

**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.

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 ditunjukan pada Tabel 4.1 sebagai berikut.

**Tabel 4.1 Spesifikasi perangkat keras**

|  |  |  |
| --- | --- | --- |
| **Perangkat Keras** | **Spesifikasi Minimum** | **Spesifikasi yang Digunakan** |
| 1. Processor | Intel Celeron (1.4 GHz) -64bit | Intel® Core™ i7-8565U |
| 1. Graphic Card | Ethernet Card | NVIDIA® GeForce® MX150 |
| 1. Hardisk | 500GB | 512GB |
| 1. RAM | 2GB DDR3 | 8GB 2400MHz DDR4 Memory |
| 1. 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 ditunjukan pada Tabel 4.2 sebagai berikut.

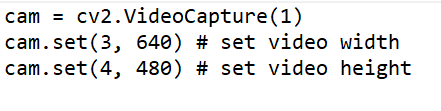
*Tabel 4.2 Spesifikasi perangkat lunak*

|  |  |
| --- | --- |
| **Perangkat Lunak** | **Spesifikasi** |
| 1. Operating System | Windows 10 64 bit Home |
| 1. Microsoft Visual Code | 1.63 Version |
| 1. Python | 3.9 Version |
| 1. Google Chrome | Version 98.0.4758.82 (Official Build) (64-bit) |

* 1. **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 identifikasi 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.



*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:



*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 :

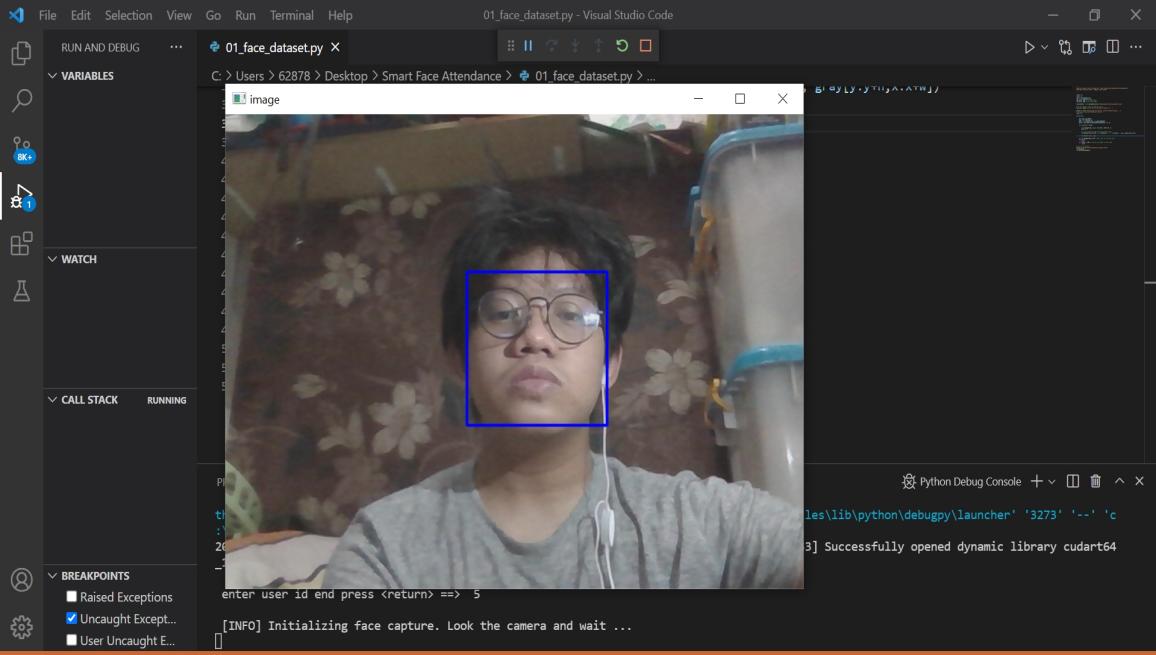


*Gambar 4.3 Source code menggambarkan ROI (Region Of Interest)*

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:

*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. Maka akan muncul sebuah tampilan video untuk meng-*capture* wajah user sampai 100 gambar. Akan ditampilkan pada gambar 4.4 sebagai berikut.



*Gambar 4.4 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 4.3 dengan menggunakan fungsi sebagai berikut:



*Gambar 4.3 Fungsi mengubah citra RGB menjadi grayscale*

* 1. **Pengolahan *Dataset***

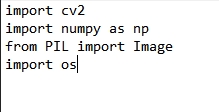
Proses pertama yang dilakukan oleh metode *Haar Cascade Classifier* untuk mendeteksi adanya fitur wajah pada sebuah gambar adalah dengan merubah gambartersebutmenjadi citra *grayscale* dan data yang akan diolah akan dilakukan proses *resize* (mengubah ukuran gambar) diubah menjadi dimensi yang berbeda-beda sesuai dengan rasio wajah yang tertangkap kamera. Proses ini dilakukan untuk dapat memudahkan proses pembelajaran sistem (*training*). Lalu hasil dari pengolahan yang sudah dilakukan tersimpan pada folder “dataset” yang berada dalam *directory* yang sama dengan *source code*. Maka 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 |
| User.2.57 | User.4.79.jpg |
| Sampel 3 | Sampel 4 |
| C:\Users\62878\AppData\Local\Microsoft\Windows\INetCache\Content.Word\User.6.1.jpg | User.3.58.jpg |

* 1. **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 4.4 berikut.



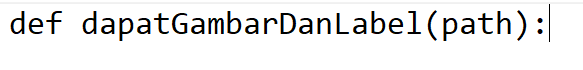
*Gambar 4.4 Importing package dan modul training*

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 4.5 sebagai berikut:



*Gambar 4.5 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 4.6



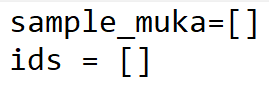
*Gambar 4.6 Source code perulangan untuk melatih*

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 4.7 berikut :



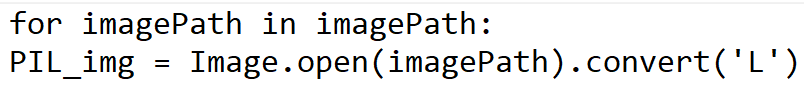
*Gambar 4.7 Source code untuk melakukan training pada seluruh folder dataset*

Selanjutnya akan dibuat definisi untuk gambar wajah dan labelnya dengan perintah source code pada gambar 4.8 sebagai berikut :



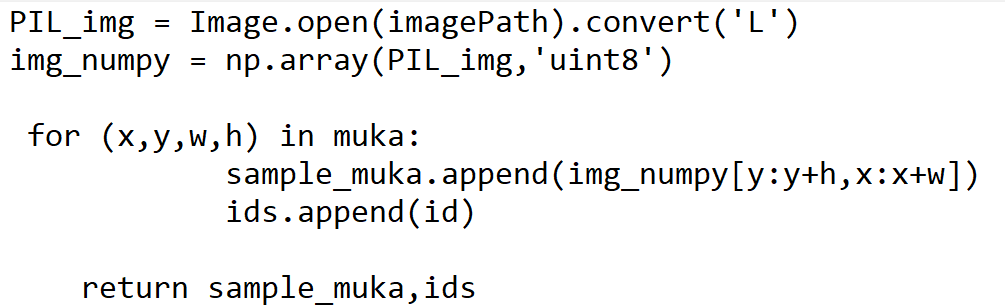
*Gambar 4.8 Source code untuk membuat definisi wajah dan label*

Lalu untuk mempelajari setiap gambar wajah, dapat menggunakan perulangan pada gambar 4.9 perintah *source code* berikut ini , yang mana perintah itu digunakan untuk mengkonversi gambar menjadi *grayscale* terlebih dahulu lalu dikonversikan lagi menjadi bentuk *array*.

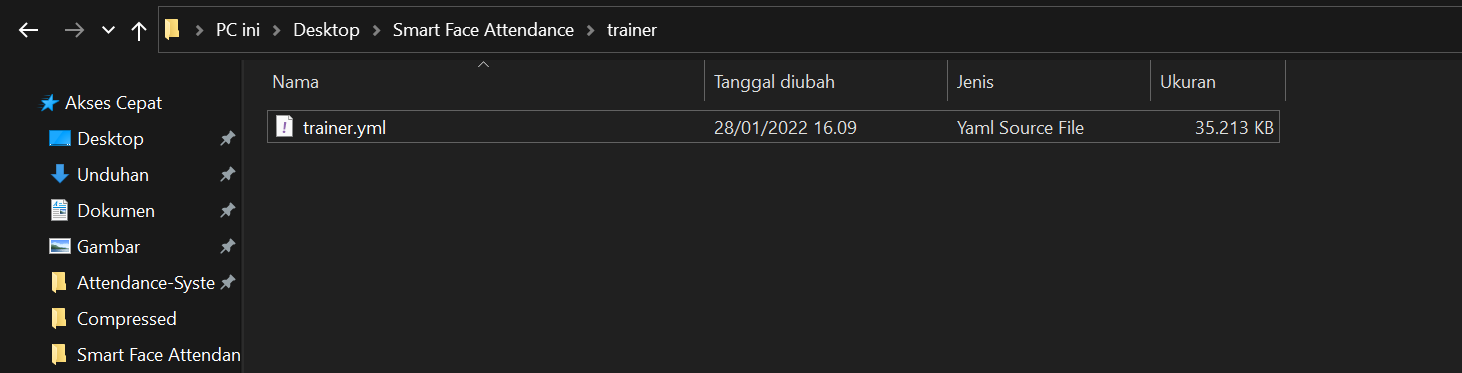


*Gambar 4.9 Source code untuk mengkonversi menjadi grayscale dan array*

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.



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 4.6



*Gambar 4.6 Directory hasil training dataset*

* 1. **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 gambarlalu 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.



*Gambar 4.7 Source code untuk inisialisasi id yang sudah terdaftar*

* + 1. **Hasil Pengujian dan Analisa**

Hasil 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 ,mengenal wajah ya/tidak. Untuk dapat melakukan pengujian menggunakan parameter seperti berikut ini:

1. Pengaruh jarak terhadap tingkat keberhasilan dan akurasi pengenalan wajah yang sudah tersimpan di database. Tabel 4.3, 4.4, 4.5 , 4.6 merupakan hasil uji coba terhadap jarak untuk pengenalan wajah.

*Tabel 4.3 Hasil Pengujian Sampel 1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Jarak | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 1.1 | 40 cm | Screenshot_81.jpg | ✓ |  |
| 1.2 | 70 cm | Screenshot_83.jpg | ✓ |  |
| 1.3 | 100 cm | Screenshot_85.jpg | ✓ |  |
| 1.4 | 180 cm | Screenshot_91.jpg |  | ✓ |

*Tabel 4.3 Hasil Pengujian Sampel 2*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Jarak | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 1.2.1 | 40 cm | Screenshot_32 | ✓ |  |
| 1.2.2 | 70 cm | Screenshot_33 | ✓ |  |
| 1.2.3 | 100 cm | Screenshot_34 | ✓ |  |
| 1.2.4 | 180 cm | Screenshot_35 |  | ✓ |

*Tabel 4.3 Hasil Pengujian Sampel 3*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Jarak | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 1.3.1 | 40 cm | Screenshot_27.jpg | ✓ |  |
| 1.3.2 | 70 cm | Screenshot_47 | ✓ |  |
| 1.3.3 | 100 cm | Screenshot_48 | ✓ |  |
| 1.3.4 | 180 cm | Screenshot_49.jpg |  | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Jarak | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 1.4.1 | 40 cm | Screenshot_72.jpg | ✓ |  |
| 1.4.2 | 70 cm | Screenshot_73.jpg | ✓ |  |
| 1.4.3 | 100 cm | Screenshot_76.jpg | ✓ |  |
| 1.4.4 | 180 cm | Screenshot_75.jpg |  | ✓ |

*Tabel 4.3 Hasil Pengujian Sampel 4*

*Tabel 4.3 Hasil Pengujian Sampel 5*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Jarak | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 1.5.1 | 40 cm | Screenshot_31.jpg | ✓ |  |
| 1.5.2 | 70 cm | Screenshot_41.jpg | ✓ |  |
| 1.5.3 | 100 cm | Screenshot_42.jpg | ✓ |  |
| 1.5.4 | 180 cm | Screenshot_43.jpg |  | ✓ |

*Tabel 4.3 Hasil Pengujian Sampel 6*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Jarak | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 1.6.1 | 40 cm | Screenshot_63.jpg | ✓ |  |
| 1.6.2 | 70 cm | Screenshot_64.jpg | ✓ |  |
| 1.6.3 | 100 cm | Screenshot_65.jpg | ✓ |  |
| 1.6.4 | 180 cm | Screenshot_66.jpg |  | ✓ |

Pada Tabel 4.3 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 dari tabel Tabel 4.3, 4.4, 4.5, 4.6 di atas yaitu, dapat disimpulkan bahwa dari 5 sampel data wajah orang yang berbeda, terdapat 15 pengujian yang dapat dikenali oleh sistem dengan benar, dan terdapat 5 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.

x100%

x100%

Akurasi *=* 83.33%

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

x100%

x100%

*Akurasi =* 16.67% 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%.

1. Pengaruh tingkat kemiringan wajah terhadap keberhasilan dan akurasi pengenalan wajah yang sudah tersimpan di *database*. Tabel 4.7, 4.8, 4.9, 4.10, 4.11 merupakan hasil dari pengujian terhadap kemiringan wajah untuk pengenalan wajah.

*Tabel 4.7 Hasil Pengujian Sampel 1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Drajat kemiringan wajah | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 2.1.1 | Tegak lurus | Screenshot_82.jpg | ✓ |  |
| 2.1.2 | 10°  Ke kanan | Screenshot_93.jpg | ✓ |  |
| 2.1.3 | 10°  Ke kiri | Screenshot_95.jpg | ✓ |  |
| 2.1.4 | 10°  Ke atas | Screenshot_92.jpg | ✓ |  |
| 2.1.5 | 40°  Ke atas | Screenshot_96.jpg |  | ✓ |

*Tabel 4.8 Hasil Pengujian Sampel 2*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Drajat kemiringan wajah | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 2.2.1 | Tegak lurus | Screenshot_33.jpg | ✓ |  |
| 2.2.2 | 10°  Ke kanan | Screenshot_38.jpg | ✓ |  |
| 2.2.3 | 10°  Ke kiri | Screenshot_37.jpg | ✓ |  |
| 2.2.4 | 10°  Ke atas | Screenshot_39.jpg | ✓ |  |
| 2.2.5 | 40°  Ke atas | Screenshot_28.jpg |  | ✓ |

*Tabel 4.9 Hasil Pengujian Sampel 3*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Drajat kemiringan wajah | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 2.3.1 | Tegak lurus | Screenshot_47.jpg | ✓ |  |
| 2.3.2 | 10°  Ke kanan | Screenshot_52.jpg | ✓ |  |
| 2.3.3 | 10°  Ke kiri | Screenshot_51.jpg | ✓ |  |
| 2.3.4 | 10°  Ke atas | Screenshot_53.jpg | ✓ |  |
| 2.3.5 | 40°  Ke atas | Screenshot_29.jpg |  | ✓ |

*Tabel 4.10 Hasil Pengujian Sampel 4*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Drajat kemiringan wajah | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 2.4.1 | Tegak lurus | Screenshot_44.jpg | ✓ |  |
| 2.4.2 | 10°  Ke kanan | Screenshot_45.jpg | ✓ |  |
| 2.4.3 | 10°  Ke kiri | Screenshot_57.jpg | ✓ |  |
| 2.4.4 | 10°  Ke atas | Screenshot_58.jpg | ✓ |  |
| 2.4.5 | 40°  Ke atas | Screenshot_59.jpg |  | ✓ |

*Tabel 4.11 Hasil Pengujian Sampel 5*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Drajat kemiringan wajah | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 2.5.1 | Tegak lurus | Screenshot_72.jpg | ✓ |  |
| 2.5.2 | 10°  Ke kanan | Screenshot_77.jpg | ✓ |  |
| 2.5.3 | 10°  Ke kiri | Screenshot_78.jpg | ✓ |  |
| 2.5.4 | 10°  Ke atas | Screenshot_79.jpg | ✓ |  |
| 2.5.5 | 40°  Ke atas | Screenshot_80.jpg |  | ✓ |

*Tabel 4.12 Hasil Pengujian Sampel 6*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Drajat kemiringan wajah | Hasil | Mengenal Wajah | |
| Ya | Tidak |
| 2.6.1 | Tegak lurus | Screenshot_63 | ✓ |  |
| 2.6.2 | 10°  Ke kanan | Screenshot_67 | ✓ |  |
| 2.6.3 | 10°  Ke kiri | Screenshot_68 | ✓ |  |
| 2.6.4 | 10°  Ke atas | Screenshot_69 | ✓ |  |
| 2.6.5 | 40°  Ke atas | Screenshot_70 |  | ✓ |

Pada Tabel 4.7, 4.8, 4.9, 4.10, 4.11 menunjukakan 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 4.7, 4.8, 4.9, 4.10, 4.11 di atas yaitu, dapat disimpulkan bahwa dari 5 sampel data wajah orang yang berbeda terdapat 20 pengujian yang dapat dikenali oleh sistem dengan benar, dan terdapat 5 pengujian gambar yang tidak dapat terdeteksi.

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

x100%

x100%

*Akurasi =* 80.00%

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

x100%

x100%

*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.sv 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

