# BAB III
# METODOLOGI PENELETIAN

## 3.1 ALAT YANG DIGUNAKAN

### 3.1.1 DATA SET

Data set yang digunakan sejumlah 5 subjek, dengan masing-masing subjek memiliki 7 data gambar wajah yang berbeda beda, dikarenakan pada saat tunanetra bertemu dengan orang, wajah yang teridentifikasi oleh kamera tidak hanya dari wajah depan saja melainkan dari sisi objek wajah yang lain dengan ekspresi wajah yang berbeda beda. Adapun ketentuan data set sebagai berikut:

Tabel 3. 1 Model karakteristik wajah

| No | Karakteristik wajah |
|---|---|
| 1 | Ekspresi wajah datar |
| 2 | Ekspresi wajah senyum |
| 3 | Ekspresi wajah marah |
| 4 | Ekspresi wajah mulut terbuka |
| 5 | Wajah menghadap kiri |
| 6 | Wajah menghadap kanan |
| 7 | Wajah dengan mata tertutup |

### 3.1.2 PERANGKAT LUNAK (*Software*)

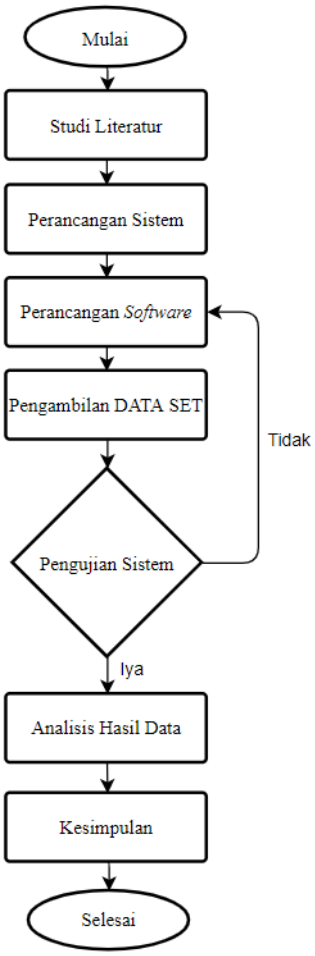
Software yang digunakan dalam perancangan ini yaitu *Pycharm* dan Arduino IDE. *Pycharm* banyak digunakan dalam penelitian di bidang pengembangan sistem, desain sistem, pembuatan model, *machine learning*, dan lain-lain. Sedangkan untuk *output* suara hasil identifikasi menggunakan Arduino IDE.

### 3.1.3 PERANGKAT KERAS (*Hardware*)

Perangkat yang digunakan dalam proses penelitian ini yaitu sebuah PC (*Personal Computer*). Spesifikasi lengkap sebagai berikut:

1. *Intel(R) core (TM) i5-9300H CPU 2.40 Ghz*
2. *Windows 11 (64-Bit)*
3. VGA NVDIA GTX 1650 4GB
4. RAM 16GB
5. SSD 512GB
6. Kamera webcam Nemesis *Full HD* 1920x1080 30fps

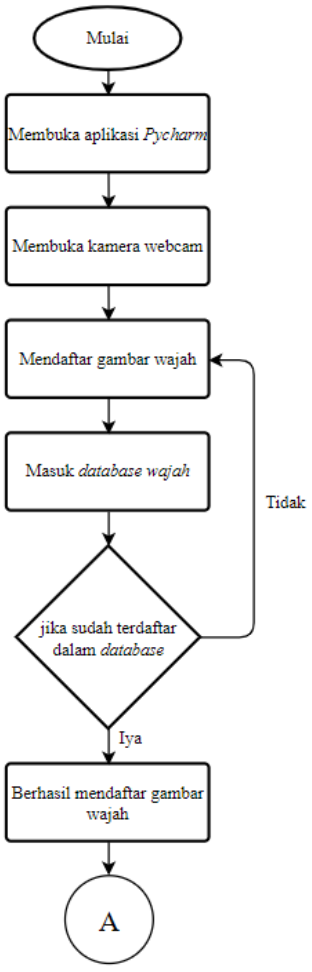
## 3.2 ALUR PENELITIAN




Gambar 3. 1 *Flowchart* Penelitian

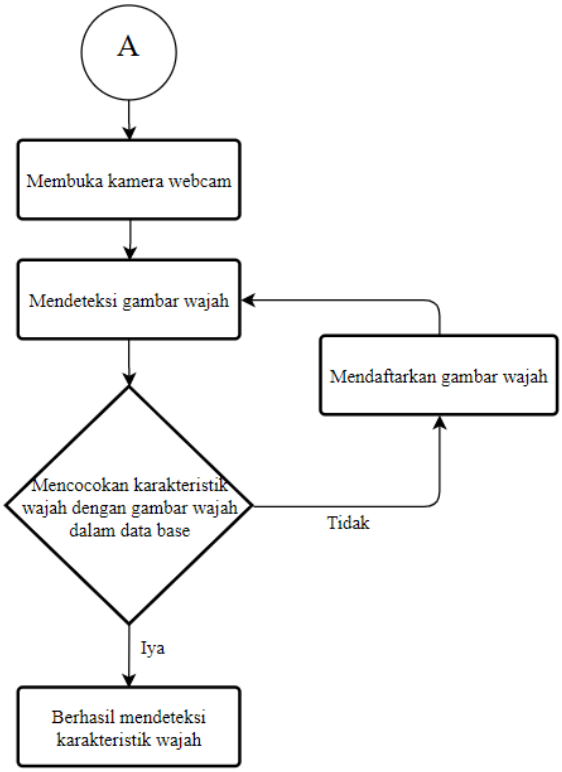
Penelitian ini akan melewati beberapa tahapan yaitu, yang pertama adalah mencari dan mengumpulkan beberapa refrensi dari jurnal serta metode yang akan digunakan pada penelitian ini, langkah selanjutnya melakukan perancangan pada *system* menggunakan *face recognition*. Selanjutnya membuat perancangan *software* menggunakan *Pycharm*. Langkah selanjutnya dilakukan pengambilan DATA SET berupa wajah manusia sebagai objek untuk melakukan identifikasi dan dimasukan pada *database*. selanjutnya dilakukan pengujian sistem keakuratan untuk mencocokan pemilik wajah asli dengan *database* yang sebelumnya telah ter-*input* pada sistem. lalu dilakukan analisis dengan hasil data yang didapat pada pengujian sistem *face recognition* identifikasi wajah dengan *output* yang diberikan berupa suara.

### 3.2.1 RANCANGAN SISTEM




Gambar 3. 2 Alur Sistem Perancangan Pendaftaran Wajah

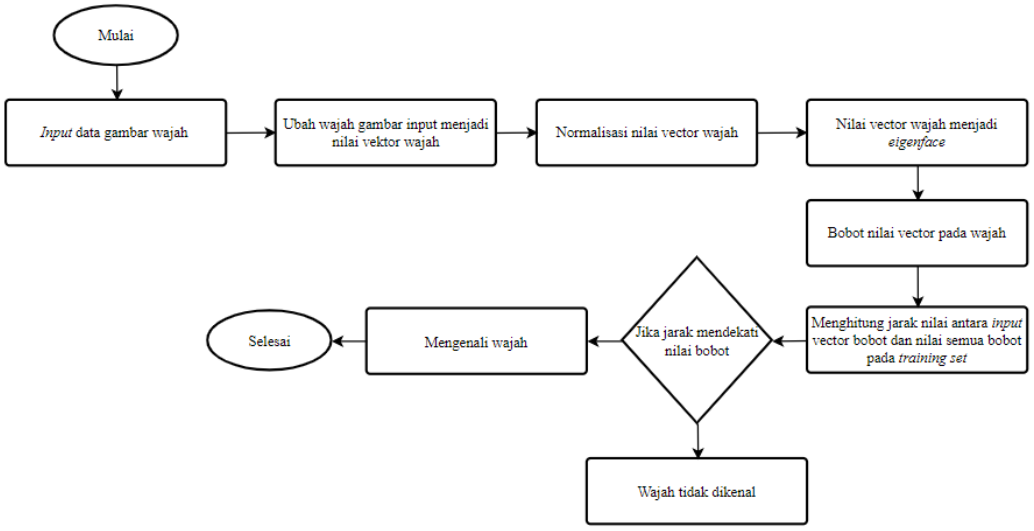
Pada gambar 3.2 merupakan alur sistem perancangan *software* untuk meng-*input* wajah seseorang pada *database*. Pada proses pertama yaitu membuka aplikasi *pycharm*, pada aplikasi tersebut menggunakan *bahasa python*. Lalu selanjutnya lakukan *running* pada program untuk membuka kamera dan lakukan *input* gambar wajah. Setelah itu wajah berhasil ter-*input* dalam *database*.




Gambar 3. 3 Alur Sistem Pengujian Identifikasi Wajah

Selanjutnya pada gambar 3.3 yaitu alur sistem pengujian identifikasi pada wajah. Dimana wajah yang sebelumnya telah ter-input dalam *database* diuji kecocokan nya dengan wajah secara *real-time* menggunakan *eigenface*. Jika sistem dapat melakukan kecocokan pemilik wajah dengan data yang sebelumnya di *input* dalam *database*, dengan begitu sistem dapat mengidentifikasi karakteristik pada wajah.

### 3.2.1.1 KARAKTERISTIK CITRA *EIGENFACE*




Gambar 3. 4 Alur Sistem *Eigenface*

Tujuan utama dari metode *eigenface* adalah untuk mendapatkan karakteristik citra dengan menggunakan karakteristik wajah, tetapi dengan menggunakan rumus transformasi matematika[15].

1. Tahapan *training*
   - Membuat database citra wajah Misal suatu citra Γ didefinisikan sebagai sebuah matriks berukuran *Nx* x *Ny* dikonversikan ke *vector* citra Γ dengan ukuran N x 1 dengan N = *Nx* x *Ny*. Ini adalah matriks yang dibentuk dengan menggabungkan kolom pada citra menjadi satu.

$$\Gamma = [\Gamma_1, \Gamma_2, \Gamma_3 \dots, \Gamma_{Mt}] \quad (1)$$

Keterangan:

Γ: citra training

Mt: jumlah citra training

   - Menghitung mean dari citra wajah

$$\Psi = \frac{1}{Mt} \sum_{i=1}^{Mt} \Gamma_i \quad (2)$$

Keterangan:

Ψ: mean citra training

Mt: jumlah citra training

Γi: citra training

Matriks mean ini berukuran (N x 1) piksel

- Menghitung perbedaan citra dengan tiap citra pada *database*.

$$\Phi = \Gamma - \Psi \quad (3)$$

Keterangan:

Φ: *zero mean*

Γ: set citra *training*

Ψ: *mean* citra *training*

$$A = [\Phi_1, \Phi_2, \Phi_3 \ldots, \Phi_{Mt}] \quad (4)$$

Keterangan:

*A*: matriks *zero mean* (*mean subtracted image*)

*Vector* dari *mean subtracted* image ini memiliki matriks berukuran *N* x *Mt pixel*. Sedangkan matriks A, masing-masing memiliki ukuran (*N* x *Mt*) *pixel*.

- Menghitung matriks kovarian.

$$P = A = \sum_{i=0}^{Mt} \Phi_i \, \Phi_i^T \quad (5)$$

Keterangan:

*P*: Matriks kovarian (berukuran *N* x *N*)

2. Tahap Reduksi

Tahap reduksi dilakukan karena ukuran matriks *N x N* terlalu besar. Perhitungan dengan menggunakan matriks *N x N* dapat memperberat kinerja CPU dan pada beberapa kasus dapat menghentikan program. Karena itu, perhitungan matriks kovarian dilakukan melalui matriks *Mt x Mt*.

3. Tahap Klasifikasi

- Proyeksi

Proses klasifikasi tidak menggunakan semua *eigenface* dari citra *training* (*Mt*), melainkan hanya menggunakan *eigenface* yang signifikan (M') saja. Langkah selanjutnya citra *training* kemudian diproyeksikan pada ruang *eigenface*, dan ditentukan bobot (*weight*) dari setiap *eigenvector*-

nya. Bobot ini merupakan *dot product* dari setiap citra dengan eigenvector-nya.

$$\omega_k = v_k^T . (\Gamma - \Psi) \quad (6)$$

Keterangan:

ω: bobot (*weight*)

v: *eigenvector*

k: 1, 2, 3, ... M'

Γ: citra *training*

Ψ: *zero mean*

Proyeksi di atas adalah proyeksi citra *training* untuk setiap *eigenvector* dengan *k* = 1, 2, 3, ... M

- Matrix Bobot

  Pada tahap ini citra dibentuk dari matriks-matriks bobot pada ruang *eigenface*. Setiap citra direpresentasikan berasal dari citra yang berukuran *Nx x Ny* pada ruang citra. Setelah dilakukan pemrosesan maka citra tersebut diwakili oleh suatu *vector* berukuran (*N'* x 1) pada ruang *eigenface*.

$$\Omega = [\omega_1, \omega_2, \omega_3 \ldots, \omega_{Mt}]^T \quad (7)$$

  Keterangan:

  Ω adalah representasi citra *training* dalam ruang *eigenface* yang berukuran *M'* x 1

4. Proyek Citra *Testing*

   Proses klasifikasi dilakukan dengan cara memproyeksikan citra baru ke dalam ruang *eigenface*. *Vector* citra *testing*, $\Gamma_T$ (*N* x 1) dikurangi dengan *mean* dari citra *training*. Matriks in berukuran *N* x 1.

$$\Phi_T = \Gamma_T - \Psi \quad (8)$$

Keterangan:

$\Phi_T$: *zero mean* citra *testing*

$\Gamma_T$: citra *testing*

Ψ: *mean* citra *training*

Kemudian dilakukan proyeksi terhadap citra *testing*. Matriks ini adalah matriks dari selisih *vector* citra *testing* dan *mean* dari citra *training* yang berukuran *N* x 1.

$$\omega_{Tk} = v_k^T . \Phi_T = v_k^T . (\Gamma_T - \Psi) \quad (9)$$

Keterangan:

$\omega_{Tk}$: bobot citra *testing*

$\Phi_T$: *zero mean* citra *testing*

$\Gamma_T$: citra *testing*

$\Psi$: *mean* citra *training*

*k*: 1, 2, 3, ... M'

Kemudian mencari matriks bobot dari citra *testing*. Matriks ini merupakan representasi dari citra *testing* pada ruang *eigenface* dengan ukuran *M'* x 1.

$$\Omega_T = [\omega_{T1}, \omega_{T2}, \omega_{T3} \dots, \omega_{TMt}]^T \quad (10)$$

Keterangan:

$\Omega_T$: bobot set citra *testing*

$\omega_T$: bobot citra *testing*

5. *Euclidean Distance*

Untuk memperoleh tingkat kesamaan antara dua citra wajah, digunakan pengukuran jarak dengan metode *Euclidean Distance*.

$$\delta_T = \|x - y\|^2 = \sum_{i=1}^{m} (x_i - x_i)^2 \quad (11)$$

Keterangan:

x: citra *training* (*x1, x2, x3, ..., xm*)

y: citra *testing* (*y1, y2, y3, ..., ym*)

### 3.2.1.2 PENGUJIAN SISTEM *FACE RECOGNITION SMART STICK*

Pengukuran pada penelitian ini sangat penting dalam melakukan pengujian pada sistem. Pengukuran membutuhkan pengujian agar alat yang digunakan sesuai dengan hasil kinerja alat dan sistem tersebut. Adapun poin-poin untuk melakukan pengukuran sistem *eigenface*:

1. Selektivitas

   Alat dapat membedakan satu objek wajah dengan objek yang lainnya. Pada penelitian ini menggunakan 3 wajah yang berbeda. setiap wajah memiliki karakteristik dan nilai *mean* vector yang berbeda.
2. Sensitivitas

   Pada penelitian ini menggunakan 7 ekspresi wajah yaitu: wajah datar, tersenyum, marah, membuka mulut, serong kanan, serong kiri, dan mata tertutup.
3. Akurasi

   Kemampuan alat dalam mendeteksi identifikasi wajah dari satu objek dengan objek lain nya. Jika wajah yang teridentifikasi benar maka akan muncul teks pada bagian wajah yang terdeteksi. Pada sistem ini objek sudah di *train* atau dikenali oleh sistem.
4. *Working Range*

   Pada penelitian ini menggunakan berbagai variasi jarak yaitu: 10cm, 30cm, 60cm, 90cm, 120cm, 150cm. Untuk melakukan identifikasi alat memiliki jarak ±1meter dari objek tersebut dengan kondisi cahaya normal.
5. Kehandalan

   Alat yang digunakan sesuai dengan prosedur yang diuji. Pada parameter ini akan di investigasi apa saja yang dapat mempengaruhi tingkat identifikasi wajah tersebut.

Sistem ini memiliki beberapa parameter yang mempengaruhi hasil deteksi wajah. Parameter terdiri menjadi 2 bagian yaitu, *resizing* dan parameter *eigenface*.

1. *Resizing*

   Merupakan proses mengubah ukuran besar citra dalam satuan *pixel* tertentu, tahapan *resizing* dilakukan dengan tujuan untuk menyesuaikan ukuran citra latih dan citra uji. Perubahan *pixel* ini dapat menghasilkan citra yang lebih besar maupun lebih kecil dari citra aslinya. Pada pengaplikasian, *resize* berpengaruh terhadap hasil deteksi wajah dikarenakan *pixel* yang terlalu kecil akan sulit di deteksi oleh sistem, sebaliknya jika pixel terlalu besar maka proses sistem akan jauh lebih lambat. *Pixel* yang biasa digunakan pada penelitian 16x16, 32x32, 64x64.

2. Parameter *eigenface*

   Terdapat 2 parameter penting dalam *eigenface* seperti *scale factor* dan *min neighbors*. *Scale factor* ialah proses dimana jendela pencarian diskalakan diantara pendeteksian berikutnya, misalnya, 1,1 berarti meningkatkan jendela sebesar 10%. Semakin kecil skala maka sistem pendeteksi akan semakin lambat dan mengalami *drop framerate persecond* (fps). Sedangkan *min neighbors* adalah parameter yang menentukan beberapa banyak tetangga yang harus dimiliki sehingga akan mempengaruhi kualitas wajah yang terdeteksi. Nilai yang lebih tinggi menghasilkan lebih sedikit deteksi tetapi dengan kualitas yang lebih tinggi.

Adapun skenario penelitian ini parameter sebagai berikut:

Tabel 3. 2 Parameter Resolusi

| *Image resize* | *Scale factor* | *Neighbors* |
|---|---|---|
| 16x16 | 20% | 5 |
| 32x32 | 20% | 5 |
| 64x64 | 20% | 5 |