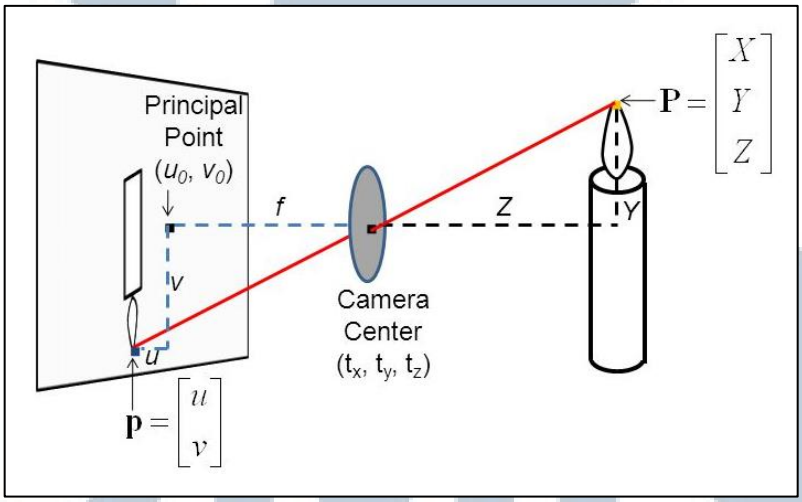
BAB II

LANDASAN TEORI

2.1 Kalibrasi Kamera

Kalibrasi kamera dilakukan untuk mendapatkan parameter intrinsik dan ektrinsik pada kamera (Rachmawati, 2013:46). Parameter intrinsik merupakan kondisi fisik yang dimiliki oleh kamera yaitu *focal length* dan titik utama (*principal point*), sedangkan parameter ekstrinsik merupakan faktor eksternal dari kamera yaitu posisi serta orientasi dari kamera tersebut.

Kamera menangkap titik-titik dari proyeksi adegan tiga dimensi yang diekspresikan sebagai $[x\ y\ z\ 1]^T$ dan mengubahnya menjadi titik-titik pada citra dua dimensi seperti pada Gambar 2.1



Gambar 2.1 Perubahan titik tiga dimensi menjadi titik dua dimensi (http://slideplayer.com/slide/5261808/, 2017)

P merupakan *world point*, sebuah titik tiga dimensi (X, Y, Z) yang di tangkap oleh kamera dan mengubahnya menjadi sebuah *image* atau citra dua dimensi yang disimbolkan sebagai p, f merupakan *focal length* yang dimiliki oleh kamera dan Z adalah kedalaman atau jarak dari lensa terhadap titik tiga dimensi.

Berikut adalah persamaan pada kamera kalibrasi yang bertujuan untuk mencari titik koordinat homogen dua dimensi yang diekspresikan sebagai $[u\ v\ w]^T$:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K[\ I_3\ \ 0_3\ ] \begin{bmatrix} R & -T \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \ldots(2.1)$$

atau

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = M \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \text{dimana } M = [\ KR \mid -KRT\ ] \quad \ldots(2.2)$$

Matriks $K[\ I_3 \mid 0_3\ ]$ merupakan parameter intrinsik yang dimiliki oleh kamera sedangkan matriks $\begin{bmatrix} R & -T \\ 0_3^T & 1 \end{bmatrix}$ merupakan parameter ekstrinsik dari kamera. Matriks K pada parameter intrinsik merupakan matriks kalibrasi kamera $K = \begin{bmatrix} fa & 0 & u_0 \\ 0 & fb & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ dimana *fa* dan *fb* merupakan panjang focus dari kamera dan Matriks $[\ I_3 \mid 0_3]$ didefinisikan sebagai $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$. Matriks R pada parameter ekstrinsik merupakan matriks rotasi, matriks T merupakan matriks translasi atau perpindahan serta matriks 0_3^T adalah vektor $[0 \quad 0 \quad 0]$. Maka dari itu matriks M merupakan matriks yang mengandung parameter intrinsik dan ekstrinsik.

Proses kalibrasi kamera dijalankan dengan menggunakan *library* EmguCV dengan memanfaatkan fungsi FindChessboardCorners yang berfungsi untuk mendeteksi papan catur dan fungsi Calibrate Camera yang berfungsi untuk mendapatkan parameter intrinsik dan ekstrinsik kamera.

```
public:
static array<PointF>^ FindChessboardCorners(
          Image<Gray, unsigned char>^ image,
          Size patternSize,
          CALIB_CB_TYPE flags
)
```

Gambar 2.2 Fungsi Deteksi Papan Catur pada Library EmguCV (http://www.emgu.com/wiki/files/3.3.0/document/html/8dee1f02-8c8a-4e37-87f4-05e10c39f27d.htm)

Gambar 2.2 merupakan fungsi yang terdapat pada *library* EmguCV yang berfungsi untuk mendeteksi papan catur pada gambar dengan memanfaatkan masukan data berupa gambar serta ukuran dari papan catur yang akan terdeteksi oleh sistem. Fungsi pada *library* EmguCV lainnya yang digunakan adalah fungsi CalibrateCamera yang tertera pada Gambar 2.3

```
public:
static double CalibrateCamera(
          array<array<MCvPoint3D32f>^>^ objectPoints,
          array<array<PointF>^>^ imagePoints,
          Size imageSize,
          IntrinsicCameraParameters^ intrinsicParam,
          CALIB_TYPE flags,
          [OutAttribute] array<ExtrinsicCameraParameters^>^% extrinsicParams
)
```

Gambar 2.3 Fungsi Mencari Intrinsik dan Ekstrinsik pada Library EmguCV(http://www.emgu.com/wiki/files/3.3.0/document/html/8dee1f02-8c8a-4e37-87f4-05e10c39f27d.htm)

Gambar 2.3 merupakan fungsi yang terdapat pada *library* EmguCV yang berfungsi untuk mendapatkan parameter intrinsik dan ekstrinsik dengan memanfaatkan data masukan berupa titik tiga dimensi dari objek, titik dua dimensi pada gambar serta ukuran dari gambar.

2.2 Face Detection

Menurut Erik Hjelmas dan Boon Kee Low dalam jurnal yang berjudul "Face Detection: A Survey" mengatakan bahwa, *face detection* atau pendeteksian wajah

merupakan langkah awal yang perlu dilakukan sebelum melakukan pengenalan wajah, bertujuan untuk menentukan posisi serta mengekstrak wajah dari latar belakang (2001:236). Perkembangan pada proses pendeteksian wajah membuat banyaknya *library* yang dapat digunakan dalam perancangan sistem salah satunya adalah *library* Face Luxand SDK. *Library FaceLuxand SDK* digunakan dalam penelitian ini dengan memanfaatkan fungsi DetectFace yang berfungsi mendeteksi wajah dan DetectFacialFeaturesInRegion yang berfungsi mendeteksi titik-titik pada wajah yang telah terdeteksi.

```
int FSDK.DetectFace(int Image, ref FSDK.TFacePosition
FacePosition);
```

Gambar 2.4 Fungsi Deteksi Wajah pada Library FaceLuxand SDK(https://www.luxand.com/facesdk/documentation/)

Gambar 2.4 merupakan fungsi yang terdapat pada *library* FaceLuxand SDK yang berfungsi untuk mendeteksi wajah pada gambar sebagai data masukan. Gambar tersebut akan diproses dan menghasilkan data keluaran berupa posisi dari wajah yang terdeteksi pada gambar. Fungsi pada *library* FaceLuxand SDK lainnya yang digunakan adalah DetectFacialFeatureInRegion yang tertera pada Gambar 2.5.

```
int FSDK.DetectFacialFeaturesInRegion(int Image, ref
FSDK.TFacePosition FacePosition, out FSDK.TPoint[]
FacialFeatures);
```

Gambar 2.5 Fungsi Deteksi Titik Wajah pada Library FaceLuxand SDK(https://www.luxand.com/facesdk/documentation/)

Gambar 2.5 merupakan fungsi yang terdapat *library* FaceLuxand SDK yang berfungsi untuk mendeteksi titik-titik pada wajah yang telah terdeteksi dengan memanfaatkan masukan data berupa gambar serta posisi wajah yang telah terdeteksi dengan menggunakan fungsi DetectFace sebelumnya.

2.3 Direct Linear Transformation

Algoritma *Direct Linear Transformation* (DLT) pertama kali dikembangkan oleh Abdel-Aziz dan Karara pada tahun 1971 dan dikembangkan kembali pada tahun 1993 oleh Gazzani. DLT merupakan sebuah algoritma untuk menentukan koordinat tiga dimensi dari sebuah titik dengan menggunakan beberapa citra dua dimensi.

Algoritma DLT dapat dilakukan jika memiliki 2 (dua) citra dua dimensi atau lebih. Pada citra dua dimensi tersebut memiliki matriks kalibrasi kamera yang dihasilkan dari proses kalibrasi kamera. Pada citra dua dimensi pertama dimisalkan dengan martiks L yang berukuran 4x3 dan memiliki koordinat (u_L, v_L) dan pada citra dua dimensi kedua dimisalkan matriks R yang berukuran 4x3 dan memiliki koordinat (u_R, v_R), maka dapat dilakukan rekonstruksi titik tiga dimensi dengan persamaan matriks berikut.

$$u_L = \frac{L_1x+L_2y+L_3z+L_4}{L_9x+L_{10}y+L_{11}z+1}$$

$$v_L = \frac{L_5x+L_6y+L_7z+L_8}{L_9x+L_{10}y+L_{11}z+1}$$

$$u_R = \frac{R_1x+R_2y+R_3z+R_4}{R_9x+R_{10}y+R_{11}z+1} \quad \ldots(2.3)$$

$$v_R = \frac{R_5x+R_6y+R_7z+R_8}{R_9x+R_{10}y+R_{11}z+1}$$

Dimana x, y, dan z merupakan data titik tiga dimensi yang akan dicari. Dengan adanya persamaan di atas maka untuk mencari titik x, y, dan z dapat dimisalkan menjadi persamaan berikut**.**

$$\begin{bmatrix} L_1 - L_9u_L & L_2 - L_{10}u_L & L_3 - L_{11}u_L \\ L_5 - L_9v_L & L_6 - L_{10}v_L & L_7 - L_{11}v_L \\ R_1 - R_9u_R & R_2 - R_{10}u_R & R_3 - R_{11}u_R \\ R_5 - R_9v_R & R_6 - R_{10}v_R & R_7 - R_{11}v_R \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u_L - L_4 \\ v_L - L_8 \\ u_R - R_4 \\ v_R - R_8 \end{bmatrix} \quad \ldots(2.4)$$

Persamaan di atas dapat dilanjutkan dengan menganggap matriks kiri menjadi matriks Q dan matriks kanan menjadi matriks q makan persamaan di atas menjadi persamaan berikut.

$$Q \begin{bmatrix} x \\ y \\ z \end{bmatrix} = q \quad \ldots(2.5)$$

Mencari titik x, y, z di atas dapat menggunakan metode Moore-Penrose pseudo-inverse, sehingga persamaan di atas menjadi berikut.

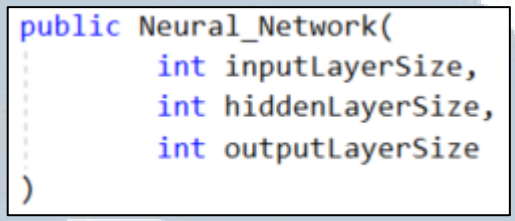
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = (Q^T Q)^{-1} Q^T q \quad \ldots(2.6)$$

Dengan menyelesaikan persamaan di atas maka titik x, y, dan z yang berupa titik tiga dimensi berhasil didapatkan.

2.4 Face Recognition

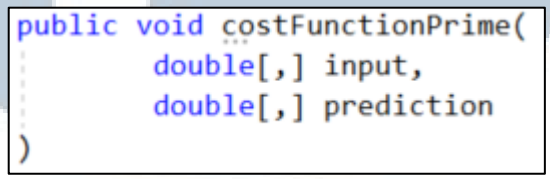
Face recognition merupakan metode biometrik yang tidak seperti metode biometrik lainnya dikarenakan metode biometrik *face recognition* bersifat *non-intrusive* dan dapat digunakan tanpa pengetahuan dari subyek (Alexander M. & Micahel M., 2003). Perkembangan pada proses pengenalan wajah membuat banyaknya *library* yang dapat digunakan dalam perancangan sistem. Proses pengenalan wajah dijalankan dengan menggunakan *library neural network backpropagation* dengan memanfaatkan fungsi Neural_Network yang berfungsi untuk membuat jaringan, CostFunctionPrime yang berfungsi untuk melakukan pelatihan pada jaringan, CostFunction yang berfungsi untuk melakaukan

perhitungan MSE, dan fungsi Fordward yang berfungsi untuk melakukan pengujian.



Gambar 2.6 Fungsi Neural_Network pada Library (Julio, 2016)

Gambar 2.6 merupakan fungsi yang terdapat pada *library neural network backpropagation* yang berfungsi untuk membuat jaringan pada *neural network* dengan memanfaatkan masukan data berupa banyak *neuron* yang terdapat pada lapisan masukan, lapisan tersembunyi, serta lapisan keluaran.



Gambar 2.7 Fungsi CostFunctionPrime pada Library (Julio, 2016)

Gambar 2.7 merupkan fungsi yang terdapat pada *library neural network backpropagation* yang berfungsi untuk melakukan pelatihan jaringan dengan menggunakan metode *backpropagation*. Fungsi tersebut dijalankan dengan memanfaatkan masukan data berupa *input* dan data prediksi keluaran.

```
public double costFunction(
        double[,] input,
        double[,] prediction
)
```

Gambar 2.8 Fungsi CostFunction pada Library (Julio, 2016)

Gambar 2.8 merupakan fungsi yang terdapat pada *library neural network backpropagation* yang berfungsi untuk melakukan perhitungan *Mean Squared Error* (MSE) dengan memanfaatkan masukan data berupa data *input* dan data prediksi keluaran.

```
public double[,] forward(
        double[,] input
)
```

Gambar 2.9 Fungsi Fordward pada Library (Julio, 2016)

Gambar 2.9 merupakan fungsi yang terdapat pada *library neural network backpropagation* yang berfungsi untuk melakukan perhitungan prediksi hasil keluaran dengan memanfaatkan data masukan berupa *input*. Terdapat 2 jenis pendekatan dari *face recognition* yaitu sebagai berikut.

2.4.1 2D Face Recognition

Pendekatan 2D *face recognition* merupakan pendekatan yang berbasis citra atau gambar, pendekatan tersebut menggunakan 2 tipe koordinat (x, y) yaitu koordinat lebar (x) dan koordinat tinggi (y). Pendekatan 2D *face recognition* memiliki kelemahan yaitu mudah dipengaruhi oleh keadaan lingkungan, orientasi wajah, ekspresi wajah, dan dandanan.

2.4.2 3D Face Recognition

Pendekatan 3D *face recognition* menggunakan 3 tipe koordinat (x, y, z) yaitu koordinat lebar (x), koordinat tinggi (y), dan koordinat kedalaman (z). Pendekatan 3D *face regocnition* sangat membantu untuk dapat mengatasi malasah intrinsik yang dimiliki oleh pendekatan 2D.

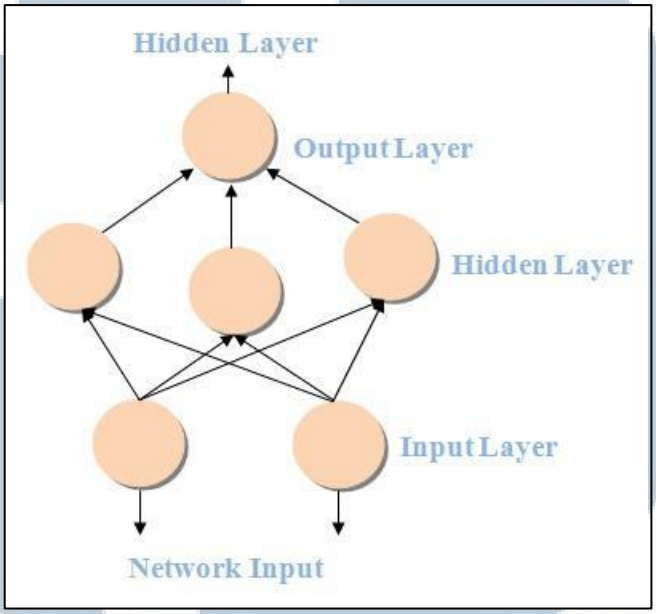
2.5 Neural Network

Neural network/jaringan syaraf tiruan adalah paradigma pengelolaan informasi yang terispirasi sistem saraf secara biologis, seperti proses informasi pada otak manusia (Manalu, 2016:35). Menurut Wuryandari dan Afrianto (2012:46), *neural network* merupakan suatu sistem pemrosesan informasi yang mempunyai karakteristik menyerupai jaringan syaraf biologi dan tercipta sebagai suatu generalisasi model matematis dari pemahaman manusia (*human cognition*) dan didasarkan atas asumsi sebagai berikut.

1. Pemrosesan informasi terjadi pada elemen sederhana yang disebut *neuron*
2. Sinyal mengalir diantara sel syaraf/*neuron* melalui suatu sambungan penghubung
3. Setiap sambungan penghubung memiliki bobot yang bersesuaian. Bobot ini akan digunakan untuk menggandakan / mengalikan sinyal yang dikirim melaluinya.
4. Setiap sel syaraf akan menerapkan fungsi aktivasi terhadap sinyal hasil penjumlahan berbobot yang masuk kepadanya untuk menentukan sinyal keluarannya.

Menurut Manalu pada jurnal yang berjudul “Jaringan Syaraf Tiruan untuk Memprediksi Curah Hujan Sumatera Utara dengan Metode Back Propagation

(Studi Kasus: BMKG Medan)" mengatakan bahwa *Neural network* memiliki arsitektur yang terdiri atas beberapa lapisan yaitu lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer*), dan lapisan keluaran (*output layer*) (2016:36).



Gambar 2.10 Arsitektur Neural Network(Manalu, 2016)

Setiap *Neuron* pada setiap lapisan akan terhubung dengan setiap *neuron* pada lapisan sebelum dan sesudahnya (Wuryandari, 2012:46). Terlihat pada Gambar 2.4, setiap *neuron* pada lapisan masukan (*input layer*) terhubung dengan setiap *neuron* pada lapisan tersembunyi (*hidden layer*) dan setiap *neuron* pada lapisan tersembunyi (*hidden layer*) terhubung dengan setiap *neuron* pada lapisan keluaran (*ouput layer*).

2.6 Backpropagation

Backpropagation merupakan salah datu metode pelatihan pada *neural network* dengan menggunakan arsitektur *multilayer network* dengan metode pelatihan *supervised learning* (Pakaja, 2012:24). *Multilayer network*/ jaringan banyak lapisan adalah sebuah arsitektur jaringan pada *neural network* yang

memiliki 1(satu) atau lebih lapisan tersembunyi diantara lapisan masukan dan lapisan keluaran (Wuryandari, 2012:46). Menurut Manalu (2016:36-37), pelatihan *backpropagation* memiliki beberapa langkah sebagai berikut.

1. Inisialisasi bobot dengan menggunakan nilai *random*.
2. Tahap perambatan maju (*fordward propagation*)
   a. Setiap *neuron* pada lapisan masukan ($X_i, i = 1,2,3, \dots, n$) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua *neuron* yang terdapat pada lapisan tersembunyi.
   b. Setiap *neuron* pada lapisan tersembunyi ($Z_j, j = 1,2,3, \dots, p$) melakukan penjumlahan bobot sinyal *input*, ditunjukan pada Rumus 2.7

$$Z_j = v_{0j} + \sum_{i=1}^{n} X_i v_{ij} \quad \dots(2.7)$$

   Serta menerapkan fungsi aktivasi untuk menghitung sinyal *output*, ditujukan pada Rumus 2.8

$$Z_j = f(Z_j) \quad \dots(2.8)$$

   Fungsi aktivasi yang digunakan adalah fungsi *sigmoid*, selanjutnya mengirimkan sinyal *output* tersebut ke semua *neuron* yang terdapat pada lapisan keluaran.
   c. Setiap *neuron* pada lapisan keluaran ($Y_k, k = 1,2,3, \dots, m$) melakukan penjumlahan bobot sinyal *input*, ditunjukan pada Rumus 2.9

$$Y_k = w_{0k} + \sum_{j=1}^{n} Z_j w_{jk} \quad \dots(2.9)$$

   Serta menerapkan fungsi aktivasi dengan tujuan yang sama dengan fungsi aktivasi pada langkah b, yaitu untuk menghitung sinyal *output*, ditujukan pada Rumus 2.10

$$Z_j = f(Z_j) \quad \text{…(2.10)}$$

3. Tahap perambatan balik (*back propagation*)

   a. Setiap *neuron* pada lapisan keluaran ($Y_k, k = 1,2,3, \dots, m$) menerima pola target yang sesuai dengan pola *input* pelatihan, kemudian hitung *error* ditujukan pada Rumus 2.11

   $$\delta_k = (t_k - y_k) f'(y_k) \quad \text{…(2.11)}$$

   f' merupakan turunan dari fungsi *sigmoid* dan selanjutnya melakukan perhitungan kolerasi bobot yang ditujukan pada Rumus 2.12

   $$\Delta w_{kj} = \propto \delta_k Z_j \quad \text{…(2.12)}$$

   Selanjutnya melakukan perhitungan koreksi bias yang ditujukan pada Rumus 2.13 serta mengirimkan δ_k ke semua *neuron* yang terdapat pada lapisan paling kanan dari lapisan tersembunyi.

   $$\Delta w_{0k} = \propto \delta_k \quad \text{…(2.13)}$$

   b. Setiap *neuron* pada lapisan tersembunyi ($Z_j, j = 1,2,3, \dots, p$) menjumlahkan delta *input* (dari *neuron* yang berada pada lapisan di kanannya), ditunjukkan pada Rumus 2.14

   $$\delta_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad \text{…(2.14)}$$

   Langkah selanjutnya adalah menghitung informasi *error* dengan cara mengkalikan nilai tersebut dengan turunan dari fungsi aktivasinya, ditunjukkan pada Rumus 2.15

   $$\delta_j = \delta_j f'(Z_j) \quad \text{…(2.15)}$$

   Kemudian lakukan perhitungan koreksi bobot seperti yang ditunjukkan pada Rumus 2.16

$$\Delta v_{jk} = \propto \delta_j X_i \quad \text{…(2.16)}$$

Langkah terakhir pada *back propagation* adalah melakukan perhitungan koreksi bias yang ditunjukkan pada Rumus 2.17

$$\Delta v_{0j} = \propto \delta_j \quad \text{…(2.17)}$$

4. Tahap perubahan bobot dan bias

Setiap *neuron* pada lapisan keluaran ($Y_k, k = 1,2,3, \dots, m$) dilakukan perubahan bobot dan bias ($j = 1,2,3, \dots, p$), ditunjukkan pada Rumus 2.18

$$w_{jk}baru = w_{jk}lama + \Delta w_{jk} \quad \text{…(2.18)}$$

Setiap *neuron* pada lapisan tersembunyi ($Z_j, j = 1,2,3, \dots, p$) dilakukan perubahan bobot dan bias ($i = 1,2,3, \dots, n$), ditunjukkan pada Rumus 2.19

$$v_{ij}baru = v_{ij}lama + \Delta v_{ij} \quad \text{…(2.19)}$$

Tahap pada pelatihan *backpropagation* tersebut akan terus dilakukan jika nilai Mean Squared Error (MSE) tidak tercapai. Nilai MSE pada satu siklus (tahap 1 – 4 pada tahap pelatihan *backpropagation*) adalah nilai kesalahan rata-rata dari pelatihan *backpropagation* dan dirumuskan sebagai berikut (Dhaneswara, 2004:118-119).

$$MSE = \frac{\sum(error)^2}{total\ record} \quad \text{…(2.20)}$$

Semakin kecil MSE maka semakin kecil kesalahan dalam memprediksi hasil dari pengujian (Dhaneswara, 2004: 118-119). Maka dari itu tujuan dari pelatihan *backpropagation* adalah untuk memperkecil nilai MSE hingga nilai MSE yang diinginkan tercapai.

2.7 False Acceptance Rate (FAR)

False acceptance rate merupakan probabilitas bahwa sistem melakukan kesalahan menyatakan kecocokan antara pola dari masukan data dengan pola yang terdapat dari database (Pankaj Sareen, 2014). Simaremare dan Kurniawan mengatakan bahwa,

> *False acceptance rate* (FAR) adalah kesalahan dalam mengenali identitas gambar masukan, baik itu kesalahan dalam mengenali identitas gambar masukan dari individu di luar *database* yang terdeteksi sebagai individu di dalam *database*, maupun kesalahan dalam mengenali identitas gambar masukan dari individu di dalam *database* yang dikenali sebagai individu lain. (Simaremare & Kurniawan, 2016:68).

$$FAR = \frac{Banyak\ FAR}{Jumlah\ Percobaan} \times 100\% \quad ...(2.21)$$

Persentase FAR didapatkan dengan mengukur persentase dari banyaknya FAR yang terjadi dibagi dengan banyaknya jumlah percobaan yang dilakukan dalam proses pengujian.

2.8 False Rejection Rate (FRR)

False rejection rate merupakan probabilitas bahwa sistem melakukan kesalahan menyatakan ketidakcocokan antara pola dari masukan data dengan pola yang terdapat dari database (Pankaj Sareen, 2014). Santikasari, Atmaja, dan Susatio mengatakan bahwa,

> FRR (*False Rejection Rate*) merupakan perhitungan kesalahan pengenalan seperti halnya FAR, akan tetapi pada FRR merupakan kasus ketika suatu individu yang tidak dikenali dikenali sebagai salah satu individu yang terdapat dalam database. (Santikasari, Atmaja, & Susatio, 2016: 425).

$$FRR = \frac{Banyak\ FRR}{Jumlah\ Percobaan} \times 100\% \quad ...(2.22)$$

Persentase FRR didapatkan dengan mengukur persentase dari banyaknya FRR yang terjadi dibagi dengan banyaknya jumlah percobaan yang dilakukan dalam proses pengujian.