BAB III

METODE PENELITIAN

3.1. Model Pengembangan

Tujuan dari tugas akhir ini akan membangun sebuah aplikasi pengenalan wajah yang dapat melakukan pencarian pada orang yang dicari. Proses pengolahan data mulai dari pengolahan citra hingga menghasilkan gambar wajah tiap orang. Gambar diolah menjadi bentuk digital atau angka-angka (menjadi matriks) dengan mengambil nilai tiap pikselnya. Lalu gambar dikonversi ke dalam bentuk digital yang lebih sederhana menggunakan metode *principle component analysis* (PCA) yaitu sebuah algoritma yang menghasilkan karakteristik dominan dari data, sehingga mewakili struktur pola citra tersebut. Kemudian dilakukan percobaan pada proses *recognition* dengan membandingkan antara data dari *database* dengan data baru yang diperoleh dari proses *recognition* tadi.

3.2. Prosedur Penelitian

Prosedur penelitian yang dipakai dalam pengerjaan tugas akhir ini adalah:

1. Studi literatur

Pencarian data-data literatur dari masing-masing fungsi pada *library* EmguCV dan OpenRobotinoAPI, melalui pencarian dari internet, dan konsep-konsep teoritis dari buku-buku penunjang serta metode yang digunakan untuk melakukan pengolahan citra.

2. Tahap perancangan dan pengembangan sistem

Dalam membuat pengembangan sistem, terdapat beberapa langkah

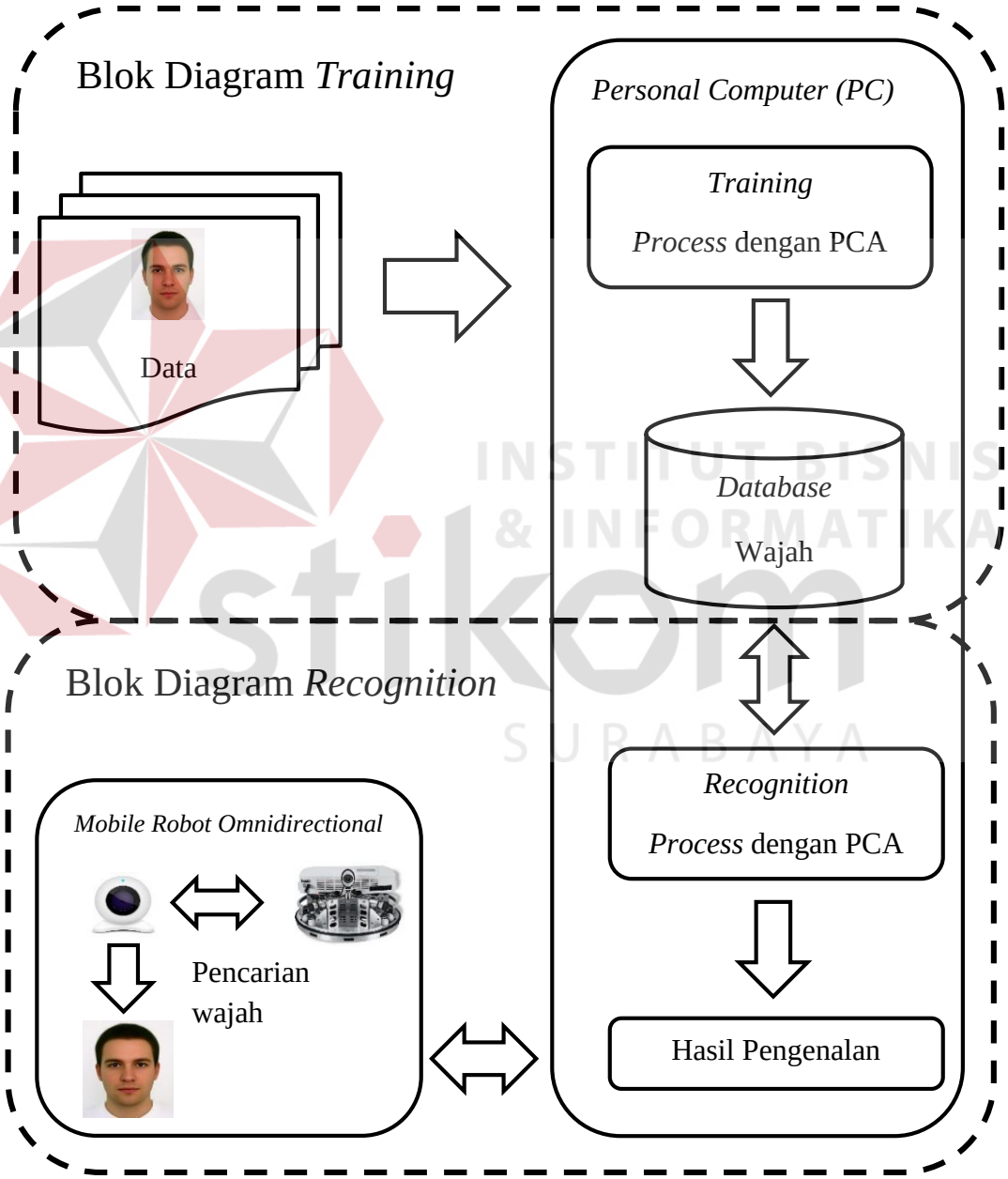
rancangan sistem yang diambil antara lain:

a. Membuat *flowchart* pada proses sistem secara keseluruhan, proses
b. *training*, proses pengolahan citra, proses PCA, dan proses *recognition.*
c. Membuat *interface* (tampilan aplikasi).
d. Mengatur fungsi-fungsi yang digunakan pada aplikasi ini dan mengelompokkan fungsi tersebut pada beberapa *class*
e. Melakukan koneksi antara komputer dengan robotino.
f. Melakukan pendeteksian wajah.
g. Melakukan pengambilan sampel data, dengan meng-*capture* wajah yang terdeteksi.
h. Melakukan proses pengolahan citra, yaitu konversi ke *grayscale* dan *histogram equation.*
i. Menerapkan metode PCA pada aplikasi.
j. Mengatur bagaimana robotino akan berjalan.
k. Melakukan percobaan pada aplikasi ini untuk memastikan apakah aplikasi ini sudah dapat berjalan dengan baik atau belum.

3.3. Diagram Blok Sistem

Sistem ini terdiri dari 2 blok utama yaitu blok proses *training* dan proses *recognition*. Dimana kerja dari kedua blok proses ini dilakukan pada komputer. Blok diagram *training* terdiri dari *webcam* dan perangkat lunak yang memproses citra masukan menjadi sebuah karakteristik citra yang hasilnya akan disimpan pada sebuah *database*. Sedangkan pada blok proses *recognition* terdiri dari *webcam* yang telah terintegrasi dengan *mobile* robot (robotino), komputer akan memproses citra

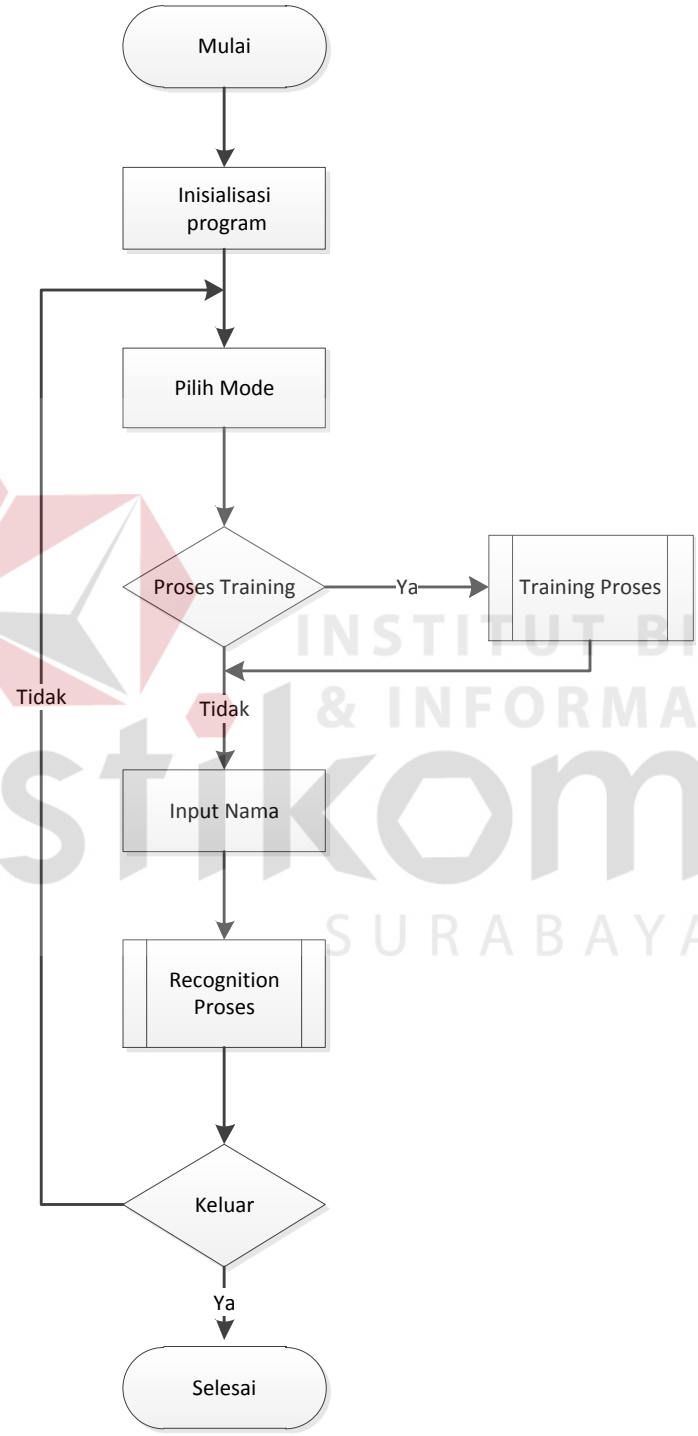
masukan menjadi karakteristik citra, lalu membandingkannya dengan karakteristik citra yang telah tersimpan pada *database* hingga memberikan *output* berupa hasil *recognition*. Untuk mempermudah dalam memahami sistem yang akan dibuat dapat dijelaskan melalui blok diagram pada Gambar 3.1.



Gambar 3.1 Blok diagram sistem secara umum

3.4. Diagram Alir Sistem

Gambar 3.2 adalah diagram alir yang akan digunakan dalam membuat algoritma program untuk mendukung sistem *recognition* wajah secara *realtime* ini:

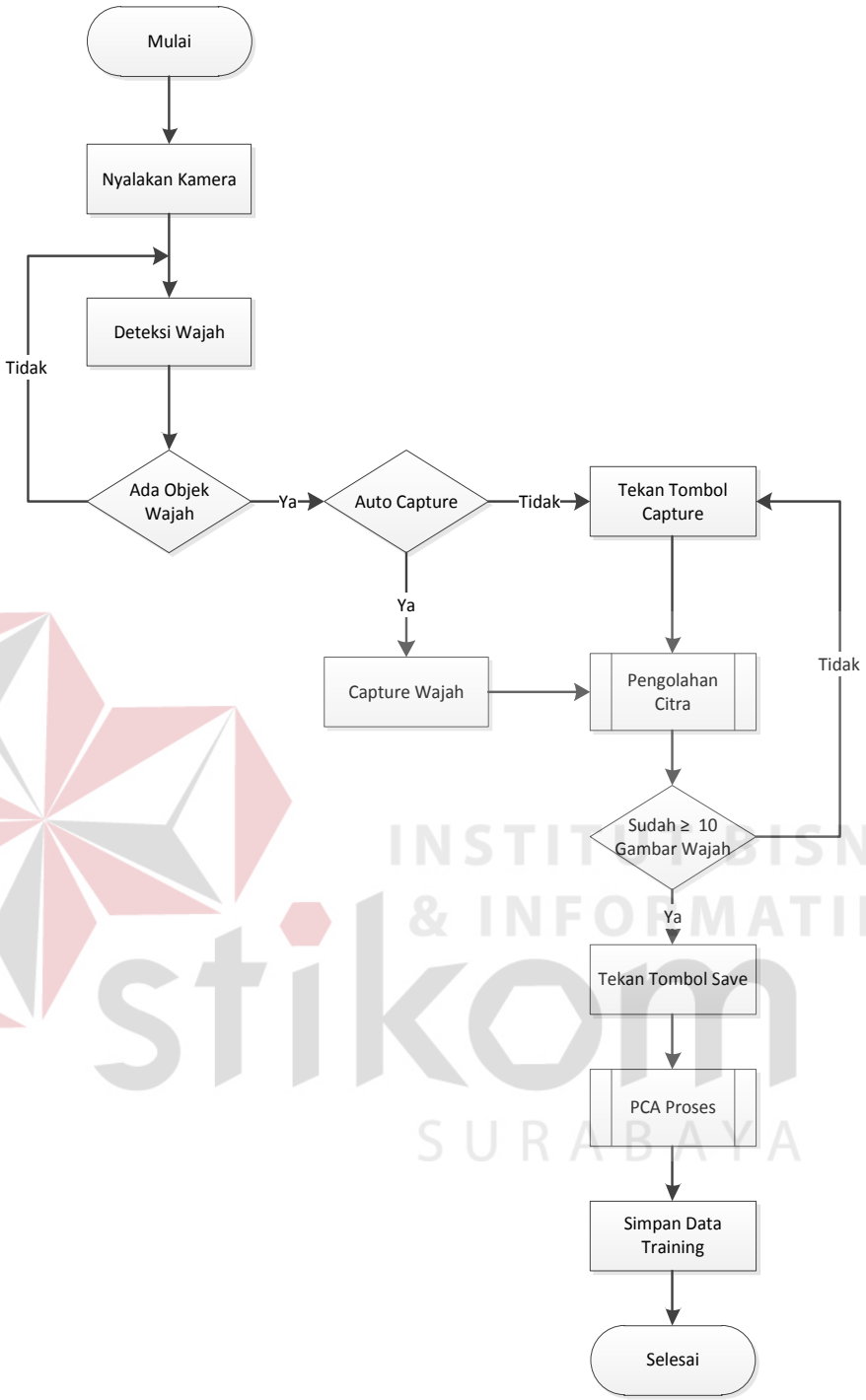


Gambar 3.2 Diagram alir sistem keseluruhan

Gambar 3.2 menunjukkan diagram alir keseluruhan sistem yang dirancang. Input awal pada sistem ini adalah pemilihan mode, jika memilih mode *training* maka akan masuk pada proses *training*, tapi jika tidak akan langsung masuk pada proses *recognition*. Proses *recogniton* akan berjalan sampai menemukan wajah dari orang yang dicari, saat orang yang dicari sudah ditemukan maka proses *recognition* akan selesai.

3.4.1. Diagram Alir Subproses *Training*

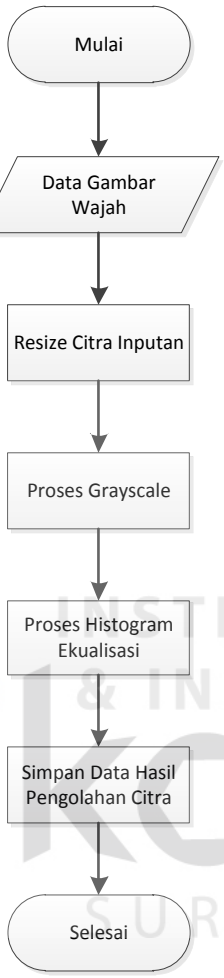
Diagram alir dari algoritma proses *training* pada sistem ini akan dijelaskan pada Gambar 3.3. Aliran proses *training* dimulai dengan mengakses *webcam* yang telah terintegrasi pada PC, lalu aplikasi akan melakukan deteksi wajah dari gambar yang didapatkan dari *webcam* tadi, saat ada wajah yang tertangkap pada gambar hasil *webcam*, lalu *user* akan menekan tombol *capture* untuk melakukan proses pengambilan gambar wajah, saat aplikasi melakukan proses *capture* proses pengolahan citra juga dilakukan, setelah didapat minimal 10 gambar wajah, proses selanjutnya dilakukan untuk mengambil ciri tiap wajah menggunakan metode PCA dan proses terakhir adalah aplikasi akan menyimpan ciri wajah tadi pada *database* yang telah ditentukan.



Gambar 3.3 Diagram alir Sistem Proses *Training*

3.4.2. Diagram Alir Pengolahan Citra

Diagram alir dari pengolahan citra yang digunakan pada sistem ini dapat dijelaskan pada Gambar 3.4

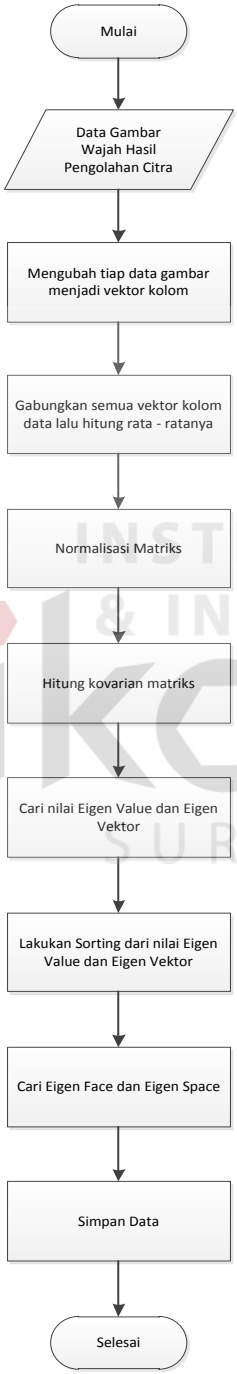


Gambar 3.4 Diagram alir pengolahan citra

Pada proses pengolahan citra sistem ini pertama setiap data gambar yang didapat pertama-tama akan diubah ukurannya menjadi 100 x 100, agar didapatkan data gambar yang seragam dengan tujuan untuk memudahkan pada proses perhitungan PCA. Setelah itu tiap data akan di ubah menjadi *grayscale*. Untuk mendapatkan data gambar yang lebih jelas dilakukan proses histogram ekualisasi. Lalu simpan semua data gambar pada direktori *database* yang telah disiapkan.

3.4.3. Diagram Alir Proses PCA

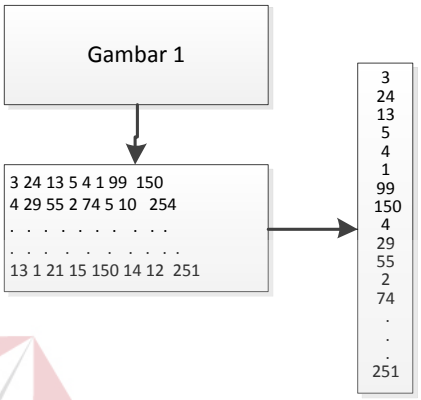
Dalam sistem ini proses yang paling utama adalah proses PCA, karena yang dapat menentukan wajah seseorang dikenali atau tidak adalah melalui proses ini. Berikut diagram alir dari proses PCA akan dijelaskan pada Gambar 3.5 :



Gambar 3.5 Diagram alir proses PCA

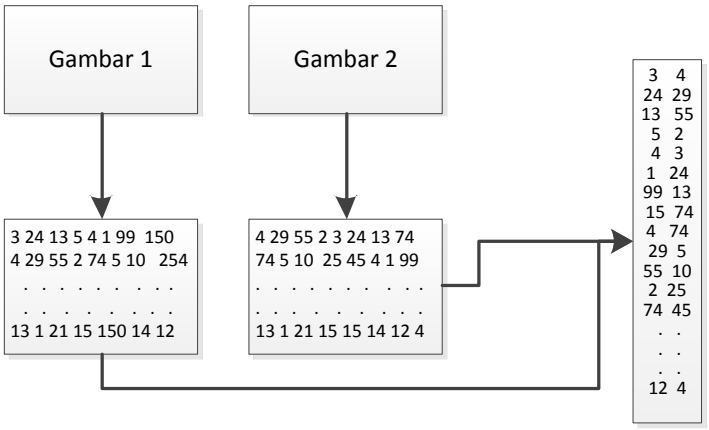
Berikut penjelasan dari diagram alir PCA:

1. Proses pembacaan matriks data gambar yang telah melalui proses pengolahan citra.
2. Proses mengubah masing-masing matriks piksel data gambar menjadi vektor kolom dengan ordo $\mathbf{N^2}$ x 1.



Gambar 3.6 Ilustrasi penyusunan matriks data menjadi vektor kolom

3. Gabungkan setiap vektor kolom dari data tersebut menjadi satu dalam sebuah matriks dengan ordo N^2 x jumlah data. Misal gambar pelatihan ada 2 gambar dengan ukuran 20 x 20 piksel maka kita akan mempunyai matriks berukuran (20 x 20) x 2 seperti pada penjelasan berikut.

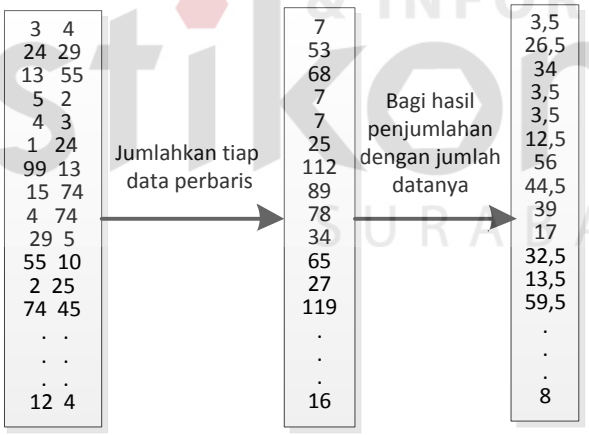


Gambar 3.7 Ilustrasi penyusunan matriks data input

Berikut adalah penggalan program untuk langkah nomer 1 sampai 3:

```
public double [,] data_trans_1D (List<Bitmap> data_inp)
        {
            Matrix_1D = new double[Maks_Dim,Jml];
            for (i = 0; i < Jml; i++)
               {
                 d=0;
                    for (j = 0; j < Width; j++)
                    {
                        for (int k = 0; k < Height; k++)
                        {
                        Matrix_1D[d,i] =
data_inp[i].GetPixel(j,k).G;
                           d++;
                        }
                    }
               }
            return Matrix_1D;
        }
```

4. Proses perhitungan rata-rata dari matrik, gabungan data *training*, dengan menambahkan semua data dalam satu baris lalu membaginya dengan jumlah data *training*.



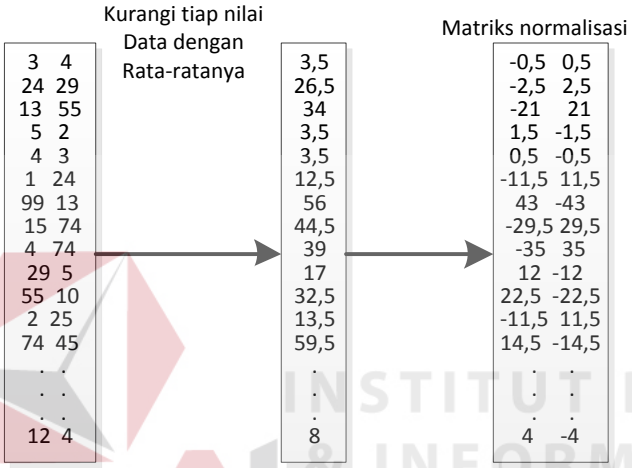
Gambar 3.8 Ilustrasi pencarian rata-rata matriks data input

Berikut ini penggalan program dari proses pencarian rata-rata pada langkah 4:

```
public double[] rata()
        {
            rata2 = new double[Maks_Dim];
            double temp=0;
            for (i = 0; i < Maks_Dim; i++)
               {
```

```
                    for (j = 0; j < Jml; j++)
                    {
                        temp += Matrix_1D[i,j];
                    }
                rata2 [i] = temp/Jml;
                temp = 0;
            }
        return rata2;
    }
```

5. Proses normalisasi matriks dengan cara mengurangkan tiap nilai pada matriks gabungan dengan nilai rata-rata.



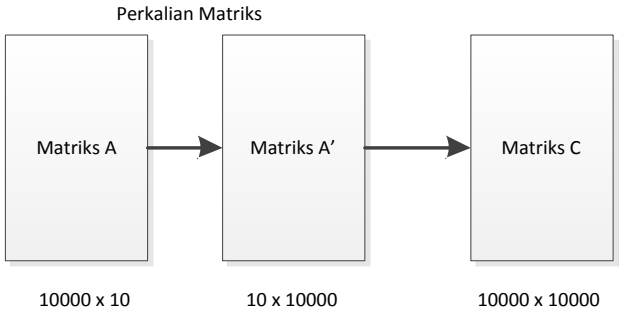
Gambar 3.9 Ilustrasi normalisasi data input

Berikut adalah penggalan program dari proses normalisasi pada langkah 5:

```
public double[,] Normalisasi()
        {
            Norm_data = new double[Maks_Dim, Jml];
            for (i = 0; i < Maks_Dim; i++)
            {
                for (j = 0; j < Jml; j++)
                {
                    Norm_data[i, j] = Matrix_1D[i, j] -
rata2[i];
                }
            }
            return Norm_data;
    }
```

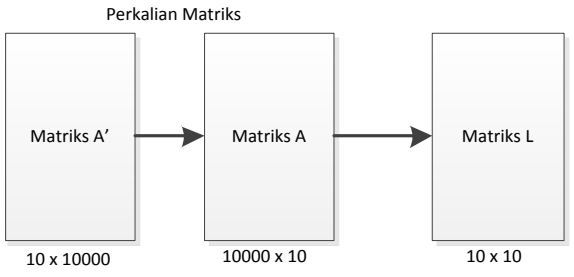
6. Proses menghitung kovarian matriks C dengan menggunakan metode PCA, karena proses perhitungan matriks C menurut persamaan 2.8 akan

menghasilkan ordo atau dimensi matriks yang sangat besar (N^2 x N^2 atau misalnya 100^2 x 100^2 dengan 100 x 100 adalah ukuran dimensi per gambar)



Gambar 3.10 Ilustrasi kovarian matriks C

Proses perkalian matriks pada Gambar 3.10 tidak efektif karena terlalu besar jumlah datanya (10000 x 10000). Melalui metode PCA, dengan menggunakan sifat perkalian matriks, maka matriks C diwakilkan dengan menggunakan matriks L dengan ordo berdasarkan jumlah citra M x M (dengan M merupakan jumlah data inputan). Perubahan dari matriks C ke matriks L sebagai penggantinya menggunakan Persamaan 2.8 dan 2.9.



Gambar 3.11 Ilustrasi kovarian matriks L

Berikut adalah penggalan program proses kovarian matriks pada langkah 6:

```
public double[,] Covarian_Mat()
        {
            Transpose = new double[Jml, Maks_Dim];
            Transpose = Norm_data.Transpose();
            Cova_mat = Transpose.Multiply(Norm_data);
            return Cova_mat;
        }
```

7. Setelah didapat nilai kovarian matriks lalu proses selanjutnya dengan melakukan perhitungan nilai eigen dan vektor eigen, dimana hasil dari proses ini berupa nilai eigen dan vektor eigen yang saling bersesuaian.
8. Melakukan pengurutan untuk nilai eigen yang didapat pada proses sebelumnya dan melakukan penentuan K *best eigen value* dan eigen vektor. Nilai K ditentukan sendiri dengan maksimum nilai adalah banyaknya data dari nilai eigen.
9. Melakukan perhitungan nilai *eigenface* dan *eigenspace* / *weight* dari hasil pemilihan K *best eigen* tadi, seperti pada persamaan 2.9 dan 2.10.

Berikut ini adalah penggalan program proses pencarian nilai eigen sampai data eigen siap untuk disimpan seperti langkah 7 sampai 9:

```
public void Eigen(out double[,] Eigen_vect, out double[]
Eigen_val)
        {
            eig = new EigenvalueDecomposition(Cova_mat);
            Eigen_val = eig.RealEigenvalues;
            Eigen_vect = eig.Eigenvectors;
            _sort(Eigen_val, Eigen_vect);
        }
public void Eig_face(double [] Eig_val ,double [,] Eig_vec,out
double[,] Eigen_face,out double[,] Eigen_space)
        {

            Eigen_face = new double[Maks_Dim, jml_Eface];
            Eigen_space = new double[Jml,jml_Eface];
            double[,] Eigen_vect_subMatrix = new double[Jml,
jml_Eface];
            Eigen_vect_subMatrix = Eig_vec.Submatrix(0, Jml -
1, 0, jml_Eface - 1);
            double norm = 0, temp = 0, temp2 = 0;
            for (i = 0; i < jml_Eface; i++) // nilai
eigenface(eigenvector C)
            {
```

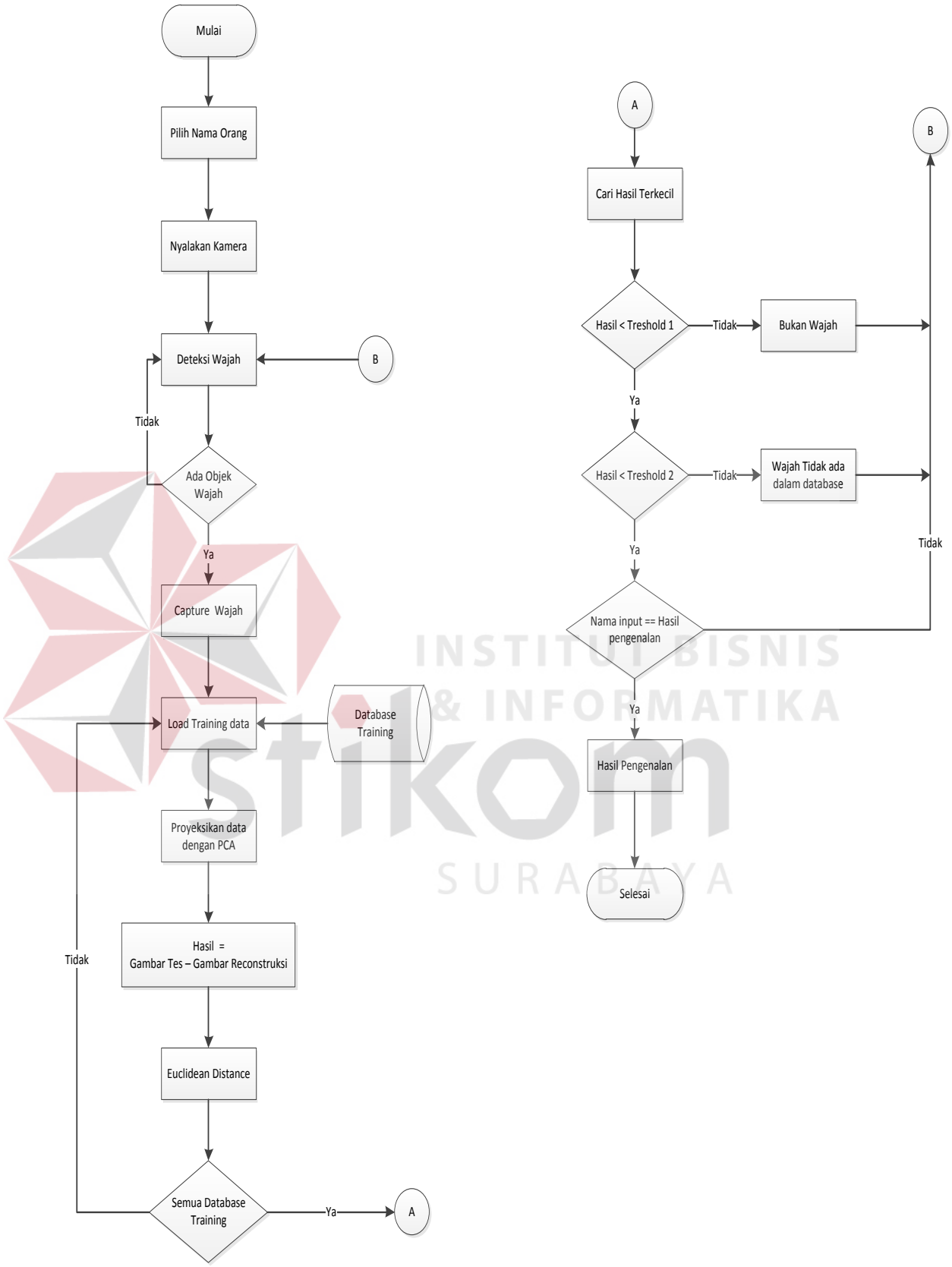
```
                    for (j = 0; j < Maks_Dim; j++)
                    {
                        for (k = 0; k < Jml; k++)
                        {
                            temp2 += Norm_data[j, k] *
Eigen_vect_subMatrix[k, i];
                        }
                        Eigen_face[j,i] = temp2;
                        temp2 = 0;
                        temp += Eigen_face[j, i] * Eigen_face[j,
i];
                    }
                    norm = Math.Sqrt(temp);
                    for ( j = 0; j < Maks_Dim; j++)
                    {
                        Eigen_face[j, i] /= norm;
                    }
                }

                for (i = 0; i < Jml; i++)
                {
                    double[] diffImage = new double[Maks_Dim];
                    double[] weight = new double[jml_Eface];
                    for (j = 0; j < Maks_Dim; j++)
                    {
                        diffImage[j] = Norm_data[j, i];
                    }
                    //weight
                    for (int l = 0; l < jml_Eface; l++)
                    {
                        for (int m = 0; m < Maks_Dim; m++)
                        {
                            weight[l] += diffImage[m] *
Eigen_face[m, l];
                        }
                    }
                }
            }
```

10. Proses terakhir menyimpan data yang didapat dari semua proses tadi pada sebuah direktori untuk digunakan sebagai *database* pada proses *recognition* nanti.

3.4.4. Diagram Alir Subproses *Recognition*

Diagram alir dari proses *recognition* pada sistem ini dijelaskan pada Gambar 3.12.



Gambar 3.12 Diagram alir proses *recognition*

Dari diagram alir proses *recognition* dapat dijelaskan bahwa langkah-langkah *recognition* adalah sebagai berikut:

1. Sistem akan menampilkan nama-nama yang terdapat pada *database*, lalu *user* akan memilih nama orang yang akan dicari.
2. Setelah itu sistem ini akan berjalan untuk memulai pencarian, dengan mulai mengakses kamera. Lalu sistem akan melakukan deteksi wajah.
3. Jika ada wajah yang terdeteksi sistem akan langsung melakukan *capture*.
4. Mengambil data yang telah tersimpan pada *database* yang didapat dari proses *training*.
5. Proses selanjutnya adalah memproyeksikan gambar yang didapat dengan metode PCA, dengan data yang telah diambil dari *database*. Berikut langkah yang dilakukan untuk memproyeksikan data gambar dengan PCA:
   a. Mengubah dimensi gambar, menjadi 1 dimensi (vektor kolom N^2 x 1), lalu mengurangkan nilai yang didapat dengan rata-rata dari data orang yang terdapat pada *database*.
   b. Menghitung *eigenspace* dari gambar baru yang didapat, dengan mengalikan data 1 dimensi dengan *eigenface* dari proses *training*.
   c. Merekonstruksi data yang baru didapat dari proses tadi, lalu membandingkannya dengan data 1 dimensi. Hasilnya akan dihitung lagi menggunakan proses *euclidean distance*, seperti pada persamaan 2.15.
   d. Lakukan proses diatas sampai semua data pada *database training* selesai digunakan.

6. Cari data yang memiliki nilai terkecil, lalu bandingkan dengan nilai *threshold* 1 yang didapatkan dari hasil melakukan percobaan beberapa kali pada sebuah objek pengenalan.
7. Jika hasil lebih kecil dari *threshold* 1, berarti objek yang ditentukan adalah wajah. Jika nilai terkecil yang terdeteksi lebih dari tershold 1 maka sistem akan mendeteksi objek tersebut sebagai objek lain dan bukan wajah, lalu sistem akan mencari objek baru dengan kembali melakukan deteksi wajah.
8. Lalu proses selanjutnya adalah melakukan pengecekan apakah nilai terkecil tadi kurang dari nilai dari *threshold* 2. Jika nilai lebih kecil dari *threshold* 2 maka wajah tersebut merupakan wajah yang ada dalam *database*, tapi jika nilai lebih dari *threshold* 2 maka objek tersebut adalah wajah yang tidak terdapat pada *database* yang sudah ada.
9. Setelah itu proses selanjutnya adalah mencocokkan hasil nama dari objek yang terdeteksi tadi dengan nama yang diinputkan oleh *user* pada awal proses. Jika hasilnya sama, maka sistem akan menampilkan data dari orang yang terdeteksi itu tadi.
10. Hal ini dilakukan sampai didapatkan orang yang sesuai dengan nama yang dicari oleh *user* pada awal proses *recognition* ini.

3.5. Inisialisasi Robot

Tahap-tahap inisialisai Robotino meliputi cara-cara *setting* koneksi Robotino, pergerakan Robotino, penerimaan data citra serta *streaming* citra Robotino. Untuk kendali Robotino digunakan OpenRobotinoAPI (*Application Programming Interface*) yaitu *library* aplikasi *programming* yang dibuat khusus

untuk Robotino, yang diciptakan untuk mempermudah *user* dalam membuat program pada Robotino. *Library* ini memungkinkan akses penuh terhadap sensor dan *actuator* pada Robotino. Komunikasi antara Robotino dengan PC melalui jaringan TCP atau UDP menggunakan media *wireless.* Untuk mengakses OpenRobotinoAPI pada program harus menggunakan *header* Robotino untuk bahasa *Microsoft Visual Csharp*, yaitu *using* rec.robotino.com dan *using* rec.robotino.com.examples.client.

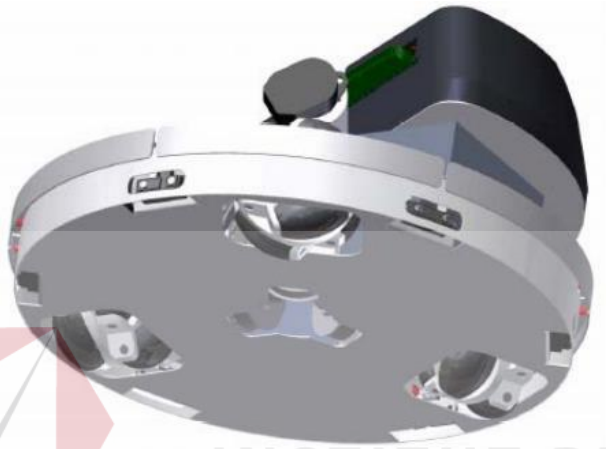
3.5.1. Koneksi Robotino

Untuk menghubungkan komputer dengan Robotino pada sistem ini dengan memanfaatkan media *wireless*. Komputer akan melakukan koneksi dengan *access point* yang sudah terintegrasi pada Robotino. Pada *prototipe* fungsi com.setAddress(hostname) yang akan digunakan untuk memberikan alamat IP yang akan diakses. Untuk simulasi pada Robotino SIM menggunakan alamat IP 128.0.0.1:8080, sedang untuk koneksi langsung pada *access point* Robotino, secara *default* alamat IP yang digunakan adalah 172.16.1.1.

Kemudian digunakan prototipe fungsi com.connect(blockUntilConnected) dimana *function blocks* ini digunakan untuk menghubungkan koneksi dari PC ke *access point* Robotino atau simulasi, dan akan bekerja apabila koneksi sudah *established* atau terjadi *error*.

3.5.2. Pergerakan Robotino

Robotino memiliki sistem pergerakan *omni-directional drive* dengan menggunakan 3 roda yang terpasang, sehingga memungkinkan robot ini untuk bergerak maju, mundur, ke samping, dan ke segala arah serta berputar ditempat. Berikut gambar sistem *omni-directional drive* Robotino pada Gambar 3.13.



Gambar 3.13 *Omni-Directional Drive* Pada Robotino

Untuk pergerakan Robotino digunakan fungsi setVelocity yang memiliki parameter vx, vy, dan omega. Parameter vx adalah parameter kecepatan pada sumbu x dan vy adalah parameter kecepatan untuk sumbu y, dengan ketentuan parameter vx dan vy dalam satuan mm/s. Dan omega merupakan parameter kecepatan sudut dengan ketentuan parameter omega dalam satuan deg/s.

3.5.3. Penerimaan Data Citra

Setiap data citra yang dikirimkan dari *webcam* Robotino diakses dengan *pointer* bertipe *const unsigned char*. Karena resolusi *default* dari Robotino adalah 320x240 maka data untuk 1 citra yang dikirimkan adalah sebanyak 230400, dimana pada 1 piksel citra terdapat 3 *channel*, dan pada setiap *channel* berukuran 8bit.

Ketika fungsi *setStreaming* bernilai *true*, fungsi *imageReceivedEvent* akan terus melakukan *streaming* citra hingga koneksi diputuskan atau saat *setStreaming* bernilai *false*. Penerimaan data citra akan ditampung pada sebuah variabel bertipe data Image<Bgr,byte>. Ketika fungsi *setStreaming* di set dengan nilai *true*, maka fungsi *imageReceivedEvent* langsung menangkap data citra secara *streaming*. Fungsi *imageReceivedEvent* perlu dideklarasikan sebagai *class* baru karena fungsi tersebut adalah fungsi *virtual*, namun untuk penulisannya harus disertakan fungsi induknya (*parent*) karena *class* yang dibuat adalah *class* turunan (*inheritance*).

berikut potongan program untuk menampilkan gambar hasil dari *webcam* pada Robotino:

1. Potongan program pada *main form*

```
robot.ImageReceived    +=    new    Robot.ImageReceivedEventHandler
    (robot_ImageReceived) ;
void robot_ImageReceived(Robot sender, Image Img)
            {
                if (this.InvokeRequired)
                {
                    this.Invoke(new MethodInvoker(Invalidate));
                }
                frame2 = new Image<Bgr, byte>(new Bitmap(Img));
            }
```

2. Program pada *class robot*

```
private class MyCamera : Camera
        {
            Robot robot;

            public MyCamera(Robot robot)
            {
                this.robot = robot;
            }

            public override void imageReceivedEvent(Image data,
    uint dataSize, uint width, uint height, uint numChannels, uint
    bitsPerChannel, uint step)
            {
                if (robot.ImageReceived != null)
                {
                    robot.ImageReceived.BeginInvoke(robot, data,
    null, null);

                }
```

```
        }
    }
```

Data citra yang ditangkap adalah data citra dengan ruang warna RGB dan disimpan langsung pada variabel *frame2* dengan tipe data Image<Bgr,byte> yaitu stuktur data untuk penyimpanan data citra pada EmguCV sebagai pengganti Iplimage pada OpenCV. Namun urutan *channel* data dalam *frame2* adalah BGR sehingga untuk menampilkan warna sesungguhnya, harus dikonversikan terlebih dahulu dengan fungsi frame2.convert<Gray,byte> seperti baris perintah seperti berikut:

```
gray_img = frame2.Convert<Gray, byte>();
```

Untuk menampilkan data citra yang sudah tersimpan pada *frame2* kedalam *window* baru digunakan prototipe fungsi pada *library* EmguCV dengan cara:

```
pictureBox_frame_proc.Image = frame2.ToBitmap();
```

Dengan proses tadi, *webcam* yang sudah terintegrasi dengan Robotino akan dapat ditampilkan pada sebuah *window*.

3.6. Perancangan Sistem

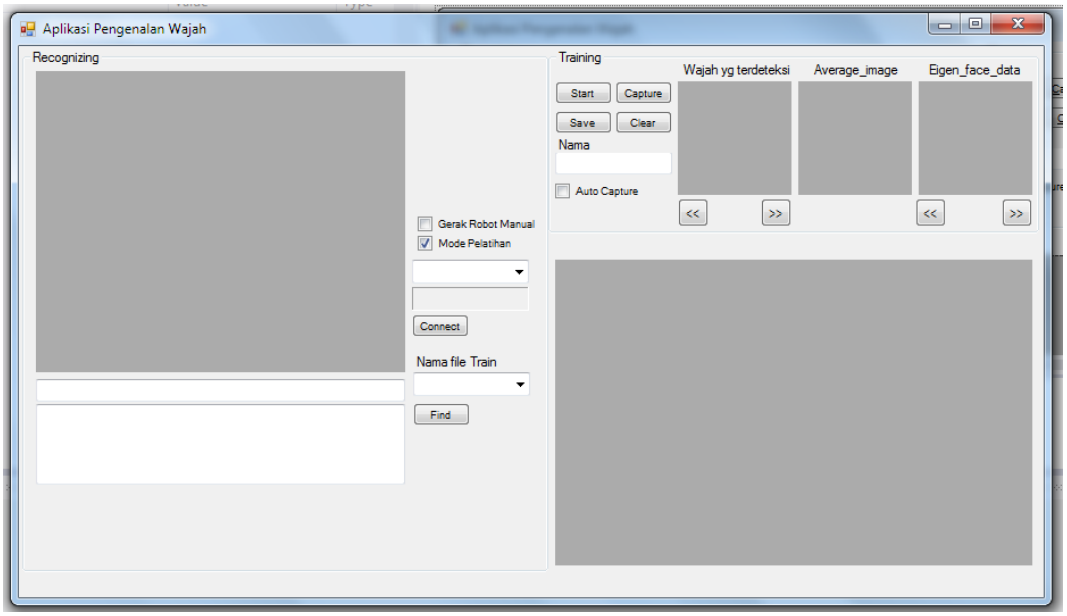
3.6.1. Konfigurasi EmguCV

EmguCV adalah *library* tambahan yang menjembatani hubungan antara *compiler Csharp* dengan *library* OpenCV. Untuk menggunakan *library* ini perlu dilakukan konfigurasi agar bisa berjalan dengan baik pada program. *Library* ini tidak terhubung secara otomatis pada program, sehingga perlu dilakukan beberapa konfigurasi dalam melakukan koneksi. Langkah-langkahnya antara lain:

1. Menginstall *library* EmguCV yang dapat bekerja pada versi *compiler* yang digunakan pada sebuah direktori yang sudah ditentukan.
2. Selanjutnya masuk ke dalam *file* direktori tempat proses instal tadi dilakukan, *copy* file yang dibutuhkan dengan ekstensi *.dll*, lalu simpan pada direktori program.
3. Lalu proses selanjutnya masuk pada program, pilih *reference* > *add reference* lalu pilih *library* mana saja yang akan digunakan pada program.
4. Proses terakhir menulis *header* dari *library* yang akan digunakan pada program. Lalu cek dengan melakukan *run* apakah ada yang masih kurang benar pada proses ini.

3.6.2. *Interface* Sistem

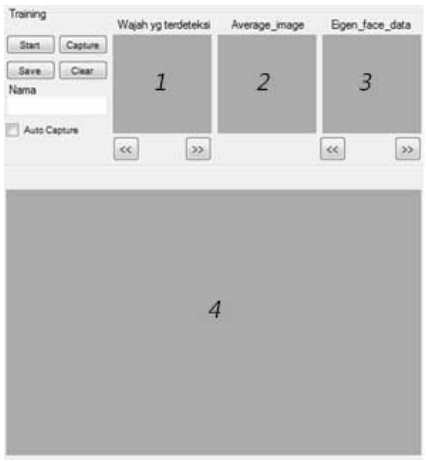
Pembuatan *interface* sangat diperlukan agar pengguna dapat melakukan interaksi dengan sistem ini. Sehingga dibutuhkan perancangan secara detil mengenai tampilan aplikasi sistem ini.



Gambar 3.14 *Interface* main program

Pada Gambar 3.14 merupakan *interface* dari program dimana semua proses yang ada pada sistem ini akan dilakukan melalui GUI (*graphical user interface*). Pada sistem ini ada beberapa proses yang dilakukan, seperti pengolahan citra, proses identifikasi wajah, proses inisialisasi kamera, dan proses perhitungan PCA. Semua proses tadi akan dikelompokkan menjadi dua proses utama, yaitu:

1. *Training*



Gambar 3.15 *Interface training* proses

Pada Gambar 3.15 dapat dilihat bahwa pada *form* ini terdapat beberapa komponen yang digunakan untuk mewakili proses apa saja yang dilakukan pada proses *training*. Dari komponen yang digunakan, tiap komponen memiliki fungsinya masing-masing, antara lain:

1. *Picturebox*:
   a. *Picturebox* 1: menampilkan gambar hasil dari proses *capture*.
   b. *Picturebox* 2: menampilkan gambar hasil dari proses perhitungan rata-rata dari satu set gambar yang didapat pada proses *capture.*
   c. *Picturebox* 3: menampilkan gambar hasil dari proses perhitungan *eigenface*.
   d. *Picturebox* 4: menampilkan gambar hasil dari kamera / *webcam* pada komputer atau laptop.
2. *Checkbox*: berfungsi untuk memilih mode *autocapture*. Saat *checkbox* ini bernilai *true* maka proses *capture* akan dilakukan sebanyak 10 kali, tanpa harus menekan *button capture* berkali-kali.
3. *Textbox*: untuk melakukan *input* nama dari orang yang melakukan *training* wajah.
4. *Button*:
   a. *Button Start* : melakukan inisialisasi kamera yang terpasang pada komputer atau laptop yang digunakan.
   b. *Button Save* : melakukan penyimpanan data pada direktori yang sudah ditentukan, sekaligus semua proses pengolahan citra dan PCA berada pada *button* ini.

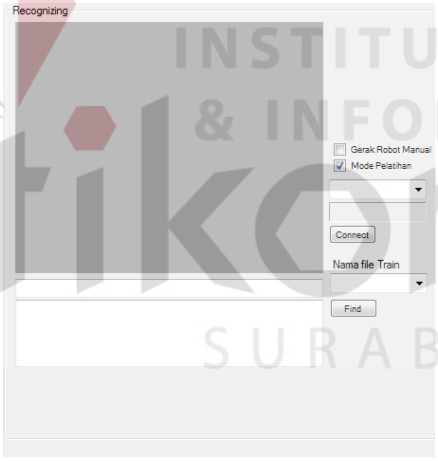
c. *Button Capture* : melakukan pengambilan gambar dari wajah yang terdeteksi.

d. *Button Clear* : menghapus data yang telah diperoleh, untuk dapat melakukan proses *training* dari awal lagi.

e. *Button next* : menampilkan data selanjutnya dari kumpulan data gambar yang didapat. Ada 2 *button* yang memiliki fungsi sama, untuk data wajah yang telah ter*capture* dan data *eigenface.*

f. *Button previous* : menampilkan data sebelumnya dari kumpulan data gambar yang didapat. Ada 2 *button* yang memiliki fungsi sama, untuk data wajah yang telah ter*capture* dan data *eigenface*.

2. *Recognition*



Gambar 3.16 *Interface recognition* proses

Pada Gambar 3.16 terdapat beberapa komponen yang digunakan pada proses *recognition*, antara lain:

1. *Picturebox* : menampilkan gambar hasil *streaming* dari kamera yang terintegrasi dengan Robotino.

2. *Checkbox* :
    a. *Checkbox* gerak robot manual : berfungsi untuk menampilkan *button* yang digunakan untuk memerintahkan robot untuk bergerak. Ada *button* maju, mundur, ke kanan, ke kiri, berputar, dan berhenti.
    b. *Checkbox* mode pelatihan : berfungsi untuk memulai proses *training* yang dijelaskan pada Gambar 3.11.
3. *Label* : menampilkan informasi koneksi antara komputer dengan Robotino.
4. *Combobox* :
    a. *Combobox* mode Robotino : menampilkan pilihan mode yang bisa dilakukan dengan Robotino. Ada 2 mode yang disediakan, mode demo yang akan menghubungkan komputer dengan Robotino SIM Demo dan mode *real* yang akan menghubungkan komputer langsung dengan *access point* pada Robotino.
    b. *Combobox* nama *file train* : menampilkan nama-nama dari orang yang sudah melakukan *training* wajah.
5. *Textbox* :
    a. *Textbox* Info file : *textbox* ini akan menampilkan informasi dari data wajah yang terdeteksi.
    b. *Textbox* status : menampilkan status apakah orang yang dicari sudah ditemukan atau belum.
6. *Button* :
    a. *Button connect* : untuk memerintahkan komputer dalam melakukan koneksi, dengan mengirimkan alamat IP yang telah di set secara *default* pada

program. Alamat IP 127.0.0.1:8080 untuk koneksi dengan Robotino SIM Demo dan IP 172.26.1.1 untuk melakukan koneksi dengan *access point* pada Robotino.

Button find : melakukan perintah pada komputer untuk mengaktifkan kamera yang terintegrasi, lalu proses *load* data dari *database* juga dilakukan. Saat proses pendeteksian dimulai dan wajah ditemukan, proses perhitungan PCA dimulai, untuk menentukan apakah wajah tersebut adalah wajah yang dicari atau bukan.

3.7. Metode Pengujian dan Evaluasi Sistem

Dalam pengujian sistem ini pengujian akan dilakukan pada Robotino dan aplikasi pada PC yang telah selesai dibuat. Pengujian yang dilakukan dimulai dari menghubungkan koneksi ke Robotino, menggerakkan Robotino, *streaming* kamera Robotino, pengambilan citra dari kamera PC, pendeteksian wajah, pengolahan citra, penerapan metode *principal component analysis*, pengujian pengenalan pada wajah dengan beberapa kondisi, dan yang terakhir adalah pengujian sistem secara keseluruhan yaitu Robotino dapat menemukan wajah yang telah ditentukan.

3.7.1. Pengujian dan Evaluasi Koneksi Robotino

Tujuan pengujian ini yaitu mengetahui apakah Robotino sudah mampu terhubung dengan PC. Pengujian ini dilakukan dengan cara menyalakan Robotino lalu melihat apakah *access point* dari Robotino telah dapat terlihat pada PC, kemudian dilakukan pengujian apakah *access point* Robotino sudah dapat

terhubung secara *wireless* dengan baik pada PC. Langkah berikutnya adalah menghubungkan koneksi dari aplikasi dengan Robotino dengan melakukan pengaturan dari alamat IP untuk Robotino yaitu 172.16.1.1. Pengujian ini dinyatakan berhasil bila aplikasi yang dibuat bisa melakukan koneksi pada *access point* Robotino. Hal ini dapat dilihat pada aplikasi, jika Robotino telah berhasil terhubung maka aplikasi akan menampilkan tulisan *"connected"*. Dengan begitu berarti Robotino telah berhasil terhubung dengan aplikasi dan siap dilakukan pengujian selanjutnya.

3.7.2. Pengujian dan Evaluasi Pergerakan Robotino

Tujuan pengujian pergerakan Robotino memanfaatkan *omni-directional drive* Robotino adalah untuk mengetahui arah pergerakan Robotino dengan mengatur parameter untuk pergerakan Robotino apakah sesuai dengan yang diinginkan. Untuk pengujian pergerakan dari Robotino dilakukan dengan cara mengubah parameter dari fungsi pada *library* yang telah disediakan oleh Festo untuk pergerakan Robotino yaitu *setVelocity*. Hasil dari pergerakan Robotino tadi akan digunakan untuk melakukan pencarian wajah yang terdeteksi oleh aplikasi untuk proses *recognition*. Pengujian ini dinyatakan berhasil apabila Robotino dapat bergerak sesuai dengan perintah dari *user*. Apabila *user* memberikan perintah berupa penekanan tombol maju, maka Robotino harus bergerak maju, jika *user* menekan tombol putar maka Robotino harus berputar. Semua pergerakan dari Robotino akan bergantung pada perintah dari *user*.

3.7.3. Pengujian dan Evaluasi *Streaming* Citra Melalui Kamera Robotino

Tujuan pengujian ini yaitu untuk mengetahui apakah aplikasi sudah mampu menampilkan data gambar dari *webcam* Robotino ke *window* pada PC, dan untuk gambar yang ditampilkan apakah sudah *streaming*. Untuk mengetahui apakah data citra dari Robotino telah dapat diakses oleh aplikasi, maka dilakukan pengujian dengan cara menjalankan aplikasi untuk melakukan inisialisasi dari Robotino. Selanjutnya pada aplikasi akan menampilkan data citra hasil dari *webcam* Robotino pada sebuah *picturebox* yang telah disiapkan. Pengujian ini dinyatakan berhasil jika hasil gambar yang ditangkap oleh kamera dari Robotino dapat ditampilkan pada *picturebox* pada aplikasi secara *realtime*. Jadi apa yang terlihat pada kamera dari Robotino akan langsung ditampilkan pada aplikasi secara terus menerus sampai ada perintah untuk berhenti menampilkan gambar.

3.7.4. Pengujian dan Evaluasi *Streaming* Citra Melalui Kamera PC

Tujuan pengujian ini yaitu untuk mengetahui apakah aplikasi sudah mampu menampilkan data citra dari kamera PC ke aplikasi pada Visual CSharp dan apakah dapat langsung diproses oleh program. Untuk mengetahui apakah data citra dari *webcam* dari PC telah dapat diakses oleh aplikasi, maka dilakukan pengujian dengan cara menjalankan aplikasi untuk melakukan inisialisasi pada kamera yang terintegrasi dengan PC. Selanjutnya pada aplikasi akan menampilkan data citra hasil dari *webcam* PC pada sebuah *picturebox* yang telah disiapkan. Pengujian ini dinyatakan berhasil jika hasil gambar yang ditangkap oleh kamera dari PC dapat ditampilkan pada *picturebox* pada aplikasi secara *realtime*. Jadi apa yang terlihat

pada kamera dari PC akan langsung ditampilkan pada aplikasi secara terus menerus sampai ada perintah untuk berhenti menampilkan gambar.

3.7.5. Pengujian dan Evaluasi Deteksi Wajah

Pengujian ini bertujuan untuk menguji apakah program sudah dapat mendeteksi wajah yang tertangkap dari hasil kamera. Pengujian deteksi wajah dilakukan pada hasil kamera yang ditampilkan pada aplikasi. Pendeteksian ini dilakukan dengan menggunakan *library* yang telah disediakan, yaitu dengan menggunakan fungsi HaarObjectDetector Object_ut. Pengujian ini akan dilakukan berdasarkan pose wajah, jarak, dan ekspresi wajah yang tertangkap oleh kamera. Pengujian ini dinyatakan berhasil jika aplikasi sudah dapat mendeteksi wajah yang tertangkap oleh kamera. Setiap wajah yang tertangkap oleh kamera dan berhasil dideteksi akan ditandai dengan sebuah kotak yang berada disekeliling wajah. Hal ini dilakukan untuk menandai area wajah yang telah berhasil dideteksi dan area kotak tersebut yang nantinya disimpan pada *database*.

3.7.6. Pengujian dan Evaluasi Pengolahan Citra

Pengujian ini bertujuan untuk mempersiapkan gambar yang dihasilkan dari proses *capture* wajah agar menjadi sebuah gambar yang siap diproses pada proses selanjutnya (PCA). Pengujian berikut adalah pengujian data citra pada hasil *capture* dari pendeteksian wajah. Pengolahan citra yang dilakukan pada proses adalah sebatas melakukan *resize* gambar wajah, konversi warna dari RGB ke *grayscale* dan proses perataan gambar dengan histogram ekualisasi. Gambar hasil dari proses

capture akan diseragamkan ukurannya, di ubah menjadi gambar wajah dengan ukuran 100x100. Lalu gambar akan di konversikan ke dalam ruang warna *grayscale* dan proses selanjutnya melakukan perataan gambar dengan proses histogram ekualisasi. Pengujian ini dinyatakan berhasil jika semua proses pengolahan citra telah berhasil dilakukan dan dihasilkan gambar *grayscale* yang jelas dan lebih merata nilai pixelnya. Gambar ini yang selanjutnya digunakan untuk melakukan proses PCA.

3.7.7. Pengujian dan Evaluasi Metode PCA pada Proses *Training*

Tujuan Pengujian ini adalah untuk melihat apakah aplikasi telah dapat mengambil nilai-nilai penting yang dibutuhkan pada metode *principal component analysis* (PCA). Pengujian pada penerapan metode *principal component analysis* dalam melakukan proses pembuatan *database* pengenalan wajah. Apakah sudah didapatkan nilai-nilai yang dibutuhkan dari proses ini, antara lain rata-rata, nilai eigen, nilai eigen vektor, nilai *eigenface* dan nilai *eigenspace*. Pengujian ini dinyatakan berhasil jika data penting seperti nilai rata-rata gambar, nilai *eigenface* dan nilai *eigenspace* dari proses *training* telah berhasil didapatkan.

3.7.8. Pengujian dan Evaluasi Metode PCA pada Proses *Recognition*

Tujuan Pengujian ini adalah untuk melihat tingkat ke akurasian dari metode *principal component analysis* dalam melakukan proses pengenalan wajah. Pengujian metode *principal component analysis* dalam melakukan proses pengenalan pada wajah meliputi pengujian terhadap ekspresi wajah, jarak wajah

terhadap kamera, dan intensitas cahaya. Pengujian ini dilakukan pada objek wajah orang yang telah melakukan *training* sebelumnya. Proses *recognition* dikatakan berhasil jika orang yang telah melakukan *training* sebelumnya telah dapat dikenali oleh aplikasi.

3.7.9. Evaluasi Sistem secara Keseluruhan

Pengujian terakhir adalah pengujian sistem secara keseluruhan dari awal hingga akhir, dimana pengujian dilakukan dengan menjalankan aplikasi secara keseluruhan. Mulai dari proses *training*, sampai proses *recognition*. Pertama-tama sistem ini akan melakukan *training* wajah pada beberapa orang. Setelah itu sistem ini akan menerima perintah dari *user* berupa sebuah inputan nama orang yang sesuai dengan nama yang ada pada *database*, lalu Robotino akan mulai berjalan untuk melakukan pencarian wajah pada tiap wajah yang terdeteksi. Robotino akan berjalan sesuai dengan posisi wajah yang terdeteksi, disini robot akan memposisikan wajah ditengah *picturebox*, jika wajah berada dipojok sebelah kiri maka Robotino akan bergerak ke kanan agar wajah yang terdeteksi berada ditengah dan begitu sebaliknya. Apabila wajah belum tepat berada ditengah dan jarak belum sesuai dengan jarak yang telah diatur maka proses *recognition* tidak akan dilakukan sampai wajah sudah berada diposisi yang telah ditentukan. Setelah wajah berada diposisi yang telah ditentukan, tiap wajah yang terdeteksi akan dicocokkan datanya dengan data yang sedang dicari, apakah sama atau tidak. Robotino akan berhenti setelah menemukan orang yang namanya sesuai dengan inputan *user* tadi.