition of features into the database. An alternative approach is establishing a balanced k-d tree, as proposed by [22]. The nearest neighbor search algorithm using the k-d tree data structure can be found in [23]. In their research, they only found one nearest neighbor. We integrate the priority k–d tree search and best bin first method (BBF) to find the nearest neighbors, and the Euclidean distance is utilized as similarity measure.

In our study, the nearest proposed neighbor search algorithm uses a data structure called a priority queue that stores the list of closest neighbors k with some distance to the query point q. The priority queue has a fixed upper limit on the number of elements (or points) that can be stored, which is the number of nearest neighbors k. Each time a new element is added to the queue, if the queue is at a predetermined capacity, the element with the highest priority value (the longest distance) is removed from the queue.

III. Proposed Method

In order to make the framework, the proposed method in this study used was the method from related research and then we developed them to be tested on our dataset. The research method used in this article consisted of several stages, as shown in Fig. 1.

A. Face Database Training Stage

1) Single Face Image

In this stage, facial images of students were captured using a digital camera. Facial data from 1,014 individuals was collected; each individual had nine images taken from different angles. A set of student facial images as training data is presented in Fig. 2.

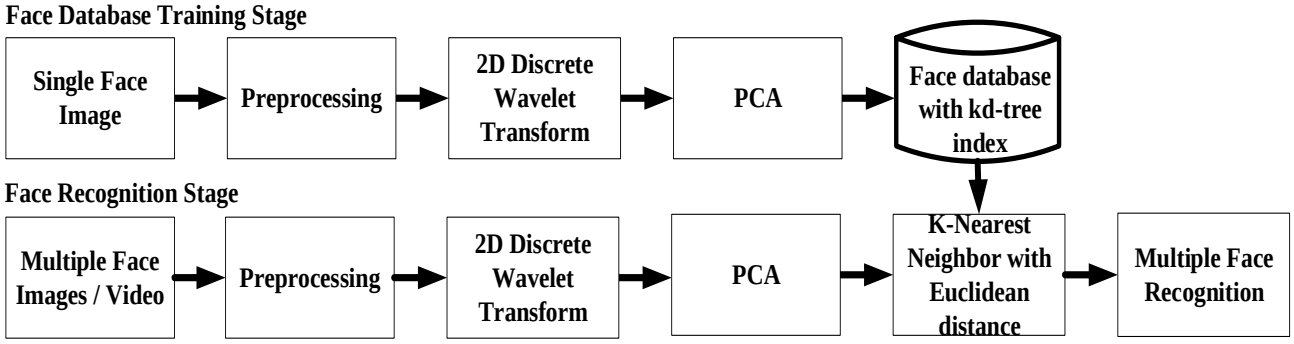

Fig. 1. The proposed method.

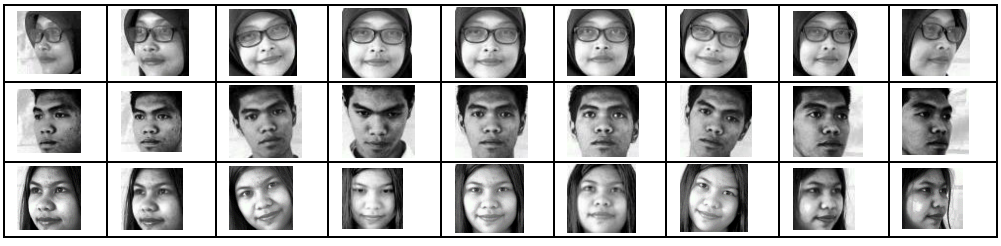

Fig. 2. A set of student facial images as training data.

2) Preprocessing

In the preprocessing stage, the following steps were conducted:

a) Facial cropping, the detection and localization of the face in a square area using the Viola-Jones method [24]

b) Changing the facial image from RGB to grayscale mode.

c) Resizing the facial image to 128x128 pixels.

d) Normalizing color brightness using a histogram equalization process.

The preprocessing stage of this research is shown in Fig. 3.

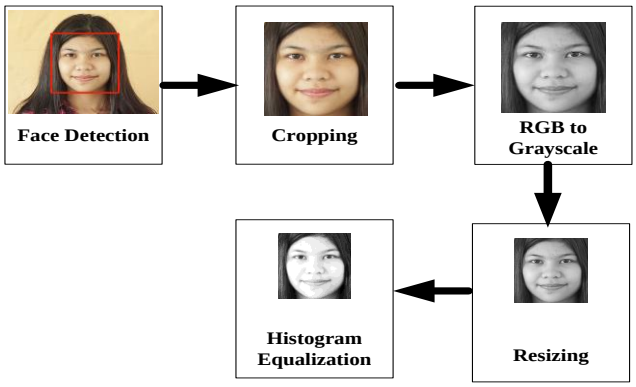

Fig. 3. Preprocessing stage.

The Viola-Jones method was used to detect faces in the images taken by digital cameras. Facial images were then stored in the database. After the detection process, images of students' faces were manually registered in the database by name and identity code. The process of registering students' faces is shown in Fig. 4.

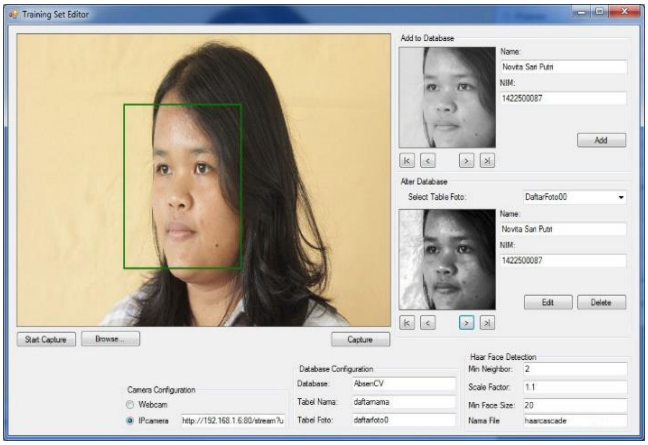

Fig. 4. The process of registering students' faces.

3) 2D Discrete Wavelet Transform

Wavelets are defined as small or short waves. The wavelet transform converts a signal into a series of wavelets. This is the basic function at a different time. Wavelets result from the scaling function. Certain wavelets are also called mother wavelets, as other wavelets result from their scaling, dilation, and shift.

Discrete Wavelet Transform (DWT) is a discrete form of the wavelet transform, consisting of a signal sampling process based on the scaling and shifting of parameters [25]. It is systematically defined by the following equation [26]:

$$DWT_{x(n)} = \begin{cases} d_{j,k} = \sum x(n)h(n-2^j k) \\ a_{j,k} = \sum x(n)g(n-2^j k) \end{cases} \quad (1)$$

The coefficient $d_{j,k}$ refers to the detailed component of the signal $x(n)$ and it is suited to the function of the wavelet, while $a_{j,k}$ refers to the approximation components of the signal. The functions $h(n)$ and $g(n)$ are the coefficients of high-pass and low-pass filters, respectively, while parameters j and k show the scale of wavelets and translational factors.

In this research, the mother wavelet used for feature extraction is the Haar wavelet. The image produced through preprocessing is decomposed at three levels. At each level, DWT was first performed in a vertical direction, followed by a horizontal direction. After the first level of decomposition, four sub-bands were obtained: LL1, LH1, HL1, and HH1. For each level of decomposition, the sub-band LL from the previous level was used as the input. The sub-band LL1 was only used for DWT calculation in the next scale. To calculate the wavelet features in the first stage, the wavelet coefficients were calculated on sub-band LL1 using a Haar wavelet function. For the second level of decomposition, DWT was applied to band LL1 by decomposing it into four sub-bands: LL2, LH2, HL2, and HH2. For the third level of decomposition, DWT was applied to band LL2 by decomposing it into four sub-bands: LL3, LH3, HL3, and HH3. LL3 contained the low-frequency band, while LH1, HL1, and HH1 contained the high-frequency band.

The face image size was 128x128 pixels obtained from the pre-processing results was then decomposed to 64x64 pixels on the wavelet level 1, then the 64x64 pixel face image was decomposed to 32x32 on the wavelet level 2, and the last was the 32x32 pixel face image decomposed using wavelets level 3 to 16x16 pixels. These three levels of facial image decomposition were used in the experiment to find the right size for the recognition process. Experiments were performed on all three levels of wavelet decomposition to find the right level of accuracy of recognition of many faces. Third face image of wavelet decomposition process will be processed by PCA (principal component analysis). The illustration of the 3-level 2D-DWT decomposition process is shown in Fig. 5.

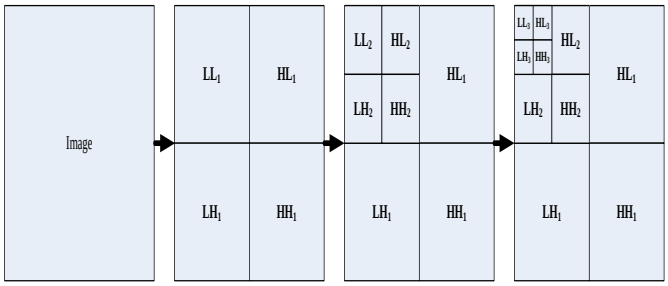

Fig. 5. The Illustration of three-level 2D-DWT decomposition.

4) Principal Component Analysis

Imagery produced through three-level 2D-DWT decomposition was subjected to the PCA process, as shown in Fig. 6. This is very significant for feature extraction, as it reduces dimensions and thus reduces the complexity of computation.

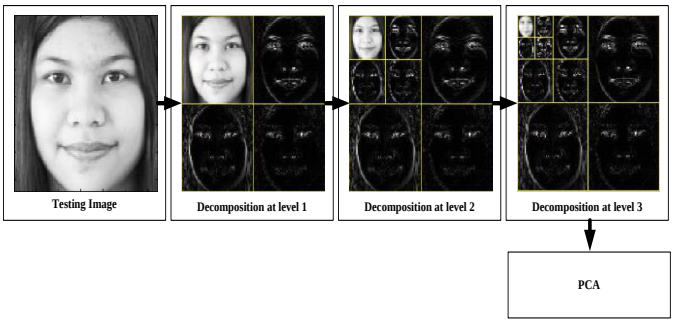

Fig. 6. Three level 2D DWT and PCA decomposition process.

PCA was used to obtain vectors, also known as principal components that could provide information regarding the maximum variance in the facial database. Each principal component is a representation of a linear combination of all training face images that have been reduced by the image mean. Combinations of facial images are known as eigenfaces; these facial images are those that will be recognized.

PCA aims to find the principal components of faces collected in the database. The significant features are termed eigenfaces and obtained from the eigenvector—a feature that describes the variance between face images—of a covariance matrix of images in the database. Each location on the face contributes to each eigenvector. Each face in the database can be represented by a linear combination of these eigenfaces. The number of eigenfaces is the same as the number of faces stored in the database. Facial images can be identified with the best eigenface (i.e. that which has the largest eigenvalue) that has most of the facial variations in the database. By taking some of the principal (most important) components, the calculation of eigenfaces may be efficient [27]. The steps to identify eigenfaces are as follows [28]:

a) Determine the dimension of the matrix of facial images which will be used.

b) Arrange the matrix of facial images into the vector of column Ψ with the size m × n

c) Diminish every facial image Γ_i by the average of matrix Ψ and store the result in variable Φ_i

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \tag{2}$$

$$\Phi_i = \Gamma_i - \Psi \tag{3}$$

d) Calculate the covariance of matrix C by finding the eigenvector e_i and eigenvalue λ_i:

$$C = \frac{1}{M}\sum_{n=1}^{M}\varphi_n\varphi_n^T = AA^T \tag{4}$$

$$Ce_i = \lambda_i e_i \tag{5}$$

e) Obtain the eigenvector e_i and eigenvalue λ_i by finding the eigenvector and eigenvalue of matrix $C_1 = A^T A$ (dimension M × M). If v_i and μ_i is the eigenvector and eigenvalue of matrix $A^T A$, then:

$$A^T Av_i = \mu_i v_i \tag{6}$$

Multiply both sides of equation (5) with A from the left, obtaining:

$$AA^T Av_i = A\mu_i v_i$$

$$AA^T (Av_i) = \mu_i (Av_i)$$

$$C(Av_i) = \mu_i (Av_i) \tag{7}$$

f) Sequence the eigenvector in columns based on the eigenvalue, from largest to smallest.

g) By selecting the eigenvector with the largest eigenvalue, the principal component is obtained from the early matrix and can form the feature vector. By forming a new matrix E, then every e_i vector is the column vector. The dimension of this matrix is N × D, with D being the desired eigenvector number. This is used for the data projection of matrix A and the calculation of vector y_i matrix $Y = (y_1, \ldots, y_m)$

$$Y = E^T A \tag{8}$$

Every original image can be reconstructed by adding the average of image Ψ by summing the weights of all vectors of e_i.

1) Change the training image being sought to find vector P, diminished by the average Ψ and projected by the matrix of eigenvectors (eigenfaces):

$$\omega = E^T (P - \Psi) \tag{9}$$

The extracted features were reduced from 16384 to 256 by the DWT procedure. However, 256 was still too large for calculation. Thus, PCA was used to further reduce the dimensions of features. The curve of the cumulative sum of variance versus the number of principal components was shown in Fig. 7.

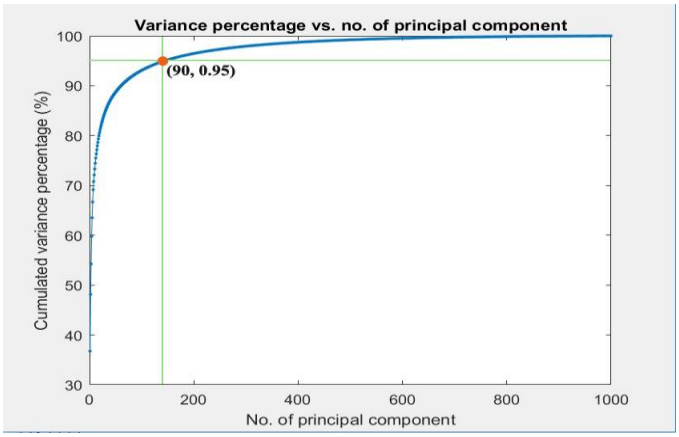

Fig. 7. The curve of variances against number of principle components (here we found that 90 features can achieve 95.03% variances).

The variances versus the number of principal components showed that only 90 principal components (bold font in the figure), which were only 90/256 = 35.15% of the original features, could preserve 95.03% of total variance.

Feature (principal component) with the highest percentage (variance) selected to be used in the process of correspondence among feature of face images[29].

5) Feature Representation

After the face image was projected into the face space, the next task was to determine which face image would be most similar to the image in the database [30]. In this research, the eigenvector with the largest eigenvalue was projected into the PCA space. This projection was stored as a k-d tree structure, indexed, and stored in a database [31]. This projection was later used in the testing stage and compared with the unknown image projection.

A k-d tree is a structure of partition space data used to set points in k-dimensional space based on the key values in the nodes [32], [21]. Components of K-d tree node in this research consisted of vector key or k-dimensional points, descriptor features, or pointers known as Left LINK and Right LINK (left subtree and right subtree). The sizes of the images used in this study after preprocessing were 64x64 pixels, 32x32 pixels, and 16x16 pixels. Each node on the K-d tree consists of key records. These nodes can be seen as points in a dimensional space. In addition, the nodes in K-d tree can also represent sub-regions of the entire space.

A feature which is the eigenvector with the value of variance percentage of PCA dimension reduction process as much as 90 principal components will be indexed. These eigenvectors were then deposited into k-d tree, with K in the k-d tree being the dimension of the template. The number of nodes in k-d tree is the same as the number of templates in the input files inserted into the tree [20].

Before constructing the tree, the data point xi must be played through the mapping of U^T to align the main axis with the coordinate axis. Given a sample there was $\{x_i | \, i = 1, \dots, N\}$ which was the set of R^d. Then eigenvector $A = \sum_{i=1}^{N} x_i {x_i}^T$. *Eigenvector* A is the principal axis of the data set, and eigenvalue is called the principal moment. If $A = U \Lambda U^T$ is the eigenvalue decomposition of A and column U is the eigenvector (orthogonal), then $x_i \rightarrow U^T x_i$ mapping the dots to the principal set of an axis that is aligned with the axis coordinates. If $U_{1:k}$ are the matrix consisting of the dominant k eigenvector, then the projection of ${U_{1:k}}^T$ is the set of points arranged into spaces stretched by k principal axis of the data. In general, if the data is aligned through the rotation of dimension U^T, the data is divided into the k-d tree and will be selected between k principal axis from the data. k is the depth of the tree. In building k-d tree, all data stored in the k-d tree must have the same dimension as the existing dimension in face space. The data consists of the dominant eigenvector and has been projected into the space stretched by the principal axis of the data k. In building or reconstructing k-d tree, all data stored in the k-d tree must have the same dimension as the existing dimension in face space. The k-d tree construction is performed using a recursive method with parameters at each iteration of arrays from points and depth. Depth value can be used to determine axis value. The initial value of arrays is all points and the depth value is 0. The Algorithm to construct k-d trees is shown in Algorithm 1.

Algorithm 1 The Algorithm to construct k-d trees

The Algorithm to construct k-d trees

Input: Set / Set of vectors $\{x_i\} \in R^d$
Output: k-d tree

1. Begin
2. Calculate the axis value with the formula, axis = depth mod 2
3. Sort array data based on axis value, if axis is smaller afterward sort done to the left, if the axis is larger, then sorting done to the right
4. Calculate the median coordinates, by:
   (1) index_median = number of coordinates div 2 and
   (2) coordinates_median = coordinates [index_median]
5. Determine the node: node = coordinates [index_median]
6. Specify left node and right node using new iteration. Next iteration use the following parameters:
   (1) array coordinates = sub array coordinates = coordinates [0]
   (2) array coordinates = sub array coordinates = coordinates [index_median + 1] [coordinate number]
7. End

B. Facial Recognition Stage

1) Multiple Face Images from Video

Images of multiple student faces in the classroom were captured from videos using an IP camera (Zavio F320). Images of students' faces had previously been registered with students' names and identification codes in the database. Five classrooms were used in this recognition stage. Each classroom consisted of forty students; as such, the total dataset used for the test consisted of 200 face images. Face images captured from video streams using an IP Camera were extracted frame-by-frame until the 60th frame [33]. Five video samples in AVI (Audio Video Interleave) format with a resolution of 1920x1080 was used. These videos had a frame rate of 30 fps [34] and duration of two minutes. The computer system in this study has a detection and facial recognition software installed on a personal computer (PC) or notebook and the lecturer should run it at the beginning of each class in the course group that has been determined on schedule. The layout system is shown in Fig. 8.

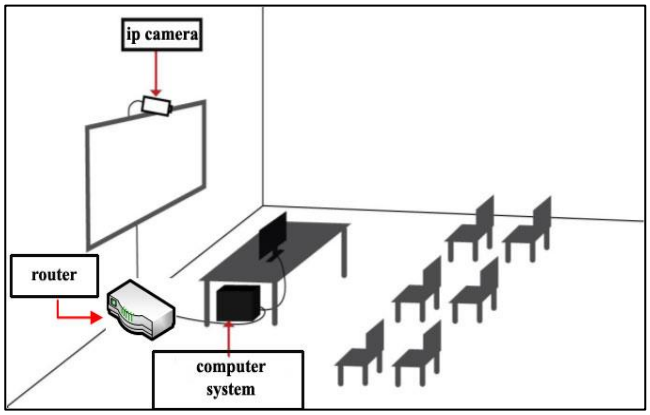

Fig. 8. The Layout System.

2) Preprocessing

The following steps were involved in the preprocessing stage:

a) Faces were cropped using multiple face detection on each video frame. Face detection was implemented based on Haar-like features using the Viola–Jones method [24], which quickly detects multiple faces in the classroom (Fig. 9).

b) Faces were changed from RGB to grayscale mode.

c) Faces were resized to 128x128 pixels

d) Images of faces were normalized using a histogram equalization process

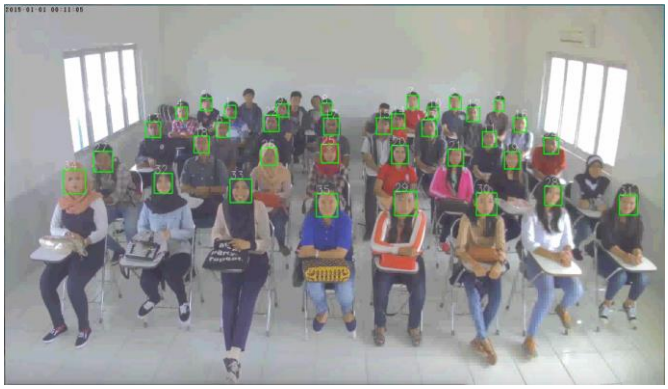

Fig. 9. Sample of Multiple Face Detection Results in the Classroom (detected faces are marked by green rectangles).

The preprocessing stage of the multiple-face images in this research is shown in Fig. 10.

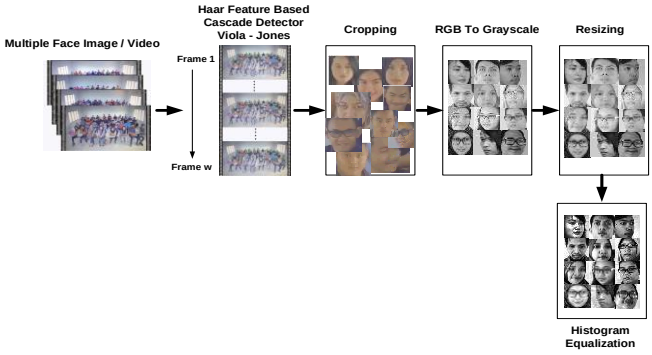

Fig. 10. Multiple face image preprocessing.

3) 2D Discrete Wavelet Transform

The 2D Discrete Wavelet Transform (2D DWT) process used the same process as the face database. Images resulting from preprocessing were decomposed at three levels. At each level of decomposition, DWT was first performed in a vertical direction, followed by a horizontal direction. After performing the first level of decomposition, four sub-bands were obtained: LL1, LH1, HL1, and HH1. For each level of decomposition, the sub-band LL from the previous level was used as input. Sub-band LL was only used for DWT calculation in the next scale. To calculate the wavelet features in the first stage, wavelet coefficients were calculated on sub-band LL1 using a Haar wavelet function. For the second level of decomposition, DWT was applied to band LL1 by decomposing it into four sub-bands: LL2, LH2, HL2, and HH2. For the third level of decomposition, DWT was applied to band LL2 by decomposing it into four sub-bands: LL3, LH3, HL3, and HH3. LL3 contained low-frequency bands, while LH1, HL1, and HH1 contained high-frequency bands. The face image size was 128x128 pixels obtained from the pre-processing results was then decomposed to 64x64 pixels on the wavelet level 1, then the 64x64 pixel face image was decomposed to 32x32 on the wavelet level 2, and the last was the 32x32 pixel face image decomposed using wavelets level 3 to 16x16 pixels. Experiments were performed on all three levels of wavelet decomposition to find the right level of accuracy of recognition of many faces. Third face image of wavelet decomposition process will be processed by PCA (*principal component analysis*).

4) Principal Component Analysis

This stage was the same as the face database stage, with images resulting from the three levels of decomposition applied in the PCA process. This is very significant for feature extraction due to the reduction in dimensions, which leads to the reduction of computation complexity. For the recognition process, a test image (the image presented to the system for the recognition process) has the same dimension as the training image presented to the system. The test image is then extracted by multiplying by the eigenvector of the training image, and producing a feature vector containing the main component having the same dimension as the vector of the training image feature. Once the feature vector was obtained from the test image, the next process was to compare the feature vector of the test image with the feature vector of the training image. The results of this PCA process will then be used in the classification stage.

5) K-Nearest Neighbor Search using Priority K-d Tree

In this section, the calculation of similarity was done by calculating the level of similarity (distance) between test data and training data. The calculation of similarity level in this research was done by using Euclidean Distance shown in equation [35].

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (10)$$

From (10) it can be explained that d (x, y) is the level of similarity between the test data (x) and the training data (y), x_i is the i feature value of the test data and y_i is the i feature value of the training data. n is the number of x and y features.

In Fig. 11(a) shows an example of a nearest neighbor priority queue that has a maximum size of five and has five elements, A-E. Suppose the nearest closest neighbor that is put into the priority queue is element F with a priority or a distance of 0.50. Since the priority queue has a maximum size of five, Element F is entered into the priority queue, while element E with the longest distance is removed from the priority queue.

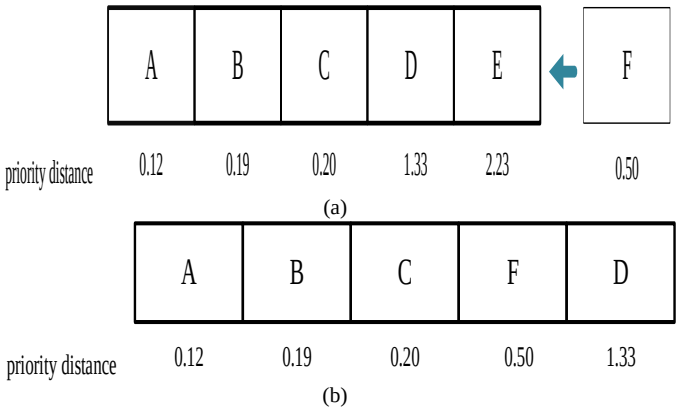

Fig. 11. An example of a nearest neighbor priority queue.

We use 10 test features in two dimensions as an example to illustrate the process of building k-d tree. Fig. 12(a) shows how the feature space is divided into hyper-rectangles iteratively, where the red points are test features and the blue one is the query feature. The tree structure in two dimensions is illustrated in Fig. 12(b).

Pseudocode for closest neighbor search using the priority k-d queue is described in Algorithm 2.

Algorithm 2 The k-d tree nearest neighbor search

The k-d tree nearest neighbor search Algorithm

Input : node, currNode, lNode, lDist, *currDist.* eucldieanDistance, value, q, min_dist, k, neighborList
Output : nearest neighbor data points // distance //nearest distance between the query point and the data points

1. Begin
2. lNode=NULL;/* last node null value */
3. lDist=Max_val; last distance with maximum value
4. If neighborlist.size > 0 then /* Calculate the Euclidean
   The distance between the last node in *neighborList* and the query point */
5. lNode = neighborList.last ();
6. lDist = euclideanDistance (lNode.value, value);
7. End
8. lDist = euclideanDistance (lNode.value, value); /* Calculate the Euclidean distance between the current node and the query point. */
9. currDist = euclideanDistance(curr.value, value); /* Add the current node to the neighbor list if necessary. */
10. if *currDist* < *lDist* then
11. if *neighborList.size* == *k AND lNode* ≠ *NULL* then
12. neighborList.remove(lNode);
13. end
14. neighborList.add(curr) ; /* Add the current node to *neighborList*. The neighbor list is automatically sorted when the current node is added to the list. */
15. else if *currDist* == *lDist* then
16. neighborList.add(curr) ; /* Add the current node to *neighborList*. Note: The neighbor list can have more than k neighbors if the last nodes have equal distances in our implementation. */
17. else if *neighborList.size* < *k* then
18. neighborList.add(curr); /* Add the current node to *neighborList*. */
19. end
20. lNode = neighborList.last();
21. lDist = euclideanDistance(lNode.value, value) ; /* Calculate the Euclidean distance between the last node in *neighborList* and the query point. *lDist* is equivalent as *r* in Equation (11). */
22. axis = curr.depth % k ; /* Get the current axis. */
23. left = curr.left ; /* Get the current node's left child. */
24. right = curr.right ; /* Get the current node's right child. */
25. if *left* ≠ *NULL AND !checked.contains(left)* then search the left child branch
26. checked.add(left) ; /* Add the left child to the examined list. */
    /* Calculate the difference between the splitting coordinate of the query point and the current node. *difference* is equivalent as $|q_i - x_i|$ in Equation (11). */
27. if *axis* == *X_AXIS* then
28. difference = abs(value.x - curr.value.x) ; /* abs is *absolute* operator. */
29. else if *axis* == *Y_AXIS* then difference = abs(value.y - curr.value.y);
30. else if *axis* == *T_AXIS* then
31. difference = abs(value.t - curr.value.t);
32. end
33. intersection = (difference < lDist); ; /* Determine if the splitting plane intersects the hyper-sphere using Equation (11). */
34. if *intersection* then continue down the left branch
35. searchNode(value, left, k, neighborList, checked);
36. end
37. end
38. if *right* ≠ *NULL AND !checked.contains(right)* then search the right child branch checked.add(right) ; /* Add the right child to the checked list. */
39. if *axis* == *X_AXIS* then
40. difference = abs(value.x - curr.value.x);
41. else if *axis* == *Y_AXIS* then
42. difference = abs(value.y - curr.value.y);
43. else if *axis* == *T_AXIS* then
44. difference = abs(value.t - curr.value.t);
45. end
46. intersection = (difference < lDist); ; /* Determine if the splitting plane intersects the hyper-sphere using Equation (11). */
47. if *intersection* then continue down the right branch
48. searchNode(value, right, k, neighborList, checked);
49. end
50. end

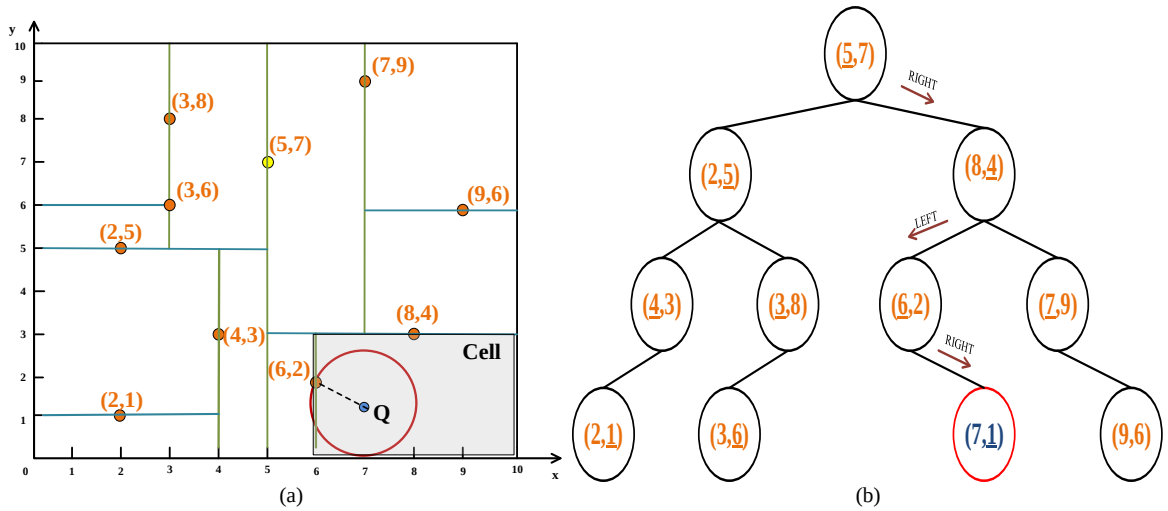

Fig. 12. (a) The feature space is divided into hyper-rectangles iteratively; (b) the tree structure in two dimensions.

In Algorithm 2, the first step is to input the k-d tree that is already constructed with the target point is $x_i = \left(x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(k)}\right)^T$, i = 1, 2, ..., n., it is explained that the algorithm gives a query point first To find the nearest neighbor in k-d tree, then the k-NN algorithm looks for similarity of features in k-d tree by traversal determines the proper branches to explore. Each branch in k-d tree represents the space partition. This is intended to explore partitions closer to the query point. Partitions closer to this query point contain features that are similar to the nearest neighbors. The nearest neighbor search algorithm works by starting from the root node and running down the k-d tree recursively to find the query point. Once the algorithm reaches the leaf node in the k-d tree, the node is stored as the closest neighbor for the moment. The algorithm performs recursion and checks the tree again. Checks are performed on each node to find a better node with the nearest neighbor. The algorithm performs recursion and checks the tree again. The leaf node contains the target node x in the k-d tree. To find the nearest neighbor in the k-d tree, the algorithm gives Q query point first as shown in Fig. 12(a), then the k-NN algorithm search similarity features in the k-d tree by traversal by specifying the right branch to explore. When the priority search descends and reaches a sub-tree, sibling of the subtree is added to the sorted list. The subtree in the sorted list is then saved. The stored subtree has a distance between the query feature and the hyperrectangle corresponding to each subtree. If the coordinates of the current point x coordinates are less than the coordinates of the cut-point, the search is performed to the left sub-node. If the current point x point coordinates are greater than the cut-off point, the search is performed to the right sub-node. Until the child's node is a leaf node. If the instance stored on the node is closer than the current closest point to the query point, the instance point is considered the current closest point. The algorithm forms a hyper-sphere centered at the query point with the radius of the circle being the distance (in this study is the Euclidean distance) which is calculated between the nearest best query point and current neighbor. The current nearest point should be in the region corresponding to the sub-node. Hyper-sphere candidates are formed by centering on the query point q (q_0, q_1, q_2, ..., q_{k-1}) and through the point of the current node. The nearest neighbor node point to the request point should be within the hyper environment. The equation for determining the hyper-sphere candidate in the hyper environment is indicated by

$$|q_i - x_i| < r \tag{11}$$

q is the query point, x is the node, r is the radius and i is the i-th dimension.

IV. Result and Analysis

In this study, static and video images from the classroom were taken for the training process using a digital camera and an ip camera for testing process. A total of 9,126 facial data were collected from 1,014 people and stored in the database. Every individual had nine face poses with different angles. Tests were conducted to evaluate performance of proposed algorithm. Tests carried out with two face dataset which are Honda/UCSD [36] (Fig. 13) and AtmafaceDB dataset (Fig. 14). To test the accuracy of images, a total of 1350 face images were used. An Intel Core i5-7200U CPU @2.50 GHz was used to process this experiment. The experiment was conducted to compare the level of recognition using different levels of discrete wavelet decomposition. First, preprocessing was applied by cropping faces for face detection and applying localization processes in a rectangular area using the Viola-Jones method. Face images that had been cropped or cut were converted from color images into grayscale images. Facial images were then resized to 128x128 pixels. The images brightness was then normalized using histogram equalization.

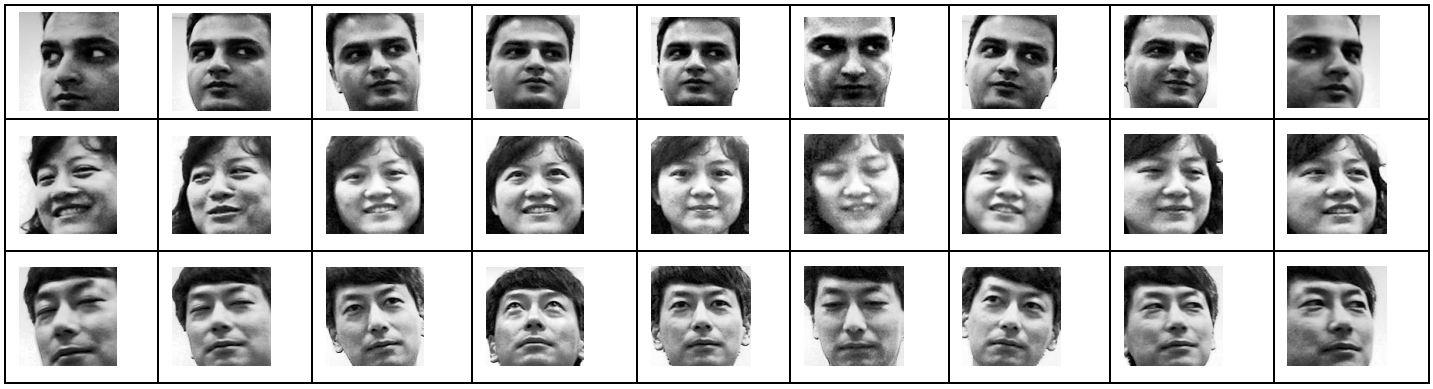

Fig. 13. Several samples of Honda/UCSD Dataset.

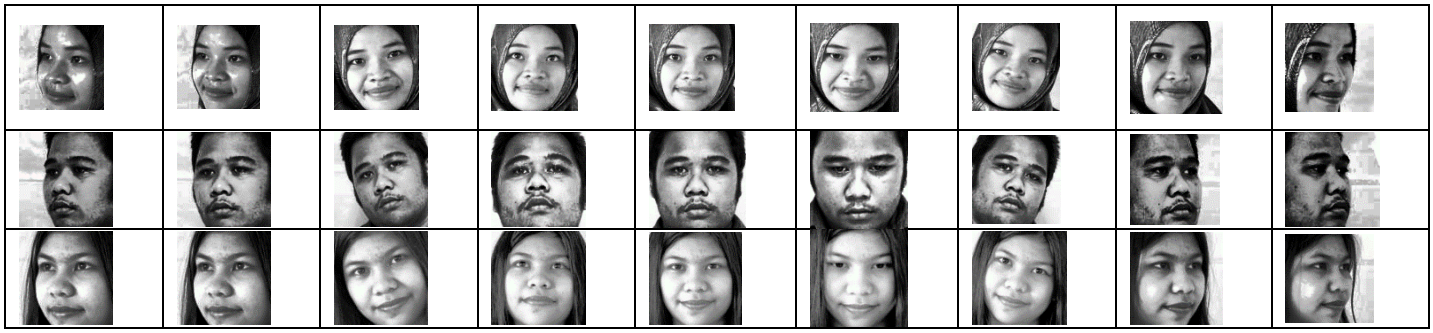

Fig. 14. Several samples of AtmafaceDB Dataset (Our Dataset).

Second, the 2D DWT-PCA method was applied. It was used for feature extraction using the face images resulting from preprocessing. After that, the result of student facial feature extraction was applied by k-d tree method. K-d tree was used to accelerate the process of facial classification using k-NN. The face recognition algorithm applied in this research is shown in Algorithm 3.

Algorithm 3 Multiple Face Recognition Algorithm

The Multiple Face Recognition Algorithm

Input: Face Images
Output: Multiple Faces Recognized
1. Face Detection
2. Face Image Preprocessing
3. n level of 2D DWT application on face images
4. Sub-band LL of 2D DWT application on PCA, followed by generation of feature vectors
5. Storing of feature vectors in the database
6. Indexing feature vectors in the database using k-d tree
7. Finding the distance between the testing sample and all training samples using a Euclidean distance algorithm
8. Sequencing of all training samples using nearest neighbor approach based on the minimal distance taken

In this study, the optimal choice of K was determined by 10-fold cross-validation [37]. Optimal K based on research from [35], [38] was used in this study. Each experiment was done 10 times, then the result was calculated from the average of 10 times of the experiment. In the k-d tree data structure was done several times of testing with different k values, namely k = 1, k = 3, k = 5 and k = 7. In this research, the time taken to calculate the result of classification with k-NN.

Based on Table 1, k-NN performs the best accuracy on k equal to 3 on the 3rd level wavelet for the Honda/UCSD dataset, while the best accuracy on k equal to 5 on the 3rd level wavelet for the AtmafaceDB dataset. This shows that k-NN gives 82.00% on 600 Honda/UCSD faces and 95.56% on 1350 AtmafaceDB faces. K equals 5 on our dataset would be used in the next test because the best result.

TABLE I. K-NN Face Accuracy

K Value	Average Accuracy (%)	
	Honda dataset	***AtmafaceDB dataset (our dataset)***
K=1	79.75	93.40
K=3	**82.00**	95.33
K=5	81.43	**95.56**
K=7	80.86	95.33

Table 2 shows the accuracy and timing of recognition on three levels of 2D DWT-PCA using k-d tree on AtmafaceDB dataset.

TABLE II. Accuracy of Face Recognition Three Levels of 2D DWT-PCA using K-D Tree (K=5)

Level of Decomposition	**Recognition Accuracy**	**Recognition Time (ms)**
1 (64x64)	89.49%	98
2 (32x32)	90.00%	69
3 (16x16)	95.56%	40

Comparison of computing time for facial recognition at each level of 2D DWT-PCA using k-d tree is shown in Fig. 15.

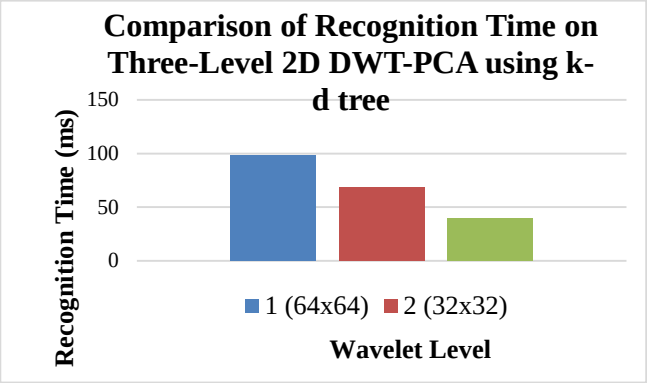

Fig. 15. The comparison of recognition time on three-levels of 2D DWT-PCA using k-d tree.

V. Conclusion

This paper proposes an attendance management system based on multiple face recognition combining the 2D DWT-Principal Component Analysis method with the k-Nearest Neighbor (k-NN) classification. Indexing using the k-d tree technique is conducted to speed up the classification process using a k-nearest neighbor (k-NN) approach. 2D DWT was decomposed into three levels. From the experimental results, k-Nearest Neighbor face recognition delivered best accuracy 95.56% on k=5. At each wavelet level, computational time for recognizing faces was compared. The three-level 2D DWT-PCA facial recognition method proposed shows good results. Research results show that this method reaches 95.56% accuracy, with a computing time of 40 milliseconds required for facial recognition. From the test results, it can be concluded that the use of data structure of k-d tree can reduce the time required when performing classification by using the method of k-nearest neighbors. The time required for classification using k-d tree also changes due to the facial image size of different wavelet decompositions. The future work of this research is to utilize GPU (Graphical Processing Unit) on detection of tracking model using KLT method and utilizing MBR (Minimum Bounding Rectangle) on k-d tree method with the aim of increasing accuracy of multiple-face recognition in classroom.

References

[1] J. Shermina and V. Vasudevan, “An efficient face recognition system based on the hybridization of invariant pose and illumination process,” Eur. J. Sci. Res., vol. 9, no. 4, pp. 228–237, 2011.

[2] L. Li, T. Losser, C. Yorke, and R. Piltner, “Fast inverse distance weighting-based spatiotemporal interpolation: A web-based application of interpolating daily fine particulate matter PM<inf>2:5 </inf>in the contiguous U.S. using parallel programming and k-d tree,” Int. J. Environ. Res. Public Health, vol. 11, no. 9, pp. 9101–9141, 2014.

[3] M. Kirby and L. Sirovich, “Application of the Karhunen-Loeve procedure for the characterization of human faces,” Pattern Anal. Mach. Intell. …, vol. 12, no. 4, 1990.

[4] M. Turk and A. Pentland, “Face recognition using eigenfaces,” Proceedings. 1991 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 586–591, 1991.

[5] L.-H. Chan, S.-H. Salleh, and C.-M. Ting, “Face Biometrics Based on Principal Component Analysis and Linear Discriminant Analysis,” J. Comput. Sci., vol. 6, no. 7, pp. 693–699, 2010.

[6] N. G. Chitaliya and a. I. Trivedi, “Feature Extraction Using Wavelet-PCA and Neural Network for Application of Object Classification & Face Recognition,” 2010 Second Int. Conf. Comput. Eng. Appl., vol. 1, pp. 510–514, 2010.

[7] M. K. Rao, K. V. Swamy, and K. A. Sheela, “Face recognition using DWT and eigenvectors,” 2012 1st Int. Conf. Emerg. Technol. Trends Electron. Commun. Netw., pp. 1–4, Dec. 2012.

[8] Naveed Khan Balcoh, M. Haroon Yousaf, Waqar Ahmad, and M. I. Baig, “Algorithm for Efficient Attendance Management: Face Recognition based approach,” IJCSI Int. J. Comput. Sci. Issues, vol. 9, no. 4, pp. 146–150, 2012.

[9] Winarno Edy, A. Harjoko, A. M. Arymurthy, and E. Winarko, “Improved Real-Time Face Recognition Based on Three Level Wavelet Decomposition-Principal Component Analysis and Mahalanobis Distance,” J. Comput. Sci., vol. 10, no. 5, pp. 844–851, 2014.

[10] Y. Mao, H. Li, and Z. Yin, “Who missed the class? - Unifying multi-face detection, tracking and recognition in videos,” in 2014 IEEE International Conference on Multimedia and Expo (ICME), 2014, pp. 1–6.

[11] S. Lukas, A. R. Mitra, R. I. Desanti, and D. Krisnadi, “Student attendance system in classroom using face recognition technique,” 2016 Int. Conf. Inf. Commun. Technol. Converg., pp. 1032–1035, 2016.

[12] S. Sayeed, J. Hossen, V. Jayakumar, I. Yusof, and A. Samraj, “Real-Time Face Recognition for Attendance,” J. Theor. Appl. Inf. Technol., vol. 95, no. 1, 2017.

[13] H. Samet, “K-nearest neighbor finding using MaxNearestDist,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 30, no. 2, pp. 243–252, 2008.

[14] J. Toyama, M. Kudo, and H. Imai, “Probably correct k-nearest neighbor search in high dimensions,” Pattern Recognit., vol. 43, no. 4, pp. 1361–1372, 2010.

[15] Y. Xu, Q. Zhu, Y. Chen, and J. S. Pan, “An improvement to the nearest neighbor classifier and face recognition experiments,” Int. J. Innov. Comput. Inf. Control, vol. 9, no. 2, pp. 543–554, 2013.

[16] Y. Wang, T. Zhou, and B. Hu, “A Step-wise Refinement Algorithm for Face,” J. Inf. Hiding Multimed. Signal Process., vol. 6, no. 3, pp. 554–567, 2015.

[17] D. Mateos-García, J. García-Gutiérrez, and J. C. Riquelme-Santos, “On the Evolutionary Weighting of Neighbours and Features in the k-Nearest Neighbour Rule,” Neurocomputing, 2017.

[18] E. Chávez, M. Graff, G. Navarro, and E. S. Téllez, “Near neighbor searching with K nearest references,” Inf. Syst., vol. 51, pp. 43–61, 2015.

[19] L. Hu and S. Nooshabadi, “Massive parallelization of approximate nearest neighbor search on KD-tree for high-dimensional image descriptor matching,” J. Vis. Commun. Image Represent., vol. 44, pp. 106–115, 2017.

[20] U. Jayaraman, S. Prakash, and P. Gupta, “Indexing multimodal biometric databases using Kd-tree with feature level fusion,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008, vol. 5352 LNCS, no. 1, pp. 221–234.

[21] M. Otair, “A Proximate K-Nearest Neighbour Based Spatial Clustering Using Kd-Tree,” Int. J. Database Manag. Syst., vol. 5, no. 1, pp. 97–108, 2013.

[22] R. A. Brown, “Building a Balanced k -d Tree in O (kn log n) Time,” J. Comput. Graph. Tech., vol. 4, no. 1, pp. 50–68, 2015.

[23] G. Heineman, G. Pollice, and S. Selkow, Algorithms in a Nutshell, vol. 26, no. 3. 2008.

[24] P. Viola and M. Jones, “Robust real-time face detection,” Int. J. Comput. Vis., vol. 20, p. 7695, 2004.

[25] P. Nicholl, A. Ahmad, and A. Amira, “Optimal discrete wavelet transform (DWT) features for face recognition,” IEEE Asia-Pacific Conf. Circuits Syst. Proceedings, APCCAS, no. December, pp. 132–135, 2010.

[26] Z. Huang, W. Li, J. Shang, J. Wang, and T. Zhang, “Non-uniform patch based face recognition via 2D-DWT ☆,” IMAVIS, vol. 37, pp. 12–19, 2015.

[27] M. Turk and A. Pentland, “Eigenfaces for recognition,” J. Cogn. Neurosci., vol. 3, no. 1, pp. 71–86, Jan. 1991.

[28] M. Slavković and D. Jevtić, "Face Recognition Using Eigenface Approach*," Serbian J. Electr. Eng., vol. 9, no. 1, pp. 121–130, 2012.

[29] G. V Sagarl, S. Y. Barker, K. B. Raja, K. S. Babu, and K. R. Venugopal, "Convolution based Face Recognition using DWT and Feature Vector Compression," in Third International Conference on Image Information Processing, 2015, pp. 444–449.

[30] H. Samet, "Foundations of Nearest Neighbor Queries in Euclidean Space ∗," Encyclopedia of GIS. Springer International Publishing, pp. 1–10, 2016.

[31] U. Jayaraman, S. Prakash, and P. Gupta, "An efficient technique for indexing multimodal biometric databases Umarani Jayaraman *, Surya Prakash and Phalguni Gupta," Int. J. Biometrics, vol. 1, no. 4, pp. 418–441, 2009.

[32] Hanan Samet, The Design and Analysis of Spatial Data Structures. Addison - Wesley Publishing Company, Inc., 1990.

[33] C. P. Shirley, A. Lenin Fred, and N. R. RamMohan, "Video key frame extraction through wavelet information scheme," ARPN J. Eng. Appl. Sci., vol. 11, no. 7, pp. 4414–4423, 2016.

[34] S. Chen, S. Mau, M. T. Harandi, C. Sanderson, A. Bigdeli, and B. C. Lovell, "Face recognition from still images to video sequences: A local-feature-based framework," Eurasip J. Image Video Process., vol. 2011, 2011.

[35] D. A. Adeniyi, Z. Wei, and Y. Yongquan, "Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method," Appl. Comput. Informatics, vol. 12, no. 1, pp. 90–108, 2016.

[36] K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman, "Video-based face recognition using probabilistic appearance manifolds," Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. I–313–I–320, 2003.

[37] J. Huang et al., "Cross-validation based K nearest neighbor imputation for software quality datasets: An empirical study," J. Syst. Softw., vol. 132, pp. 226–252, 2017.

[38] Y. F. Li, M. Xie, and T. N. Goh, "A study of project selection and feature weighting for analogy based software cost estimation," J. Syst. Softw., vol. 82, no. 2, pp. 241–252, 2009.