



INSTITUT TEKNOLOGI INDONESIA

**IMPLEMENTASI SISTEM FACE RECOGNITION DALAM
APLIKASI KEHADIRAN OTOMATIS DENGAN MENGGUNAKAN
ALGORITMA LBPH (*LOCAL BINARY PATTERN HISTOGRAM*)**

SKRIPSI

MUHAMMAD NAUFAL MAHENDRA

1151700054

TEKNIK INFORMATIKA

TANGERANG SELATAN

2022

BAB 4

IMPLEMENTASI DAN PENGUJIAN

Pada Bab ini akan menguji dan implementasi untuk menjelaskan mengenai perangkat yang akan digunakan dalam membangun sebuah model sistem pengenalan wajah untuk melakukan kehadiran otomatis dalam aplikasi “*Smart Attendance Recognition*”. dengan menggunakan metode *Haar Cascade Classifier* dan *Local Binary Patterns Histogram* (LBPH). Perangkat keras (*hardware*) dan perangkat lunak (*software*) yang akan dijelaskan berikut ini.

4.1 Perangkat Keras (*hardware*)

Pada kebutuhan perangkat keras (*hardware*) yang digunakan untuk membangun sebuah model sistem pengenalan wajah untuk dapat mengenali pengguna dan melakukan kehadiran. Akan ditunjukkan pada Tabel 4.1 sebagai berikut.

Tabel 4.1 Spesifikasi perangkat keras

Perangkat Keras	Spesifikasi Minimum	Spesifikasi yang Digunakan
Processor	Intel Celeron (1.4 GHz) -64bit	Intel® Core™ i7-8565U
Graphic Card	Ethernet Card	NVIDIA® GeForce® MX150
Hardisk	500GB	512GB
RAM	2GB DDR3	8GB 2400MHz DDR4 Memory
WebCam	420p HD Camera	720p HD Camera

4.1.2 Perangkat Lunak (*software*)

Perangkat lunak (*software*) yang digunakan pada masing-masing perangkat keras (*hardware*) yang sebelumnya telah dijelaskan, dalam membangun sebuah model kesesuaian metode *face recognition* untuk dapat mengidentifikasi seseorang. Akan ditunjukkan pada Tabel 4.2 sebagai berikut.

Tabel 4.2 Spesifikasi perangkat lunak

Perangkat Lunak	Spesifikasi
Operating System	Windows 10 64 bit Home
Microsoft Visual Code	1.63 Version
Python	3.9 Version
Google Chrome	Version 98.0.4758.82 (Official Build) (64-bit)

4.2 Pengambilan Dataset

Dataset yang digunakan dalam implementasi ini berupa *dataset* yang dibuat sendiri dengan mengambil gambar wajah yang ingin diidentifikasi melalui kamera secara *real time*. Setelah itu citra tersebut akan diolah menggunakan metode pendeteksian wajah berupa *Haar Cascade*. Wajah yang terdeteksi ditunjukkan berupa garis bujur sangkar ROI (*Region of Interest*) yang digunakan adalah objek (muka)

Untuk dapat melakukan mengidentifikasi sebuah objek (muka), *user* harus melakukan pengambilan gambar terlebih dahulu secara *real time* sebanyak 100 gambar untuk dijadikan dataset. Maka implementasi *source code* untuk melakukan *open* kamera untuk pengambilan gambar secara *real time*. Akan ditampilkan pada *source code* Gambar 4.1 sebagai berikut.

```
cam = cv2.VideoCapture(1)
cam.set(3, 640) # set video width
cam.set(4, 480) # set video height
```

Gambar 4.1 Source code untuk menampilkan kamera webcam

Kemudian, python akan membaca video *real time* yang ada pada laptop. Setelah itu memberi program pada python agar dapat mendeteksi mana yang merupakan wajah dan mana yang bukan merupakan wajah dengan perintah *source code* pada gambar 4.2 sebagai berikut:

```
pendeteksi_muka = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Gambar 4.2 Source code mendeteksi wajah

Kemudian menggambar ROI (*Region Of Interest*) dalam bentuk kotak berwarna biru sesuai dengan koordinat dan dimensinya yaitu (nilai x,y,w,h). maka contoh *source code* pada gambar 4.3 sebagai berikut :

```
cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
```

Gambar 4.3 Source code menggambar ROI (Region Of Interest)

Kemudian python akan meng-*capture* wajah *user* yang akan melakukan identifikasi wajah akan diminta untuk memasukan id untuk dapat menyimpan inisial dari nama *user* yang akan disimpan kedalam dataset, setiap wajah harus ditandai dengan no Id. Maka contoh *source code* pada gambar 4.4 sebagai berikut:

```
face_id = input('\n Masukan nomor id dan tekan enter ==> ')
```

Gambar 4.4 Source code untuk memasukan Id User

Lalu sistem akan mendeteksi wajah. Jika wajah terdeteksi maka akan ditunjukkan muncul tanda berupa garis bujur sangkar ROI (*Region of Interest*) pada wajah. Lalu akan muncul sebuah tampilan video untuk meng-*capture* wajah *user* sampai 100 gambar. Yang akan ditampilkan pada gambar 4.5 sebagai berikut.



Gambar 4.5 Pengambilan gambar user

Data gambar yang telah melakukan identifikasi wajah akan memproses dan hasil dari pengolahan yang sudah dilakukan tersimpan pada folder “dataset” yang berada dalam *directory* yang sama dengan *source code*. Lalu data akan mentransformasi berbentuk *grayscale*, oleh karena itu masing-masing data gambar yang ditangkap akan dikonversi menjadi *grayscale* akan ditampilkan source code pada gambar 4.6 dengan menggunakan fungsi sebagai berikut:




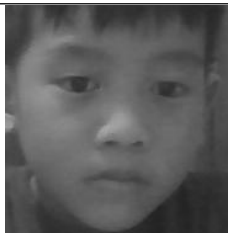








```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Gambar 4.6 Fungsi mengubah citra RGB menjadi grayscale

4.3 Hasil Pendeteksian Fitur Wajah

Proses pertama yang dilakukan oleh metode *Haar Cascade Classifier* untuk mendeteksi adanya fitur wajah pada sebuah gambar adalah dengan merubah gambar tersebut menjadi citra *grayscale* dan data yang akan diolah akan dilakukan proses *resize* (mengubah ukuran gambar) diubah menjadi dimensi yang berbeda-beda sesuai dengan rasio wajah yang tertangkap kamera. Proses ini dilakukan untuk dapat memudahkan proses pembelajaran sistem (*training*). Setelah berhasil diambil sebanyak 100 gambar *user* dari hasil yang telah didapat berupa citra *grayscale*, Maka pengolahan yang sudah dilakukan tersimpan pada folder “dataset” yang berada dalam *directory* yang sama dengan *source code*. Maka salah satu contoh hasil dari pengambilan gambar yang telah ditransformasi menjadi citra *grayscale* akan ditampilkan pada Tabel 4.3 sebagai berikut.

Tabel 4.3 Hasil pengolahan dataset yang sudah dikonversi menjadi grayscale

Sampel 1		Sampel 2	
			
Sampel 3		Sampel 4	
			
Sampe 5		Sampel 6	
			

4.4 Proses Pembelajaran Sistem (*Training*)

Sebelum melakukan proses *training*, terdapat modul dari *numpy*, *pillow* yang akan digunakan. Modul-modul tersebut akan diimport terlebih dahulu seperti yang ditampilkan pada Gambar 4.7 berikut.

```
import cv2
import numpy as np
from PIL import Image
import os
```

Gambar 4.7 Importing package dan modul training

Pada proses *training* atau pembelajaran sistem ini akan menggunakan algoritma *Local Binary Patterns Histogram* (LBPH). Algoritma *Local Binary Pattern Histogram* (LBPH) akan dapat mengenali gambar dalam *folder dataset* secara terus-menerus untuk melatih sistem sehingga dapat mengenali gambar wajah tersebut dengan tepat. Maka ditampilkan source code pada Gambar 4.8 sebagai berikut:

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
```

Gambar 4.8 Metode *Local Binary Pattern Histogram*

Kemudian membuat suatu fungsi perulangan untuk melatih sistem dalam mengenali gambar secara benar. Berikut adalah perintah *source code* pada gambar 4.9

```
def dapatGambarDanLabel(path):
```

Gambar 4.9 Source code perulangan untuk melatih

Lalu dibutuhkan perintah untuk mengambil data dari *folder dataset* lalu akan melakukan *training* untuk keseluruhan dan menelusuri gambar yang ada pada *folder dataset*. Yang mana perintah source code pada gambar 4.10 berikut :

```
imagePath = [os.path.join(path,f) for f in os.listdir(path)]
```

Gambar 4.10 Source code untuk melakukan training pada seluruh folder dataset

Selanjutnya akan dibuat definisi untuk gambar wajah dan labelnya dengan perintah *source code* pada gambar 4.11 sebagai berikut :

```
sample_muka=[]  
ids = []
```

Gambar 4.11 Source code untuk membuat definisi wajah dan label

Lalu untuk mempelajari setiap gambar wajah, dapat menggunakan perulangan pada gambar 4.12 perintah *source code* berikut ini , yang mana perintah itu digunakan untuk mengkonversi gambar menjadi *grayscale* terlebih dahulu lalu dikonversikan lagi menjadi bentuk *array*.

```
for imagePath in imagePath:  
PIL_img = Image.open(imagePath).convert('L')
```

Gambar 4.12 Source code untuk mengkonversi menjadi grayscale dan array

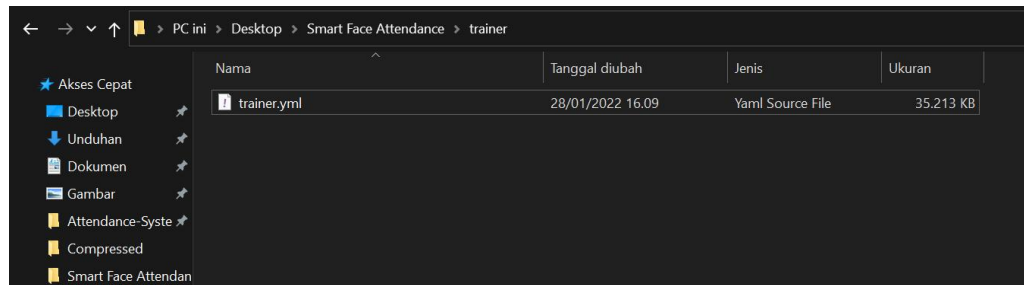
Kemudian dibutuhkan juga modul untuk mengelola yaitu PIL(*Python Imaging Library*) yang akan digunakan untuk membuka, memanipulasi dan menyimpan diri dari berbagai format *file* gambar dengan perintah *source code* pada gambar 4.13.

```
PIL_img = Image.open(imagePath).convert('L')  
img_numpy = np.array(PIL_img,'uint8')  
  
for (x,y,w,h) in muka:  
    sample_muka.append(img_numpy[y:y+h,x:x+w])  
    ids.append(id)  
  
return sample_muka,ids
```

Gambar 4.13 Source code untuk mengelola PIL (Python Imaging Library)

Jika semua pengulangan telah selesai, maka akan disimpan dalam sebuah file dengan ekstensi “trainer.yml”. Selanjutnya dapat mengambil semua data *user* dari *dataset* yang telah dibuat sebelumnya lalu membuat folder “trainer” dalam *directory* yang sama dengan *source code* selanjutnya di proses menggunakan **OpenCV Recognizer** dan dilakukan secara langsung dengan fungsi **OpenCV**. Lalu hasil dari proses training tersebut adalah dalam bentuk file “trainer.yml” yang akan disimpan pada

folder “trainer”. Maka untuk hasil training dalam bentuk file “trainer.yml” tersebut akan ditampilkan pada Gambar 4.14



Gambar 4.14 Directory hasil training dataset

4.5 Pengujian Sistem Identifikasi Wajah

Dalam proses pengujian (*testing*) ini dilakukan saat menguji akurasi dan kesesuaian metode *face recognition* untuk dapat mengidentifikasi seseorang. Tujuan dari pengujian dari proses ini adalah untuk memvalidasi bahwa sistem yang dibangun berjalan dengan baik.

Proses ini dilakukan dengan metode *Haar cascade classifier* dan *Local Binary Pattern Histogram* (LBPH) dengan menggunakan 6 Sampel wajah yang berbeda. Proses pengujian ini akan dilakukan dengan cara merekam atau meng-*capture* wajah *user* sebanyak 100 gambar lalu merubah gambar menjadi *grayscale*. Kemudian akan tersimpan kedalam folder “dataset” yang berada dalam *directory* yang sama dengan *source code* selanjutnya kumpulan *dataset* tersebut akan dilatih menggunakan metode *Local Binary Pattern Histogram* (LBPH) , setelah berhasil dilatih akan tersimpan pada folder trainer dan hasil pelatihan dataset tadi akan mengeluarkan *output* berupa file “trainer.yml”. Lalu selanjutnya menjalankan file “face_recognition” untuk melakukan pengujian identifikasi wajah dapat dilakukan ketika proses training dataset telah berhasil.

Akan dilakukan pengujian akurasi pengenalan wajah akan disajikan dalam bentuk tabel agar lebih mudah dimengerti oleh pembaca. Tabel penyajian akan memiliki 4 kolom yaitu no, jarak, hasil, dapat mengenali wajah ya/tidak. Untuk dapat melakukan pengujian menggunakan parameter seperti berikut ini:

- A. Pengaruh jarak terhadap tingkat keberhasilan dan akurasi pengenalan wajah yang sudah tersimpan di database. Tabel 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 merupakan hasil uji coba terhadap jarak untuk pengenalan wajah.
- B. Pengaruh tingkat kemiringan wajah terhadap keberhasilan dan akurasi pengenalan wajah yang sudah tersimpan di *database*. Tabel 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 merupakan hasil dari pengujian terhadap kemiringan wajah untuk pengenalan wajah.

A. Parameter Pertama





Tabel 4.4 Hasil Pengujian Sampel 1

No	Jarak	Hasil	Mengenal Wajah	
			Ya	Tidak
1.1	40 cm		✓	
1.2	70 cm		✓	
1.3	100 cm		✓	
1.4	180 cm			✓





Tabel 4.5 Hasil Pengujian Sampel 2

No	Jarak	Hasil	Mengenali Wajah	
			Ya	Tidak
1.2.1	40 cm		✓	
1.2.2	70 cm		✓	
1.2.3	100 cm		✓	
1.2.4	180 cm			✓

Tabel 4.6 Hasil Pengujian Sampel 3

No	Jarak	Hasil	Mengenal Wajah	
			Ya	Tidak
1.3.1	40 cm		✓	
1.3.2	70 cm		✓	
1.3.3	100 cm		✓	
1.3.4	180 cm			✓





Tabel 4.7 Hasil Pengujian Sampel 4

No	Jarak	Hasil	Mengenal Wajah	
			Ya	Tidak
1.4.1	40 cm		✓	
1.4.2	70 cm		✓	
1.4.3	100 cm		✓	
1.4.4	180 cm			✓

Tabel 4.8 Hasil Pengujian Sampel 5






No	Jarak	Hasil	Mengenai Wajah	
			Ya	Tidak
1.5.1	40 cm		✓	
1.5.2	70 cm		✓	
1.5.3	100 cm		✓	
1.5.4	180 cm			✓

Tabel 4.9 Hasil Pengujian Sampel 6






No	Jarak	Hasil	Mengenal Wajah	
			Ya	Tidak
1.6.1	40 cm		✓	
1.6.2	70 cm		✓	
1.6.3	100 cm		✓	
1.6.4	180 cm			✓

B. Parameter Kedua






Tabel 4.10 Hasil Pengujian Sampel 1

No	Drajat kemiringan wajah	Hasil	Mengenal Wajah	
			Ya	Tidak
2.1.1	Tegak lurus		✓	
2.1.2	10° Ke kanan		✓	
2.1.3	10° Ke kiri		✓	
2.1.4	10° Ke atas		✓	
2.1.5	40° Ke atas			✓






Tabel 4.11 Hasil Pengujian Sampel 2

No	Drajat kemiringan wajah	Hasil	Mengenal Wajah	
			Ya	Tidak
2.2.1	Tegak lurus		✓	
2.2.2	10° Ke kanan		✓	
2.2.3	10° Ke kiri		✓	
2.2.4	10° Ke atas		✓	
2.2.5	40° Ke atas			✓






Tabel 4.12 Hasil Pengujian Sampel 3

No	Drajat kemiringan wajah	Hasil	Mengenal Wajah	
			Ya	Tidak
2.3.1	Tegak lurus		✓	
2.3.2	10° Ke kanan		✓	
2.3.3	10° Ke kiri		✓	
2.3.4	10° Ke atas		✓	
2.3.5	40° Ke atas			✓






Tabel 4.13 Hasil Pengujian Sampel 4

No	Drajat kemiringan wajah	Hasil	Mengenal Wajah	
			Ya	Tidak
2.4.1	Tegak lurus		✓	
2.4.2	10° Ke kanan		✓	
2.4.3	10° Ke kiri		✓	
2.4.4	10° Ke atas		✓	
2.4.5	40° Ke atas			✓

Tabel 4.14 Hasil Pengujian Sampel 5

No	Drajat kemiringan wajah	Hasil	Mengenal Wajah	
			Ya	Tidak
2.5.1	Tegak lurus		✓	
2.5.2	10° Ke kanan		✓	
2.5.3	10° Ke kiri		✓	
2.5.4	10° Ke atas		✓	
2.5.5	40° Ke atas			✓

Tabel 4.15 Hasil Pengujian Sampel 6

No	Drajat kemiringan wajah	Hasil	Mengenal Wajah	
			Ya	Tidak
2.6.1	Tegak lurus		✓	
2.6.2	10° Ke kanan		✓	
2.6.3	10° Ke kiri		✓	
2.6.4	10° Ke atas		✓	
2.6.5	40° Ke atas			✓

4.5.6 Hasil Pengujian dan Analisa

A. Parameter Pertama

Pada parameter pertama menunjukkan bahwa terdapat wajah yang dapat dideteksi dan dikenali pada jarak 40 cm hingga pada jarak 200 cm sudah tidak dapat mengenali wajah tetapi masih dapat mendeteksi wajah.

Total Pengujian Benar	20
Total Pengujian Salah	4
Total Data Pengujian	24

Lalu berdasarkan Tabel 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 yaitu dapat disimpulkan bahwa dari 6 sampel data wajah orang yang berbeda, terdapat 20 pengujian yang dapat dikenali oleh sistem dengan benar, dan terdapat 4 pengujian gambar yang tidak dapat dikenali tapi masih bisa mendeteksi wajah.

Cara untuk menghitung nilai akurasi dan kesalahan dari proses pengujian diatas dapat dihitung dengan menggunakan persamaan berikut ini.

$$\text{Akurasi} = \frac{\text{Jumlah data yang benar}}{\text{Jumlah seluruh data}} \times 100\%$$

$$\text{Akurasi} = \frac{20}{24} \times 100\%$$

$$\text{Akurasi} = 83.33\%$$

Sedangkan proses perhitungan untuk data yang tidak berhasil dikenali atau dideteksi adalah sebagai berikut.

$$\text{Akurasi} = \frac{\text{Jumlah data yang salah}}{\text{Jumlah seluruh data}} \times 100\%$$

$$\text{Akurasi} = \frac{4}{24} \times 100\%$$

$$\text{Akurasi} = 16.67\% \text{ atau } 17\%$$

Dari perhitungan akurasi dan kesalahan di atas, didapat bahwa persentase nilai akurasi sistem pengenalan wajah untuk melakukan kehadiran otomatis. sebesar 83.33% dengan kesalahan sebesar 16.67% atau jika dibulatkan menjadi 17%.

B. Parameter Kedua

Pada parameter kedua menunjukkan bahwa tingkat kemiringan mempengaruhi pengenalan wajah. Dalam uji coba ini penulis melakukan kemiringan wajah ke kanan, ke kiri dan ke atas. Lalu ketika tingkat kemiringan kurang lebih 40 derajat wajah sudah tidak bisa dideteksi dan dikenali.

Total Pengujian Benar	24
Total Pengujian Salah	6
Total Data Pengujian	30

Lalu berdasarkan dari tabel 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 yaitu dapat disimpulkan bahwa dari 6 sampel data wajah orang yang berbeda terdapat 30 pengujian yang dapat dikenali oleh sistem dengan benar, dan terdapat 6 pengujian gambar yang tidak dapat terdeteksi.

Cara untuk menghitung nilai akurasi dan kesalahan dari proses pengujian diatas dapat dihitung dengan menggunakan persamaan berikut ini.

$$Akurasi = \frac{Jumlah\ data\ yang\ benar}{Jumlah\ seluruh\ data} \times 100\%$$

$$Akurasi = \frac{24}{30} \times 100\%$$

$$Akurasi = 80.00\%$$

Sedangkan proses perhitungan untuk data yang tidak berhasil dikenali atau dideteksi adalah sebagai berikut.

$$Akurasi = \frac{Jumlah\ data\ yang\ salah}{Jumlah\ seluruh\ data} \times 100\%$$

$$Akurasi = \frac{6}{30} \times 100\%$$

$$Akurasi = 20.00\%$$

Dari perhitungan akurasi dan kesalahan di atas, didapat bahwa persentase nilai akurasi sistem pengenalan wajah untuk melakukan kehadiran otomatis dalam aplikasi “*Smart Attendance Recognition*” sebesar 80.00% dengan kesalahan sebesar 20.00% atau jika dibulatkan menjadi 20%.

4.5.1 Database Sistem Kehadiran

Dari hasil pendeteksian dan pengenalan yang sudah dilakukan akan mengeluarkan *output* dalam bentuk file c.csv yang bernama absen.csv berfungsi sebagai penyimpanan data nama , tanggal dan waktu ketika melakukan presensi. Berikut tampilan *database* pada sistem kehadiran pengenalan wajah ini dapat dilihat pada Gambar 4.8

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name , Date, Time														
2															
3	Naufal,10:37:27,Tuesday-February-01-2022														
4	Rasya,10:37:34,Tuesday-February-01-2022														
5	Keyla,10:41:02,Tuesday-February-01-2022														
6	Nurahmawati,10:41:04,Tuesday-February-01-2022														
7	Ivan,12:24:06,Wednesday-February-02-2022														
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															