

Implementation of K-Nearest Neighbors Face Recognition on Low-power Processor

Eko Setiawan¹, Adharul Muttaqin²

Program of Information Technology and Computer Science, Brawijaya University,
Veteran Road No. 8 Malang, Jawa Timur, Indonesia 65145, Telp/Fax 0341 - 577 911

*Corresponding author, e-mail: ekosetiawan@ub.ac.id¹, adharul@ub.ac.id²

Abstract

Face recognition is one of early detection in security system. Automation encourages implementation of face recognition in small and compact devices. Most of face recognition research focused only on its accuracy and performed on high-speed computer. Face recognition that is implemented on low-cost processor, such as ARM processor, needs proper algorithm. Our research investigate K-Nearest Neighbor (KNN) algorithm in recognizing face on ARM processor. This research sought best k -value to create proper face recognition with low-power processor. The proposed algorithm was tested on three datasets that were Olivetti Research Laboratory (ORL), Yaleface and MUCT. OpenCV was chosen as main core image processing library, due to its high-speed. Proposed algorithm was implemented on ARM11 700MHz. 10-fold cross-validation showed that KNN face recognition detected 91.5% face with $k=1$. Overall experiment showed that proposed algorithm detected face on 2.66 s on ARM processor.

Keywords: face recognition, K-Nearest Neighbors, ARM processor

Copyright © 2015 Universitas Ahmad Dahlan. All rights reserved.

1. Introduction

Smart building becomes top issues in last years. Some researches keep doing to propose better smart building system. A smart building should be able to modify environment according to the occupant command. First, the system must identify who is its occupant. Natural identification technique is face recognition that gives better solution. Applying face recognition, system could identify the occupant without disturbing the other people.

For years, face recognition become top issues in research paper. Several techniques were applied to propose better recognition. Changing in pose, expression variations and differences in the position of the light give difficulty to resolve in face recognition. Basically, face recognition can be approached with facial biometric features or statistical method. Facial biometric recognition technique offers high accuracy with long calculation. Hence, statistical approaches offer speed calculation. Due to the speed, a statistical approach is applied more widely than facial biometric [1]. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two popular statistical approaches in face recognition.

Several studies had shown that PCA (known as Eigenface) and LDA (known as Fisherface) method had high accuracy in face recognition [2]. Both of these methods calculated eigenvalues of face image. Calculation of eigenvalues required long steps. Several methods were proposed to improve them in accuracy and speed. Arif improved the Fisherface efficiency by calculate it in two-dimension. The improvement gave good result in 100% accuracy [3]. Two-dimension calculation takes more steps with time-cost consequence. Wijaya combined holistic feature and linear discriminant analysis to reduce the training time and keep the accuracy [4].

However, the previous works were applied on computer equipped with high-power processor. Since energy conservation is one of key issues in smart building research, low power processor becomes interesting topic to be investigated and evaluated of face recognition. Based on Yong et al survey [2], K Nearest Neighbor (KNN) placed third rank after PCA and LDA on accuracy and KNN placed first rank on processing speed. In this study, we interest to measure how suitable KNN applied in low power processor compare with PCA/Eigenface and LDA/Fisherface.

2. K-Nearest Neighbor

K-Nearest Neighbor (KNN) is data classification method that can be used as face recognition method. Each pixel in face represents unique information. This paper recognized face based on each pixel classification. Face was determined by most class resulted in each pixel classification. In recognition, pixel matrix of face image should be reshape into vector before classification. The proposed KNN face recognition algorithm is described as follows:

1. Modify dimensions of M-row and N-column face matrix ($M \times N$) into face transpose vector ($1 \times MN$)
2. Arrange each face vector into matrix form ($K \times MN$) with K is number of training face images. Each row represents a single image and each column would represent same pixels position in each face image.
3. Modify testing image matrix into face transpose vector, as training images ($1 \times MN$).
4. Calculate the Euclidean Distance (d) of each column (i) in testing image (x) to each column (i) in training image (y).

$$d_E(x, y) = \sum_{i=1}^N \sqrt{x_i^2 - y_i^2}$$

5. Determine classification based on the shortest distance of whole column in each row.
6. Determine face recognition based on the k nearest neighbor and its distance.

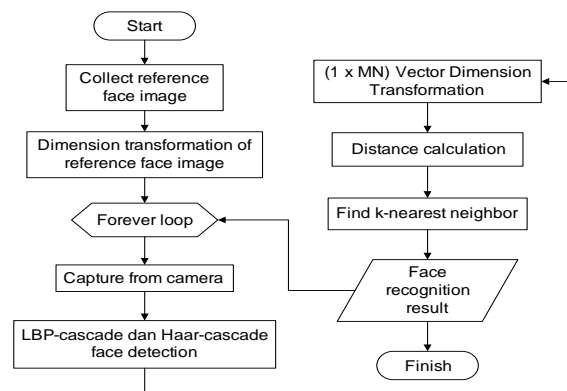


Figure 1. Face Recognition Flowchart

3. Implementation

The proposed method was implemented with OpenCV library [6] in C language to ensure fast image processing. Raspberry Pi which was equipped with ARM11 700MHz core, was selected to be main processing unit. Overall algorithm that was implemented in video face recognition system, was shown in Figure 1. Main process recognized face continuously by capturing a picture from attached webcam periodically. Captured image was pre-processed by applying face detection. Haar-cascade was applied due to its robustness in face detection. Pre-processed image would be processed by proposed face detection method. The whole process would be repeated so as to produce a video-based face recognition.

4. Results and Discussion

Tests were conducted to evaluate performance of proposed algorithm. Tests carried out with three face dataset which are ORL [7], Yaleface [8] and MUCT [9] dataset. Before applying as training dataset, every images was applied face detection and equalized in same size. The result of preprocessing showed that some faces could not be used as dataset. The 296 ORL faces, 156 Yalefaces and 3644 MUCT faces would be used as training and testing data. Figure 2 shows the overall images that are used in training and testing.

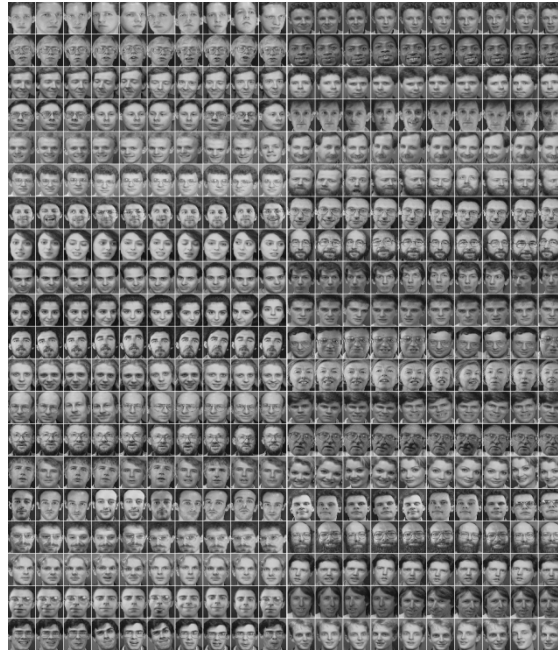


Figure 2. ATT Face Database [7]

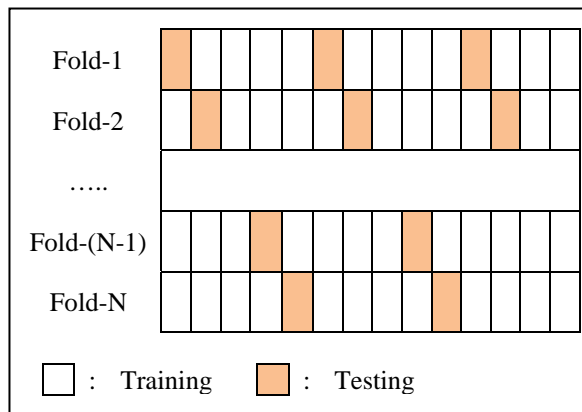


Figure 3. k-Fold Cross Validation

First testing was to find the k values that produced high accuracy. Second testing would compare accuracy and execution time of proposed method with common Eigenface and Fisherface. Final testing would observe time execution of continuous face recognition system in low-level processor.

Table 1. Cross Validation on ORL dataset

Number of 10-fold	Accuracy (%)			
	K=1	K=3	K=5	K=7
1	93.3	83.3	73.3	70.0
2	96.7	86.7	90.0	86.7
3	100	90.0	90.0	86.7
4	93.3	86.7	86.7	83.3
5	96.7	93.3	86.7	86.7
6	93.3	86.7	80.0	76.7
7	79.3	72.4	65.5	69.0
8	93.1	89.7	75.9	79.3
9	86.2	79.3	82.8	82.8
10	82.8	82.8	79.3	72.4
Average	91.5	85.1	81.0	79.3

Tabel 2. KNN face accuracy

K Value	Average Accuracy (%)		
	ORL dataset	Yaleface dataset	MUCT dataset
K=1	91.5	78.8	70.0
K=3	85.1	75.0	63.6
K=5	81.0	71.8	62.1
K=7	79.3	64.8	60.2

First testing was conducted in 10-fold Cross Validation. Figure 3 showed how to determine training and testing images. Accuracy on each fold was calculated. Tabel 1 was detail accuracy on each fold in ORL dataset. The total accuracy was calculated on average of 10-fold. Tabel 2 shows the results of accuracy in different k value on several dataset.

Based on Table 2, KNN done best accuracy on k equal to 1. It showed that KNN gave 91.5% on 295 ORL faces, 78.8% on 156 Yalefaces and 70% on 3644 MUCT faces. Enormous number of MUCT dataset made the system confuse in recognition. K was equal to 1 and ORL dataset would be used in next test due to its best result. The next phase test was to compare the proposed method with Eigenface and Fisherface. Tests carried out to obtain information on how feasible the proposed method when implementing in low-power processor. Tests conducted on computer with Intel Core i7, 2.8 GHz and Raspberry Pi with Broadcom ARM11, 700MHz. The accuracy and execution time would be compared. Accuracy testing was held on same scenario of k-value adjustment. The final results of testing were shown in Table 3 and Table 4.

Tabel 3. Face recognition on computer

Method	Accuracy (%)	Learning requirement	Learning time (s)	Recognition time (s)	Total time (s)
Eigenface	91.5	Yes	3.935875	0.006887	3.94276
Fisherface	91.5	Yes	2.997280	0.000540	2.99782
KNN	91.5	No	0	0.003689	0.00369

Tabel 4. Face recognition on low-power processor.

Method	Accuracy (%)	Learning requirement	Learning time (s)	Recognition time (s)	Total time (s)
Eigenface	91.5	Yes	299.221	0.459	299.680
Fisherface	91.5	Yes	234.252	0.027	234.279
KNN	91.5	No	0	0.152	0.152

Table 3 showed that face recognition execution time on computer was not more than 4 seconds. Table 4 informed that execution time of fisherface and eigenface increased significantly into 299.7 seconds and 234.3 seconds respectively in low-power processor. Both of these popular methods were not appropriate low-power processor due to its long execution time. The proposed method took shorter time than the eigen or fisherface. This fact confirmed that KNN was proper face recognition method in low-power processor. On next scenario, KNN would be selected as the recognition method.

Final test aimed to observe performance of continuous face recognition on low-power processor. Experiment applied face detection by Local Binary Pattern and Haar-Cascade. Detection result was recognized by KNN. Experiment was done by showing printed test image on online camera. The camera captured image. Figure 4 shows the face detection and recognition process on low-power processor.

Tabel 4. CPU Load and memory usage.

Camera Pixel	CPU Load (%)	Memory Usage (MB)
1024 x 768	84	67.6
960 x 640	79	62.8
640 x 480	82	59.1
320 x 240	85	53.2
160 x 120	75	51.5

Tabel 5. Execution time of face recognition

Trial	Execution time (s)	Trial	Execution time (s)
1	2.32	16	2.73
2	2.76	17	2.70
3	2.67	18	2.66
4	2.65	19	2.70
5	2.81	20	2.61
6	2.91	21	2.61
7	2.92	22	2.67
8	2.29	23	2.42
9	2.56	24	2.28
10	2.52	25	2.85
11	2.52	26	2.79
12	2.45	27	2.71
13	2.45	28	2.95
14	2.66	29	2.85
15	2.68	30	3.00
Average time (s)		2.66	

Experiment evaluated CPU load and memory usage of continuous face recognition on several camera pixel size. The system show different CPU Load and memory on each resolution. Detail CPU Load and memory usage was shown clearly on Table 4. Smaller camera resolution delivered smaller memory usage. CPU load shown anomaly between various resolution. The experiment shown that 640 x 480 resolution gave optimum in CPU load and memory usage. The resolution produced not high CPU load and memory usage. Experiment also performed face recognition 30 times sequentially and recorded each execution time. Based on Table 5, the average time of the face recognition process was 2.66 seconds on embedded systems. Test results showed that KNN face recognition was feasible to be implemented on embedded systems.

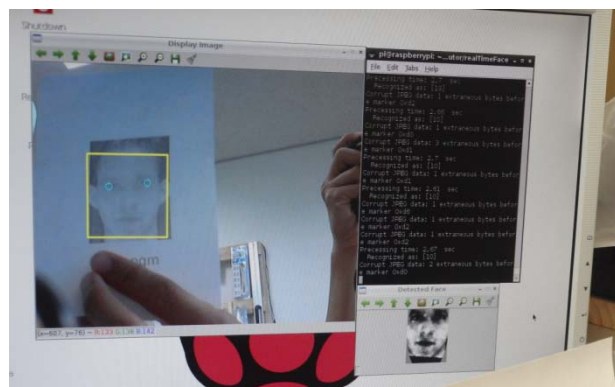


Figure 4. Continuous face recognition

5. Conclusion

The research concluded some informations about face recognition on low-power processor. K-Nearest Neighbor face recognition delivered best accuracy 91.5% on $k=1$. KNN showed the faster execution time compared with PCA and LDA. Time execution of KNN to recognize face was 0.152 seconds on high-processor. Face detection and recognition only need 2.66 second to recognize on low-power ARM11 based system . Overall proposed method work well on low-power processor. Appropriate face detection on low-power system have potentialy to boost recognition.

References

- [1] Rabia J, Hamid RA. A Survey of Face Recognition Techniques. *Journal of Information Processing*. 2009; 5(2).

-
- [2] Peter NB, Joao PH, David JK. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. 2014.
 - [3] Arif M. New Modelling of Modified Two Dimensional Fisherface Based Feature Extraction. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2014; 12(1): 115-122.
 - [4] IGPS Wijaya, K Uchimura, G Koitaki. Face Recognition Using Holistic Features and Linear Discriminant Analysis Simplification. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2012; 10(4): 771-782.
 - [5] Yong X, Xiao zhao F, Xuelong L, Jian Y, Jane Y, Hong L, Shaohua T. Data Uncertainty in Face Recognition. *IEEE Transaction on Cybernetics*. 2014.
 - [6] OpenCV Dev Team. 2014. The OpenCV Reference Manual Release 2.4.9.0. accessed on September 10th 2014.
 - [7] F Samaria, A Harter. *Parameterisation of a Stochastic Model for Human Face Identification*. 2nd IEEE Workshop on Applications of Computer Vision. Sarasota (Florida). 1994.
 - [8] P Belhumeur, J Hespanha, D Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1997: 711-720.
 - [9] S Milborrow, J Morkel, F Nicolls. The MUCT Landmarked Face Database. *Pattern Recognition Association of South Africa*. 2010.