Face Recognition Application Using Adaptive Boosting and Gray Level Co-Occurrence Matrix

Chairisni Lubis[1], Novario Jaya Perdana[2], Hengki Pranoto[3]
{ chairisnil@fti.untar.ac.id[1], novariojp@fti.untar.ac.id[2], hengki.535150015@stu.untar.ac.id }

Universitas Tarumanagara, Jakarta, Indonesia

Abstract. Face recognition has been an interesting field to explore. Although some previous researches have successfully proved to detect faces, there are still some difficulties to automatically recognize whose faces in one image. Human face is a dynamic object which a high degree of variability exists in its appearance. Therefore, every face has their own uniqueness. These unique features could be used for recognizing one from the other. This research is intended to offer new approach on the field. It is done by combining three methods for both detecting and recognizing faces. It begins with applying Adaptive Boosting (AdaBoost) to detect faces on a picture, then employing Gray Level Co-Occurrence Matrix (GLCM) to extract their features. Finally, K-Nearest Neighbor is used to recognize the owner of the face. This combination of methods is proven to get significant result of face detection but only fair result of face recognition. In addition to the research, an application has been developed. After doing research and development of this application, it can be concluded that the combination of the methods could become another approach for face recognition.

Keywords: Adaptive Boosting, GLCM, Human Face Recognition, K-Nearest Neighbor

1 Introduction

Nowadays, the development of information technology has been rising. The availability of technology capable of making work easier. One of the implementations of technology is human face recognition. However, there are some difficulties on implementation because computer doesn't have an intelligence and ability like humans to differentiate human faces. Every human has their own uniqueness, between one person and another there are differences even when they are twins. The point that differentiate one from another like face, ear, hair, etc. The key to differentiate one from another is unique features on the face. Before recognizing human faces, first we need to detect human faces using Adaptive Boosting (AdaBoost). AdaBoost was chosen because it is quite decent on decision making. This method combines several weak classifiers into a strong classifier. This method was implemented by Viola-Jones giving 93,7% detection rate [1]. Region of image that detected as face will be cropped to get only its face area. The cropping result will be normalized by its sizes. Then, we can do feature extraction to get the unique features from the face itself. The method we use to extracting face features is Gray Level Co-Occurrence Matrix (GLCM). Image operations on GLCM has several angles used as an approach such as: 0, 45, 90, 135 degrees. Features on GLCM is being used to recognize human faces. We will compare the values of each feature from train

ICONISTECH-1 2019, July 11-12, Bandung, Indonesia

DOI 10.4108/eai.11-7-2019.2297826

images and test images to recognize the owner of the face using K-Nearest Neighbor (KNN). This application goal is to recognize human faces.

2 Method

The development of this application including preprocessing images, extrating haar feature, computing adaptive boosting, extracting texture features using gray level co-occurrence matrix and predicting the image class using KNN and euclidean distance.

2.1 Grayscalling

To make the image process a lot easier, first we need to convert RGB images into grayscale images. To convert images from RGB to grayscale we can take an average from red, green, and blue value from the image [2].

$$s = \frac{r + g + b}{3} \tag{1}$$

Where s is grayscale value, r is red pixel value, g is green pixel value, and b is blue pixel value.

2.2 Integral Image

Integral image was used to optimize image operation to become more efficient when finding Haar Feature. Haar Feature itself can produce many possibilities up to 160.000 features, so it makes finding Haar Features rather time consuming. From that facts, integral image was then used to speed up the computational process [1].

To get an integral image value on a pixel, an addition operation was needed for each pixel that are on the left and top side from the reference pixel. The illustration integral image computation can be seen on Figure. 1.

$$ii(x, y) = i(x, y) + ii(x - 1, y) + ii(x, y - 1) - ii(x - 1, y - 1) \tag{2}$$

Where i(x,y) is the value of reference pixel, ii(x,y) is the value of integral image on the reference pixel, ii(x-1,y) is the value of integral image on the left side of reference pixel, ii(x,y-1) is the value of integral image on the top side of reference pixel, and ii(x-1,y-1) is the value of integral image on both left and top side of reference pixel. With integral image we can find the area / region of the image rapidly. Illustration to find integral image on area of the image can be seen on Figure. 2.

$$ii(x', y') = ii(A) + ii(D) - ii(B) - ii(C) \tag{3}$$

Where ii(x',y') is the value of integral image on the reference area pixel, ii(A) is the value of the integral image on top-left bound, ii(B) is the value of integral image on top-right bound,

ii(C) is the value of integral image on bottom-left bound, ii(D) is the value of integral image on bottom-right bound.

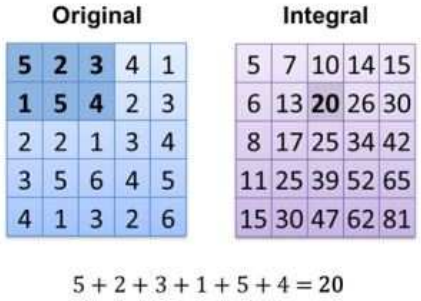


Figure. 1. Integral Image Illustration

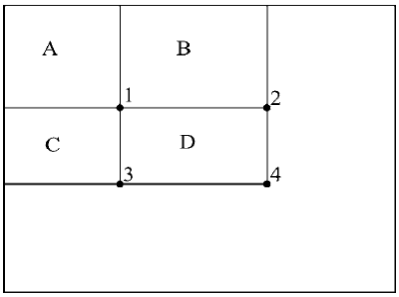


Figure. 2. Illustration to Find Area on Integral Image

2.3 Haar Feature

Haar feature was used as a weak classifier for Adaptive Boosting method and used as a learning of face image and non-face image. To find Haar feature we compute the differences between sum of dark pixel and sum of bright pixel. We use 5 Haar feature to construct this application that can be seen on Figure. 3. To reach optimal results we repeat the same procedure on face area with variated sizes. Every computation of Haar feature will be used as a threshold value. On Viola-Jones construction that implemented Haar feature there are approximately 160000 features on an image with 24 x 24 window size [1].

$$Haar = \left| \sum dark\ area\ - \sum brightarea \right| \quad (3)$$

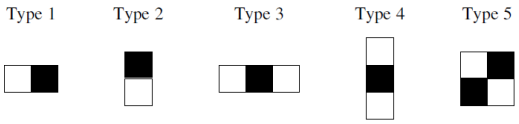


Figure. 3. Haar Feature

2.4 Adaptive Boosting

Adaptive Boosting (AdaBoost) was a learning method that construct weak classifier iteratively. On each iteration we call a simple learning algorithm that return a classifier with given weight. Final classification will be decided through voting from base classifier. If the error from base classifier was fewer, it will be given a bigger weight. Base classifier that being used was slightly better than a random guess [3]. AdaBoost combines many weak classifiers to create a strong classifier. The Algorithm were:

1. Initiate weight values with W(i) = 1/n; n = number of objects.
2. For each iteration 1…T
    a. Compute error values

$$e(i) = \sum_{i=1}^{n} W(i)\, I\{H(i) \neq Y(i)\} \begin{cases} 1, & if\ H(i) \neq Y(i) \\ 0, & otherwise \end{cases} \quad (3)$$

    b. Compute coefficient weights

$$a(i) = \frac{1}{2}\ln\left(\frac{1-e(i)}{e(i)}\right) \quad (4)$$

    c. For weight updates, iterate each i values, if H(i) != Y(i); H(i) = haar feature value, if higher than threshold the value is 1, otherwise the value is -1, Y(i) = if the object is face the value is 1, otherwise the value is -1, then :

$$w\_new(i) = \frac{w\_old(i)}{2e(i)} \quad (5)$$

otherwise

$$w\_new(i) = \frac{w\_old(i)}{2(1-e(i))} \quad (6)$$

3. Finally, the AdaBoost prediction value was computed.

$$f(i) = \sum_{i=1}^{T} a(i)H(i) \quad (7)$$

2.5 Cascade Classification

Classifier was trained using positive and negative samples with the same size. After the classifier trained, next we find the region of interest (ROI) in input images. The sub window that predicted as object that we wanted to find, in the case the faces will produce the output value of 1, otherwise the output value is 0. To find an object in on a whole input image, we can scan from image pixel starting point until end point and find every location using

classifier. Classifier designed to finding objects on various sizes. The word cascade on classifier refer to classifier that contain some stage on ROI until candidates pass through all of classifier stages or candidates rejected [4].

2.6 Gray Level Co-occurrence Matrix

Gray Level Co-Occurrence Matrix (GLCM) method was used to extract texture features on an image. GLCM value was obtained from grayscale image. GLCM computes how often a grayscale pixel with i value occur horizontally, vertically, or diagonally to a neighbor pixel with radius d and value j. The angle used on GLCM analysis was horizontal (0 degree), vertical (90 degree), and diagonal (45 and 135 degree).

GLCM value for 4 angles above with radius 1 stated on matrix P(d,θ) with d is radius, and θ is angle. On this application the radius that will be used is 1. Then each matrix was normalized by dividing each matrix cell with matrix size. For computational efficiency, μx, μy, stdx, stdy, Px, Py, Px+y(k), dan Px-y(k) were computed [5].

$$P_x(i) = \sum_{j=0}^{G-1} P(i,j) \tag{8}$$

$$P_y(j) = \sum_{i=0}^{G-1} P(i,j) \tag{9}$$

$$\mu_x(i) = \sum_{i=0}^{G-1} i \sum_{i=0}^{G-1} P(i) \tag{10}$$

$$\mu_y(j) = \sum_{i=0}^{G-1} \sum_{i=0}^{G-1} jP(i) \tag{11}$$

$$stdx = \sqrt{\sum_{i=0}^{G-1} (px(i) - \mu x)^2} \tag{12}$$

$$stdy = \sqrt{\sum_{j=0}^{G-1} (py(j) - \mu y)^2} \tag{13}$$

$$Px + y(k) = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j) \quad i + j = k \; where \; k = 0,1,\dots,2(g-1) \tag{14}$$

$$Px - y(k) = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j) \quad |i - j| = k \; where \; k = 0,1,\dots,g-1 \tag{15}$$

Where P(i,j) is GLCM matrix, Px(i) is x-vector on GLCM matrix, Py(j) is y-vector on GLCM matrix, μx(i) is x-mean on GLCM matrix, μy(j) is y-mean on GLCM matrix, stdx is standard deviation value on x-axis GLCM matrix, stdy is standard deviation value on y-axis

GLCM matrix. Px+y(k) is diagonal vector on GLCM matrix, Px-y(k) is diagonal vector on GLCM matrix. After that, we will compute each feature value.

1. *Homogeinity, Angular Second Moment* (ASM)

$$ASM = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(P(i,j))^2 \tag{16}$$

2. *Contrast*

$$Contrast = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(i-j)^2 P(i,j) \tag{17}$$

3. *Local Homogeinity, Inverse Difference Moment* (IDM)

$$IDM = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}\frac{1}{1+(i-j)^2}P(i,j) \tag{18}$$

4. *Entropy*

$$Entropy = -\sum_{i=0}^{G-1}\sum_{j=0}^{G-1}P(i,j)\, xlog\big(P(i,j)\big)\ where\ P(i,j) = 0 \tag{19}$$

5. *Correlation*

$$Correlation = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}\frac{(ixj)\ x\ P(i,j) - (\mu x + \mu y)}{stdx + stdy} \tag{20}$$

6. *Sum of Square, Variance*

$$Variance = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(i-\mu)^2 P(i,j) \tag{21}$$

7. *Sum Average*

$$Aver = \sum_{i=0}^{2G-2} iPx + y(i) \tag{22}$$

8. *Sum Entropy*

$$Sent = -\sum_{i=0}^{2G-2} Px + y(i)\ x \log\big(Px + y(i)\big)\ where\ Px + y(i)! = 0 \tag{23}$$

9. *Difference Entropy*

$$Dent = -\sum_{i=0}^{G-1} Px - y(i)\ x \log\big(Px - y(i)\big)\ where\ Px + y(i)! = 0 \tag{24}$$

10. *Inertia*

$$Inertia = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(i-j)^2 P(i,j) \tag{25}$$

11. *Cluster Shade*

$$Shade = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(i+j-\mu x-\mu y)^3 P(i,j) \quad (26)$$

12. *Cluster Prominence*

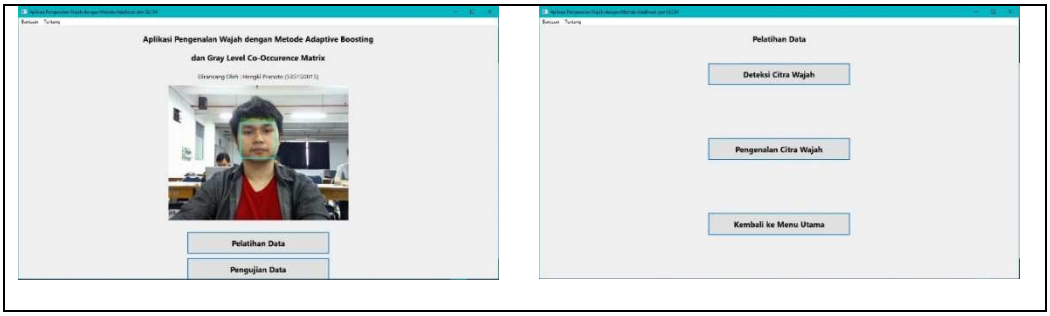
$$Prom = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(i+j-\mu x-\mu y)^4 P(i,j) \quad (27)$$

2.7 K-Nearest Neighbor

K-Nearest Neighbor (KNN) was an algorithm that used to divide data into some classes for prediction classification on new samples [6]. This algorithm decides classes from test data with finding k nearest point on train data. After that, we find which class is dominant from that point, then the dominant class will be classified as a class for new samples. The equation that used for computing new data point and database data point was Euclidean Distance. In this application, the point that will be compared is train image face features and test image face features.

3 User Interface

The application has several modules, each module has their own usage. The modules are main menu, training, detection, recognition, testing, help, and about. Main menu module is the main page of the application where user would use it for navigating to other modules. Training module is divided into 2 parts, such as detection and recognition. The detection module is for training the data detector to differentiate between faces and non-faces images. The recognition module is for recognizing the owner of the faces that has been detected before. Moreover, the testing module is used for recognizing faces through face images input by loading face images data or capturing face images, then the face will be recognized by the application. The output is the name of the person in the image.



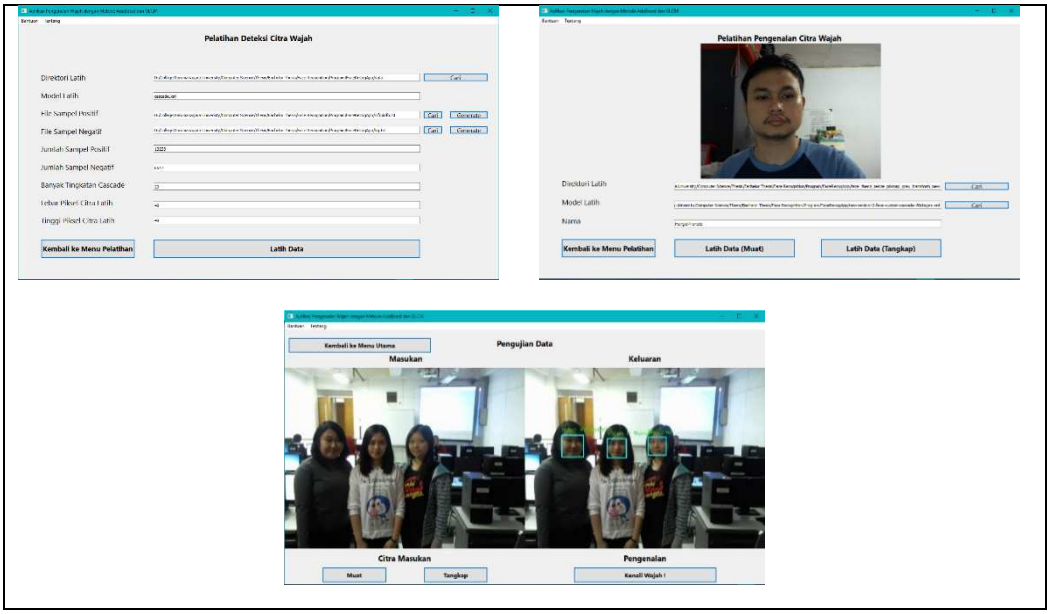


Figure. 4. Interfaces of the Application

4 Experimental Results

The face images dataset divided into 2 parts, the first one is the Labeled Faces in the Wild University of Massachusetts Amherst [8,9], and the second one is face images which was taken ourselves. The first dataset was obtained through online searching and can be used for research purposes, image on dataset was used for training face detector. The second dataset was obtained through collecting and capturing face images using webcam, image on dataset was used for training and testing face recognition.

The samples were taken on different spots, such as: classroom, laboratories room, organization room depends on time and place of the targeted students. In the process of taking face images on one place, we divided the process into two sessions. The first session was for training face images, and the second session was for testing face images.

For the first session, we took them one by one. The distance between the subject and camera was 1 meter. The subject was then captured in 5 different directions (frontal, up, down, left, and right), the purpose was to tolerate if we had titled image. For the second session, we captured the images in group, 2-4 persons in each image. However, they were captured only in one direction, frontal face. If not possible, probably because the face was too tight on the frame, then the face should be titled a bit as long as the face still facing the camera. There are several additional tests, to test the face directions, such as frontal, up, down, left, and right.

All the process of taking, training, and testing data was done using the developed application, except for Labeled Faces in the Wild University of Massachusetts Amherst which was taken manually and then trained using the application.

The results of application testing on the second dataset, resulted succession rate of 93,79% on detection, meanwhile the average of recognition success rate was 73,37% and the

average of detection and recognition success rate was 68.82%. The testing graphic for face recognition can be seen on Figure 9 and the testing graphic for face detection & recognition can be seen on Figure 10.

We tried 7 testing conditions which is train data = test data, testing data using KNN with k values such as k=1, k=2, and k=3, and testing data using KNN with geometric transformation on test images and k values such as k=1, k=2, and k=3. Lastly, we tested the distance between the image and the camera to measure how far the application be able to detect human faces. The result is listed on Table 1.

Table 1. Distance Experiment **Results**

Distance	Experimental results
50 cm	Success
100 cm	Success
150 cm	Success
200 cm	Success
250 cm	Fail

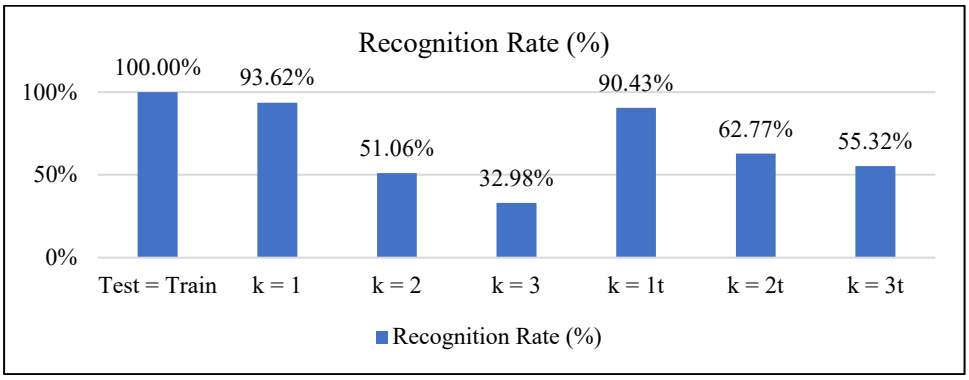


Figure. 5. Face Recognition Testing Graphic

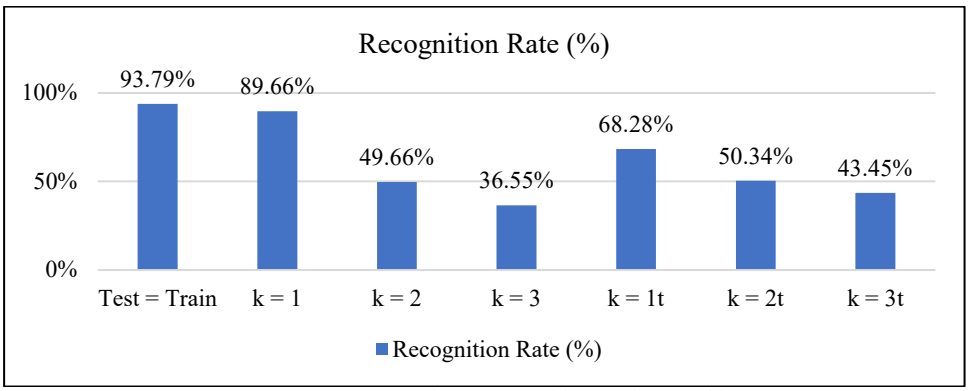


Figure. 6. Face Detection & Recognition Testing Graphic

4 Conclusion

AdaBoost is best for detecting faces in images. Using the weak classifier, AdaBoost was proven well on detecting faces. The success rate was 94%. However, GLCM was only successfully recognize 73% faces of the test images. Combining both, we only got 69% successful rate for testing human face detection and recognition.

Face orientations and additional objects on the subject face, such as glasses or moles could be the cause of the low successful rate of the combination process. However, this needs further research, since they were not the subject in this research.

References

[1] Viola, “, Paul and Jones, Michael. “International Journal of Computer Vision. Robust Real-time Object Detection. Vol. 57, no. 2. Berlin: Springer, May 2004.”

[2] Santi, “Rina Candra Noor. ‘Jurnal Teknologi Informasi Dinamik’. Mengubah Citra Berwarna Menjadi Gray-Scale dan Citra Biner, Vol. 16, no 1. Semarang: Universitas Stikubank , January 2011.”

[3] Kegl, “Balasz. Introduction to Adaboost. Orsay: Paris-Sud University, 2014.”

[4] OpenCV, “dev team. Cascade Classification, https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html, 24 August 2018.”

[5] Albregtsen, “Fritz. Statistical Texture Measures Computed from Gray Level Coocurrence Matrices. Oslo: Image Processing Laboratory Department of Infotmatics University of Oslo, 2008.”

[6] Sutton, “Oliver. Introduction to K Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction. Leicester: Department of Mathematics University of Leicester, 2012.”

[7] Weisstein, “Eric W. Distance, http://mathworld.wolfram.com/Distance.html, 2 October 2018.”

[8] Sanderson, “C. and Lovell, B.C. ‘Lecture Notes in Computer Science’. Multi-Region Probabilistic Histogram for Robust and Scalable Identity Inference. Vol. 5558, no. 1. Berlin: Springer, June 2009.”

[9] Huang, “G.B.; Ramesh, M.; Berg, T.; and Miller, E. Learned. ‘University of Massachusetts, Amherst, Technical Report’ Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. No. 7-49. Amherst: University of Massachusetts,.”