BAB III
METODE PENELITIAN DAN PERANCANGAN

Didalam bab metode penelitian dan perancangan akan dibahas mengenai studi literatur yang digunakan, analisis kebutuhan, perancangan sistem pengenalan sidik yang meliputi *database*, proses pengenalan sidik jari dan perancangan ERD, perhitungan manual, perancangan *interface* dan skenario pengujian.

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi Citra Digital, Pengenalan Sidik Jari (*Fingerprint Recognition*), *Thresholding*, Metode Otsu, Skeletonisasi, Algoritma Zhang-Suen, Morfologi, *Structuring Element*, Ekstraksi fitur, Metode *Template Matching*, dan *Hidden Markov Model*

3.2 Analisis Kebutuhan

Seluruh proses yang dilakukan dibuat menggunakan perangkat lunak untuk memecahkan masalah matematis dimana komputer yang digunakan memiliki spesifikasi sebagai berikut :

a. Performa : Intel (R) Core (TM) i5 M460 @2,53 GHz
b Sistem operasi : Windows 7 Ultimate 32-bit (6.1, Build 7600)
c. VGA card : ATI Mobility Radeon HD 5470
d. Aplikasi dibangun dengan Microsoft Visual Studio 2008.
e. Aplikasi database yang digunakan adalah MySQL yang ada dalam XAMPP 1.7.7
f. Data citra sidik jari didapatkan dari citra sidik jari Verifinger_Sample_DB yang terdapat di halaman website http://www.neurotechnology.com/verifinger.html. Data didapatkan dalam bentuk file dengan format gambar TIFF. Setiap file memiliki format nama file sebagai berikut : xxx_y_z dimana x adalah ID orang, y adalah ID jari, dan z adalah nomor dari *scan* dengan ukuran 504 x 408 piksel.

3.3 Perancangan Sistem

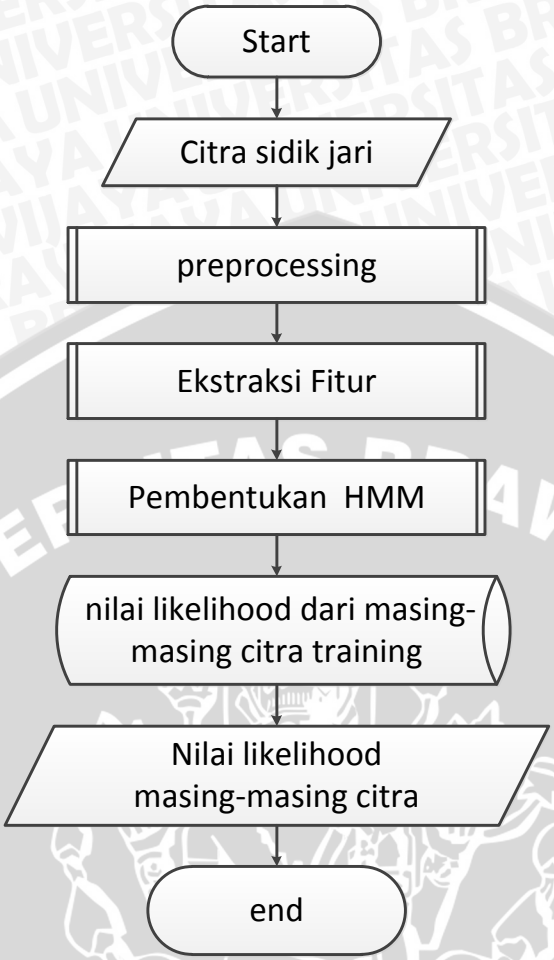
Pada bagian ini akan dipaparkan mengenai langkah-langkah yang akan dijalankan untuk melakukan pelatihan dan pengenalan citra sidik jari dari Verifinger_Sample_DB.

3.3.1 Perancangan *Training*

Langkah-langkah yang akan dilakukan untuk membentuk data *training* adalah :

1. Memasukkan citra sidik jari yang ingin dijadikan sebagai data *training.*
2. Citra yang sudah diinputkan akan di *preprocessing*.
3. Dari hasil *preprocessing* akan dicari *bifurcation* (percabangan) pada masing-masing citra sidik jari proses ini dinamakan ekstraksi fitur.
4. Setelah memasukkan hasil ekstraksi fitur, *state*, label, dan id ke *database* langkah selanjutnya adalah membentuk parameter HMM dan mengestimasinya sampai konvergen.
5. Setelah konvergen maka akan dievaluasi sehingga didapat nilai *likelihood* yang optimal. Nilai *likelihood* akan dimasukkan ke dalam *database* sesuai dengan id masing-masing citra.
6. proses *database* selesai.

Diagram alir proses data *training* dapat digambarkan pada Gambar 3.1 berikut ini :



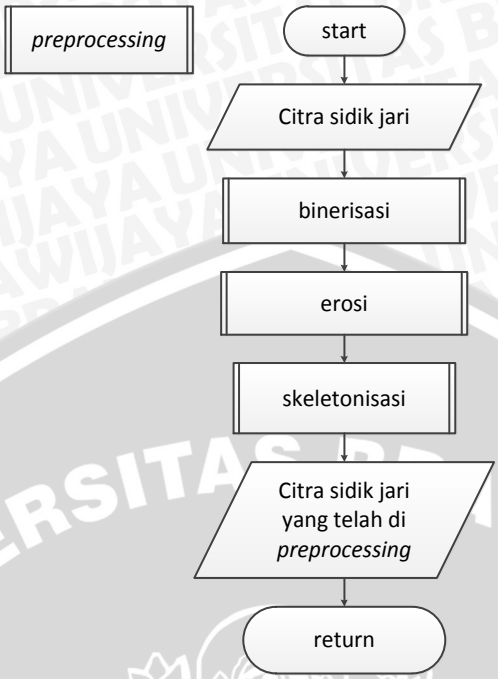
Gambar 3.1 Diagram alir *training*

3.3.2 Perancangan *Preprocessing*

Pada tahap *preprocessing* citra sidik jari langkah-langkahnya antara lain :

1. Memasukkan citra sidik jari.
2. Memproses citra sidik jari menjadi citra biner menggunakan metode Otsu.
3. Hasil dari citra biner akan dierosi untuk menghilangkan noise.
4. Hasil dari erosi akan diskeletonisasi yaitu menjadikan citra sidik jari setebal 1 *pixel* dengan algoritma Zhang-Suen.
5. Proses *preprocessing* selesai.

Diagram alir proses *preprocessing* dapat digambarkan pada Gambar 3.2 berikut ini :



Gambar 3.2 Diagram alir *preprocessing*

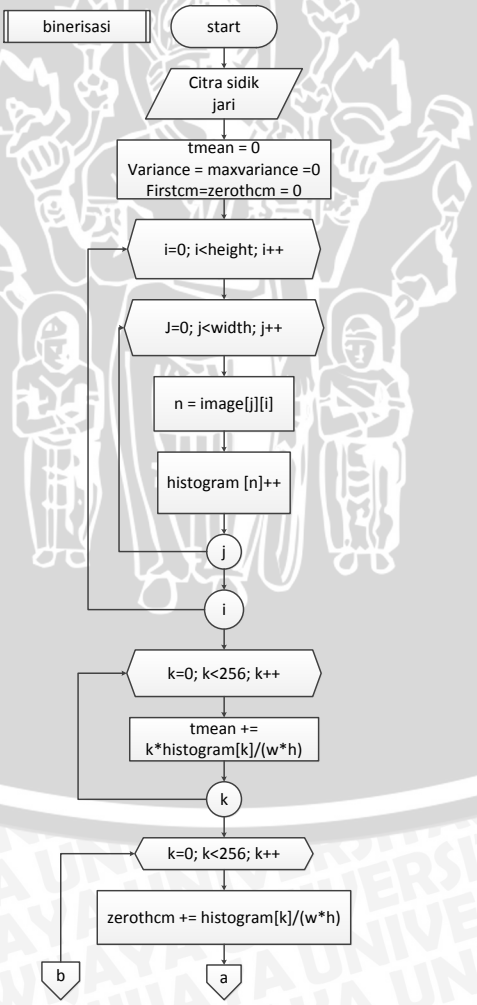
3.3.2.1 Perancangan Binerisasi

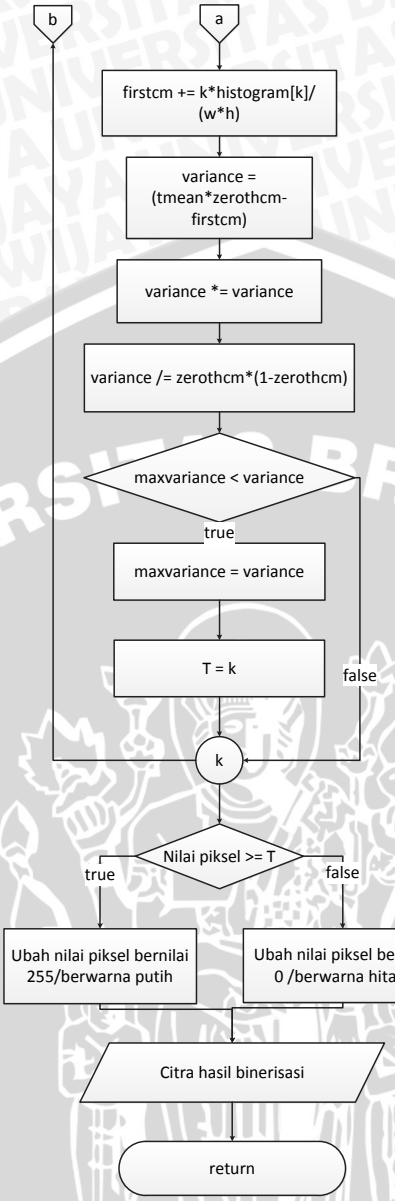
Langkah-langkah yang ada dalam proses binerisasi adalah sebagai berikut :

1. Memasukkan citra sidik jari.
2. Memberi nilai awal *tmean, variance, maxvariance, firstcm,* dan *zerothcm*.
3. Melakukan perulangan *height* dan *width* pada citra.
4. Membentuk histogram warna.
5. Melakukan iterasi hingga selesai dan didapatkan histogram citra.
6. Melakukan perulangan untuk nilai piksel.
7. Menghitung *tmean* dari semua piksel dengan persamaan (2.5).
8. Melakukan iterasi hingga selesai dan didapatkan nilai *tmean*.
9. Melakukan perulangan untuk nilai piksel.
10. Menghitung nilai *zerothcm* masing-masing piksel dengan persamaan (2.3).
11. Menghitung nilai *firstcm* masing-masing piksel dengan persamaan (2.4).
12. Menghitung nilai *variance* pada masing-masing piksel dengan persamaan (2.7).

13. Jika nilai *maxvariance* lebih kecil dari *variance* maka *maxvariance* = *variance* dan nilai *threshold* sama dengan nilai piksel, jika tidak maka ke langkah 14.
14. Melakukan iterasi hingga selesai sehingga didapatkan nilai *variance* tertinggi.
15. Membandingkan nilai masing-masing piksel dengan nilai *threshold* jika nilai piksel lebih besar sama dengan nilai *threshold* maka nilai piksel diubah menjadi 255 (putih) sedangkan jika nilai piksel kurang dari nilai *threshold* maka nilai piksel diubah menjadi 0 (hitam).
11. Proses binerisasi dengan metode Otsu selesai.

Diagram alir proses binerisasi dengan metode otsu dapat digambarkan pada Gambar 3.3 berikut ini :





Gambar 3.3 Diagram alir binerisasi dengan metode Otsu

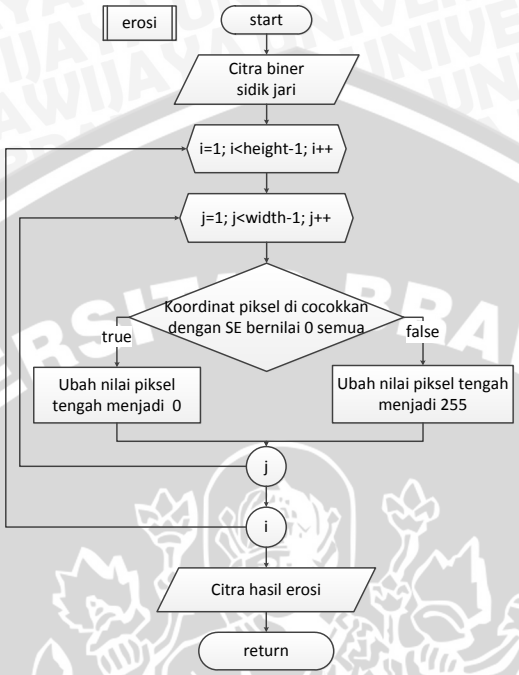
3.3.2.2 Perancangan Erosi

Langkah – langkah yang ada dalam proses erosi sebagai berikut :

1. Citra sidik jari dari hasil proses binerisasi
2. Melakukan perulangan *height* dan *width* pada citra sidik jari
3. Jika nilai piksel citra sama dengan nilai SE maka nilai tengah piksel diubah menjadi 0, sedangkan jika salah satu atau semua nilai piksel berbeda maka nilai tengah piksel diubah menjadi 255.

4. Melakukan iterasi hingga selesai dan didapatkan hasil erosi.
5. Proses erosi selesai.

Diagram alir proses erosi dapat digambarkan pada Gambar 3.4 berikut ini :



Gambar 3.4 Diagram alir proses erosi

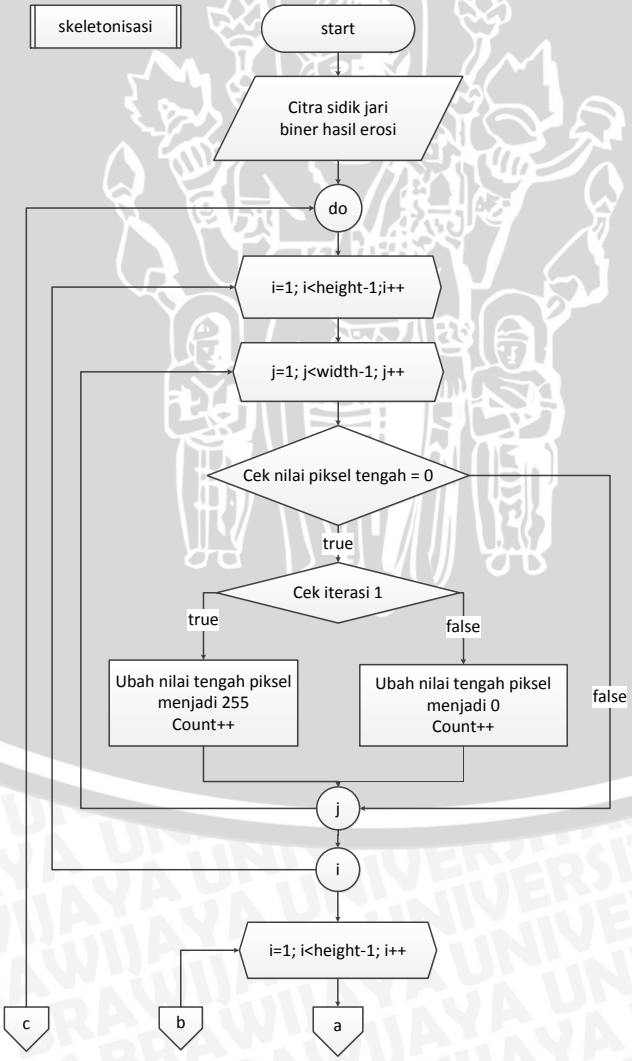
3.3.2.3 Perancangan Skeletonisasi

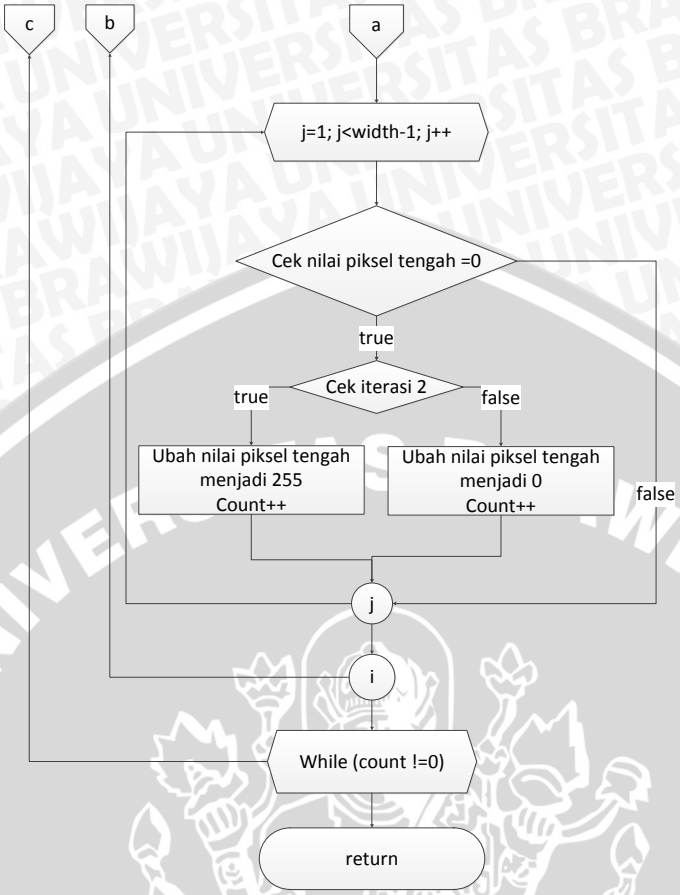
Langkah – langkah yang ada dalam proses skeletonisasi sebagai berikut :

1. Hasil citra sidik jari yang telah dierosi.
2. Melakukan perulangan *height* dan *width* pada citra.
3. Jika nilai piksel tengah tidak bernilai 0 maka melakukan perulangan *width* dan *height* lagi sampai menemukan piksel yang bernilai 0, jika benar maka piksel yang bernilai 0 akan diiterasi pertama.
4. Jika iterasi pertama bernilai benar semua maka ubah nilai tengah piksel menjadi 255, jika salah satu atau semuanya salah maka ubah nilai piksel tengah menjadi 0 dan tambahkan 1 *count*-nya.
5. Melakukan iterasi pertama hingga selesai.
6. Melakukan perulangan *height* dan *width* pada citra hasil iterasi pertama.

7. Jika nilai piksel tengah tidak bernilai 0 maka melakukan perulangan *width* dan *height* lagi sampai menemukan piksel yang bernilai 0, jika benar maka piksel yang bernilai 0 akan diiterasi kedua.
8. Jika iterasi kedua bernilai benar semua maka ubah nilai tengah piksel menjadi 255, jika salah satu atau semuanya salah maka ubah nilai piksel tengah menjadi 0 dan tambahkan 1 *count*-nya.
9. Melakukan iterasi kedua hingga selesai
10. Melakukan langkah 2 sampai dengan langkah 9 sampai *count* tidak berubah atau tetap.
11. Proses skeletonisasi dengan algoritma Zhang-Suen selesai.

Diagram alir proses skeletonisasi dengan algoritma Zhang-Suen dapat digambarkan pada Gambar 3.5 berikut ini :





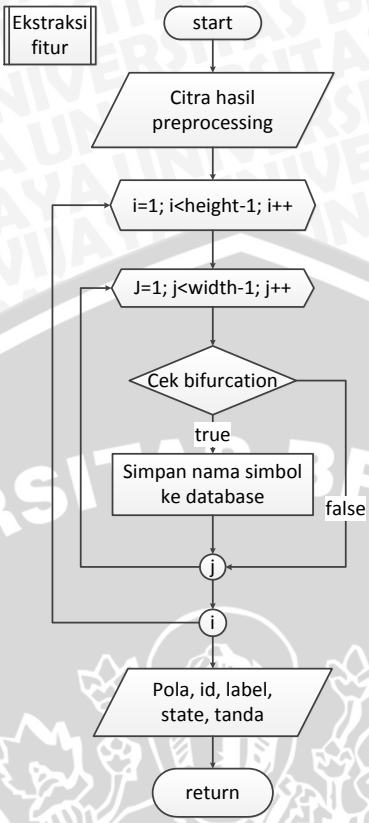
Gambar 3.5 Diagram alir proses skeletonisasi dengan algoritma Zhang-Suen

3.3.3 Perancangan Ekstraksi fitur

Langkah-langkah proses ekstraksi fitur adalah :

1. Citra sidik jari dari hasil proses *preprocessing*.
2. Melakukan perulangan *height* dan *width* pada citra.
3. Jika terdapat *template bifurcation* yang sama seperti pada Gambar 2.6 maka simpan nama simbol *bifurcation* ke dalam *database*, Jika tidak melakukan perulangan *width* dan *height* lagi sampai habis dan cek *bifurcation*nya.
4. Menyimpan *state* , id, dan label ke dalam *database*.
5. Menandai area yang terdapat *bifurcation*.
6. Proses ekstraksi fitur selesai.

Diagram alir proses ekstraksi fitur dapat digambarkan pada Gambar 3.6 berikut ini :



Gambar 3.6 Diagram alir proses ekstraksi fitur

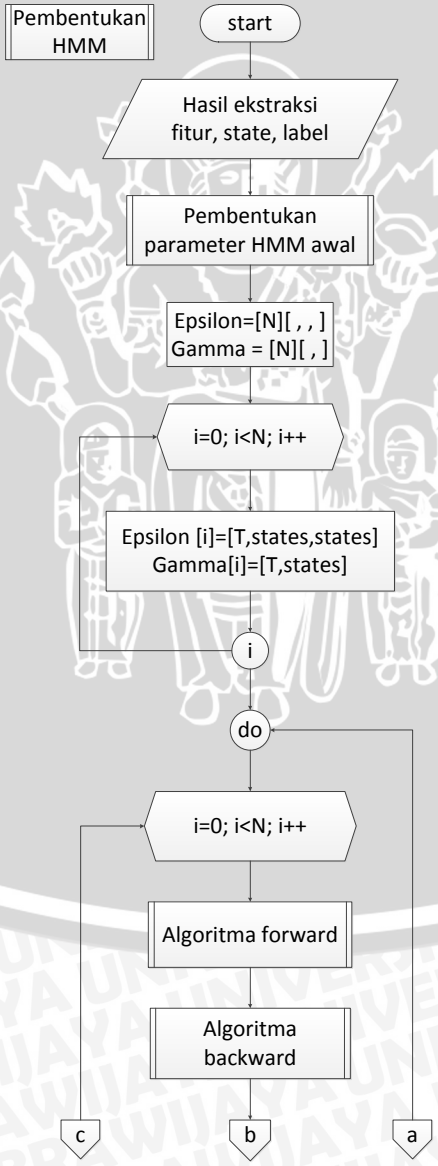
3.3.4 Perancangan Parameter HMM

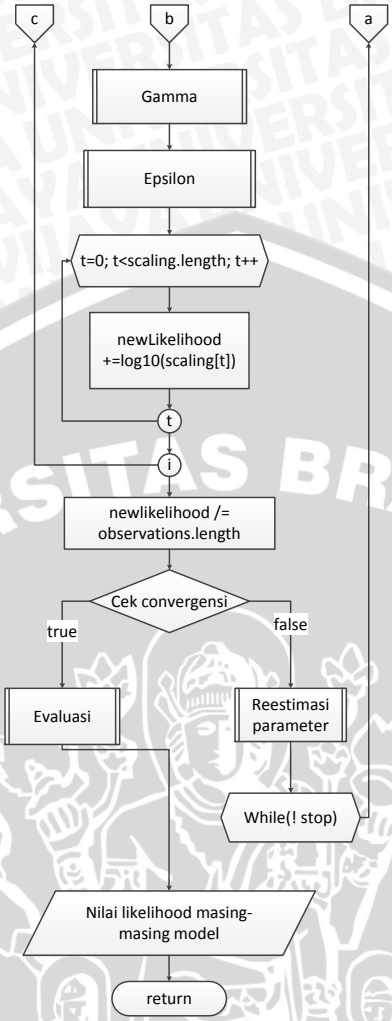
Langkah-langkah dalam proses parameter HMM adalah sebagai berikut :

1. Hasil ekstraksi fitur, *state*, dan label sebagai data input.
2. Membentuk parameter HMM awal.
3. Menentukan ukuran matriks *epsilon* dan *gamma*.
4. Melakukan perulangan untuk jumlah label dan menentukan definisi matriks *gamma* dan *epsilon*.
5. Melakukan perulangan untuk jumlah label.
6. Menggunakan algoritma *forward* untuk menghitung nilai *epsilon* dan *gamma*.
7. Menggunakan algoritma *backward* untuk menghitung nilai *epsilon* dan *gamma*.
8. Menghitung nilai matriks *gamma*.
9. Menghitung nilai matriks *epsilon*.

10. Melakukan perulangan untuk jumlah deretan atau *sequence*.
11. Menghitung nilai *likelihood* baru.
12. Melakukan iterasi hingga selesai.
13. Menetapkan *likelihood* baru sama dengan *likelihood* baru dibagi 1.
14. Jika rata-rata *likelihood* baru sudah konvergen maka akan dievaluasi, jika tidak maka parameter HMM akan diestimasi kembali sampai konvergen.
15. Menampilkan dan menyimpan nilai *likelihood* model.
16. Proses parameter HMM selesai.

Diagram alir proses parameter HMM dapat digambarkan pada Gambar 3.7 berikut ini :





Gambar 3.7 Diagram alir parameter HMM

3.3.4.1 Perancangan Parameter HMM Awal

Langkah-langkah dalam proses parameter HMM awal adalah :

1. Hasil ekstraksi fitur, *state* dan simbol sebagai data input
2. Memproses data input ke dalam proses matriks emisi awal.
3. Memproses data input ke dalam proses matriks transisi dan inisial awal.
4. Proses parameter HMM awal selesai

Diagram alir proses parameter HMM awal dapat digambarkan pada Gambar 3.8 berikut ini :

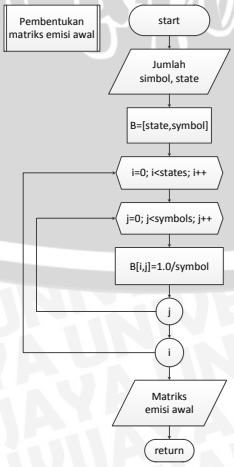
Gambar 3.8 Diagram alir parameter HMM awal

3.3.4.1.1Perancangan Matriks Emisi Awal

Langkah-langkah dalam proses matriks emisi awal adalah :

1. Jumlah dari simbol dan *state* sebagai data input.
2. Menentukan ukuran matriks emisi /observasi.
3. Melakukan perulangan untuk *state*.
4. Melakukan perulangan untuk simbol.
5. Menghitung nilai emisi.
6. Melakukan iterasi hingga selesai dan didapatkan nilai matriks emisi awal.
7. Proses matriks emisi awal selesai.

Diagram alir proses matriks emisi awal dapat digambarkan pada Gambar 3.9 berikut ini :



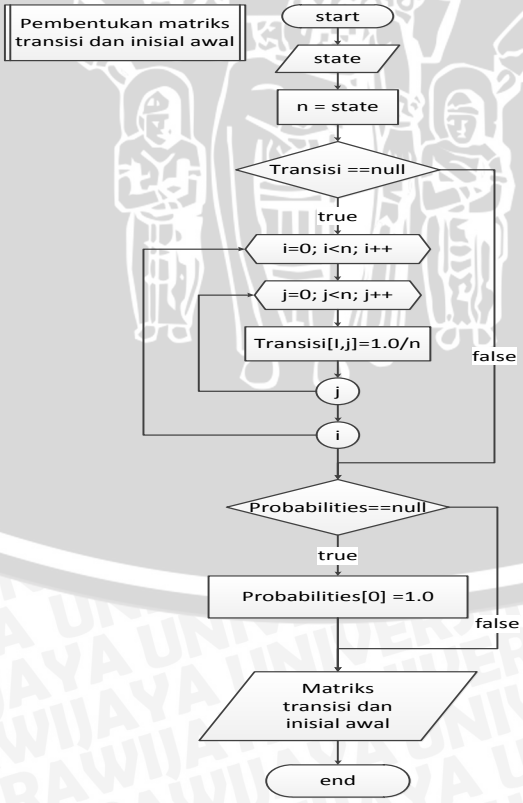
Gambar 3.9 Diagram alir matriks emisi awal

3.3.4.1.2Perancangan Matriks Transisi Dan Inisial Awal

Langkah-langkah pada proses transisi dan inisial awal adalah :

1. Nilai *state* sebagai data input.
2. Menentukan ukuran *n.*
3. Jika nilai transisi kosong (tidak ada isinya) maka berlanjut ke langkah 4, jika tidak maka berlanjut ke langkah 8.
4. Melakukan perulangan untuk *state i.*
5. Melakukan perulangan untuk *state j.*
6. Melakukan perhitungan nilai transisi.
7. melakukan iterasi *state i* dan *j* hingga selesai.
8. Jika probabilitas sama dengan kosong (tidak ada isinya) maka berlanjut ke langkah 10 jika tidak maka berlanjut ke langkah 11.
9. Menentukan nilai probabilitas urutan ke 0 dari *array*.
10. Proses matriks transisi dan matriks inisial awal selesai.

Diagram alir proses matriks transisi dan inisial awal dapat digambarkan pada Gambar 3.10 berikut ini :



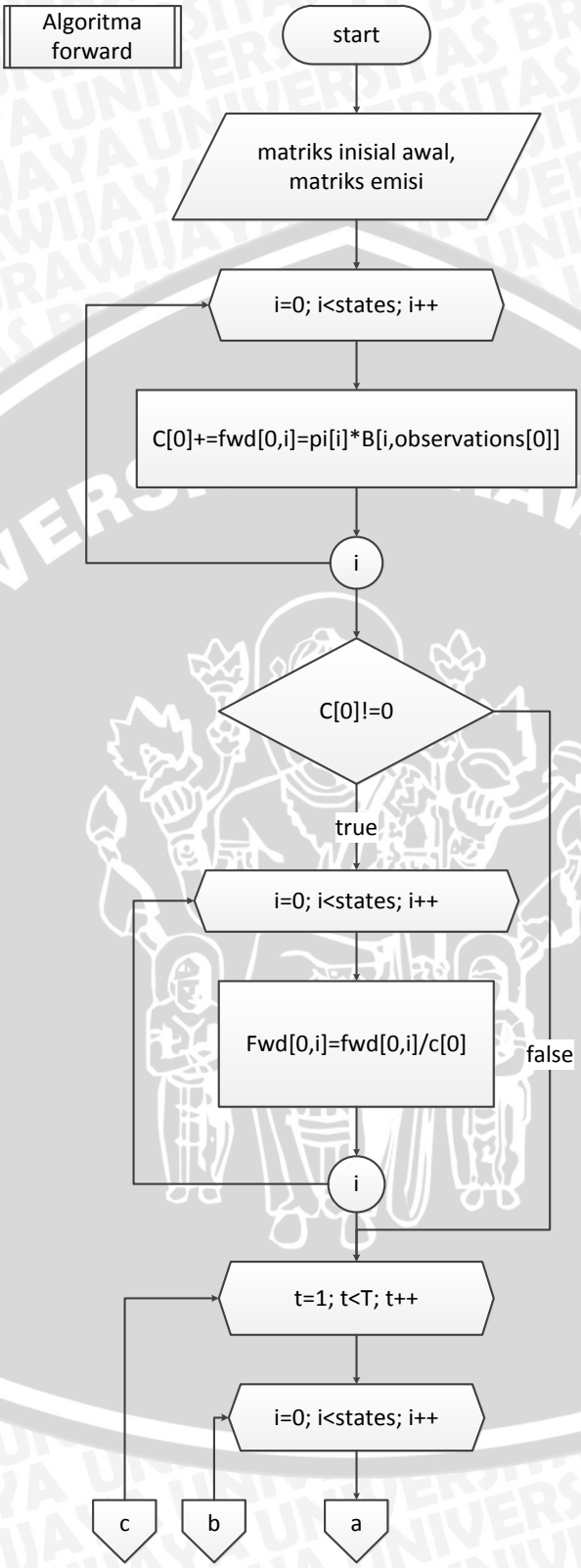
Gambar 3.10 Diagram alir matriks inisial dan transisi awal

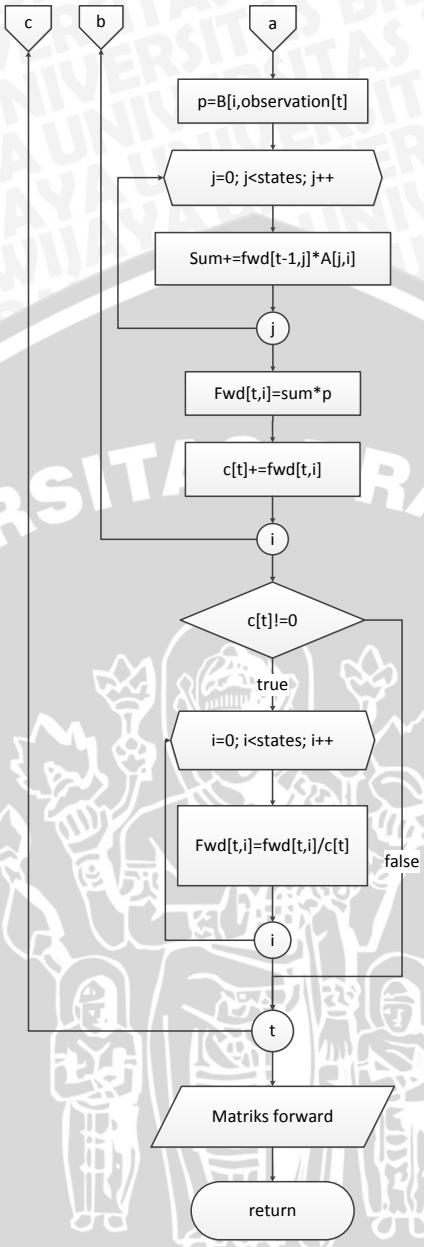
3.3.4.2 Perancangan Algoritma *Forward*

Langkah-langkah dalam proses algoritma *forward* adalah :

1. Matriks inisial awal dan matriks emisi sebagai data input.
2. Melakukan perulangan untuk *state*.
3. Menghitung insialisasi dari algoritma *forward*.
4. Jika nilai inisialisasi sama dengan 0 maka berlanjut ke langkah 7, jika tidak maka berlanjut ke langkah 5.
5. Melakukan perulangan untuk *state*.
6. Menghitung nilai normalisasi dari inisialisasi algoritma *forward*.
7. Melakukan perulangan untuk panjang deretan / *sequence*.
8. Melakukan perulangan untuk *state i.*
9. Menentukan nilai *p* adalah matriks emisi.
10. Melakukan perulangan untuk *state j.*
11. Menghitung proses induksi dari algoritma *forward*.
12. Jika nilai proses induksi tidak sama dengan 0 maka ke langkah 13, jika sama dengan 0 maka ke langkah 15.
13. Melakukan penelusuran untuk *state i.*
14. Menghitung nilai normalisasi dari proses induksi.
15. Melakukan iterasi hingga selesai dan didapatkan output nilai matriks *forward* yang ternormalisasi.
16. Proses algoritma *forward* selesai.

Algoritma forward digunakan untuk mengevaluasi parameter HMM. sesuai dengan namanya algoritma forward berjalan dengan runut maju. Adapun Diagram alir proses algoritma *forward* dapat digambarkan pada Gambar 3.11 berikut ini :


Algoritma forward
start
matriks inisial awal, matriks emisi
i=0; i<states; i++
C[0]+=fwd[0,i]=pi[i]*B[i,observations[0]]
i
C[0]!=0
true
i=0; i<states; i++
Fwd[0,i]=fwd[0,i]/c[0]
false
i
t=1; t<T; t++
i=0; i<states; i++
c
b
a



Gambar 3.11 Diagram alir algoritma *forward*

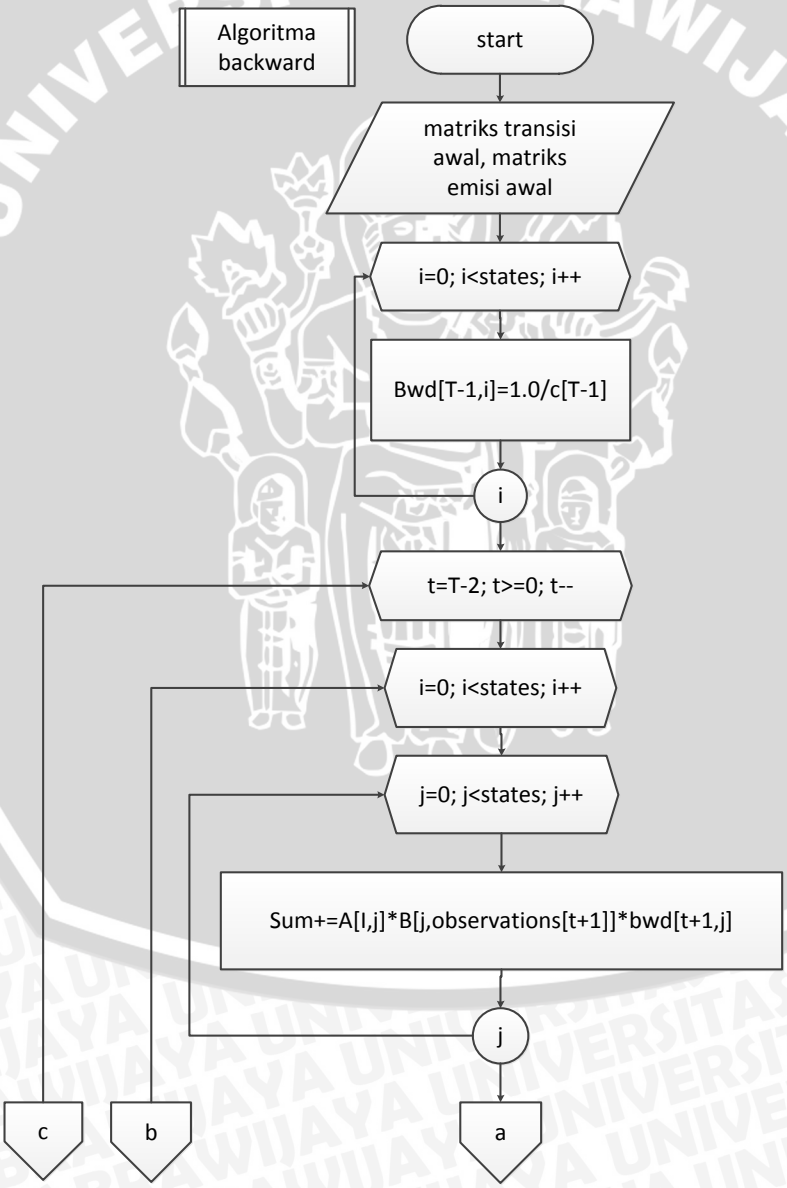
3.3.4.3 Perancangan Algoritma *Backward*

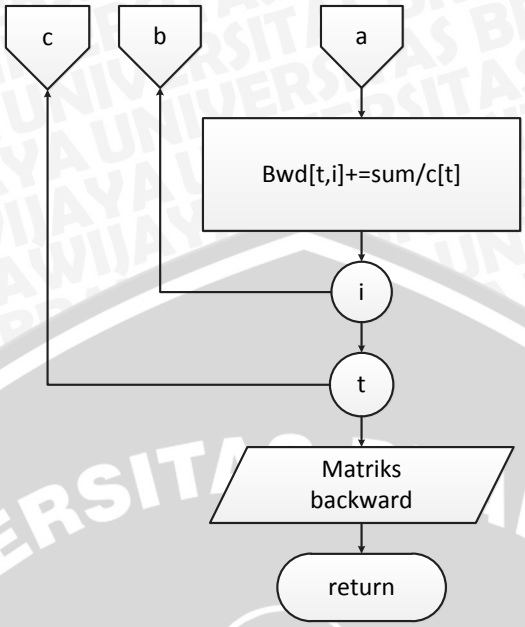
Langkah-langkah dalam proses algoritma *backward* adalah :

1. Matriks transisi awal dan matriks emisi awal sebagai data input
2. Melakukan perulangan untuk *state i.*
3. Menghitung nilai inisialisasi dari algoritma *backward*.
4. Melakukan perulangan untuk panjang deretan / *sequence*.

5. Melakukan perulangan untuk *state i.*
6. Melakukan perulangan untuk *state j.*
7. Menghitung nilai induksi dari algoritma *backward*.
8. Menghitung nilai normalisasi dari nilai induksi algoritma *backward*.
9. Melakukan iterasi hingga selesai dan didapatkan output nilai matriks *backward* yang ternormalisasi.
10. Proses algoritma *backward* selesai.

Diagram alir proses algoritma *backward* dapat digambarkan pada Gambar 3.12 berikut ini :





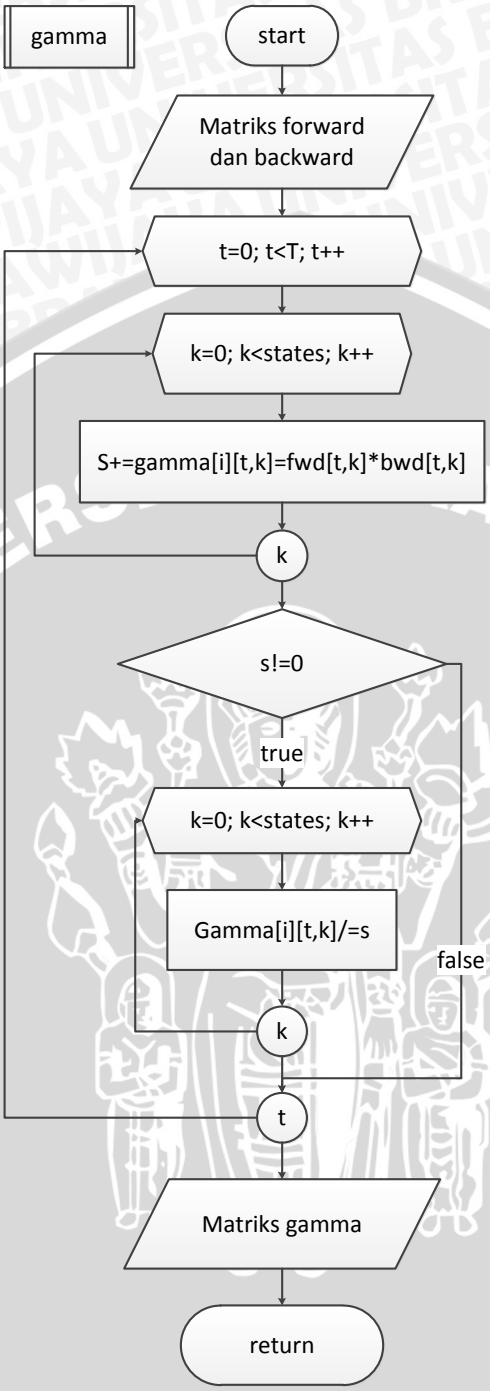
Gambar 3.12 Diagram alir algoritma *backward*

3.3.4.4 Perancangan Matriks *Gamma*

Langkah-langkah pada proses matriks *gamma* adalah :

1. Matriks *forward* dan *backward* sebagai data input.
2. Melakukan perulangan untuk panjang deretan / *sequence*.
3. Melakukan perulangan untuk *state k.*
4. Melakukan perhitungan nilai *gamma.*
5. Jika *s* sama dengan 0 maka ke langkah 9, jika tidak maka ke langkah 6.
6. Melakukan perulangan untuk *state k.*
7. Menghitung nilai normalisasi dari matriks *gamma*.
8. Melakukan iterasi hingga selesai dan didapatkan sebuah output matriks *gamma* yang ternormalisasi.
9. Proses matriks *gamma* selesai.

Diagram alir proses matriks *gamma* dapat digambarkan pada Gambar 3.13 berikut ini :



Gambar 3.13 Diagram alir matriks *gamma*

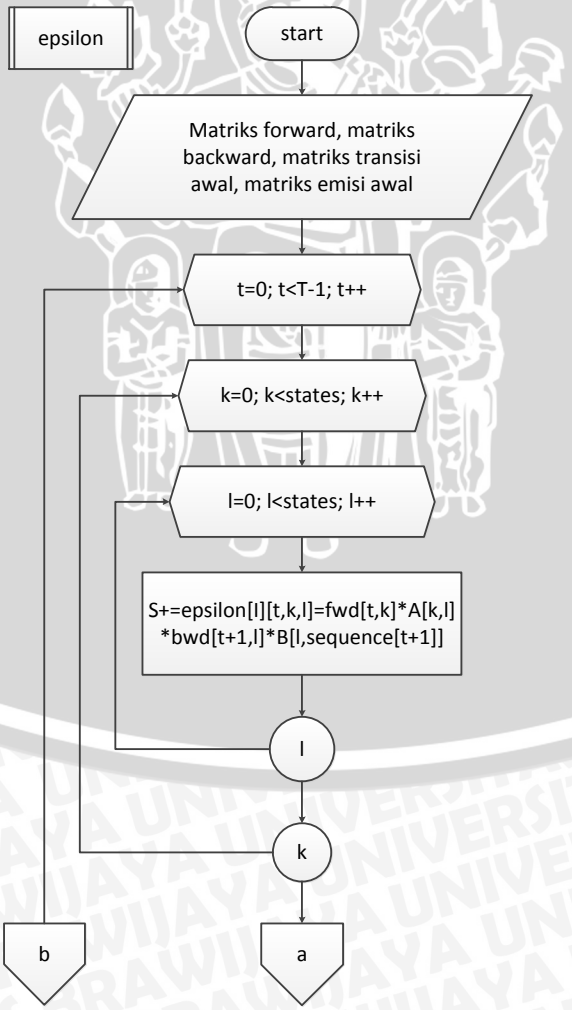
3.3.4.5 Perancangan Matriks *Epsilon*

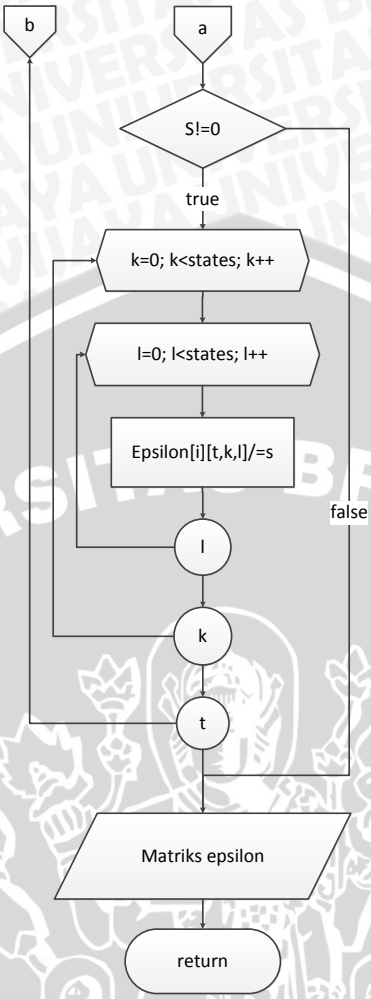
Langkah-langkah pada proses matriks *epsilon* adalah :

1. Matriks *forward*, matriks *backward*, matriks transisi awal, dan matriks emisi awal sebagai data input.

2. Melakukan perulangan untuk panjang deretan / *sequence*.
3. Melakukan perulangan untuk *state k*.
4. Melakukan perulangan untuk *state l*.
5. Melakukan perhitungan matriks *epsilon*.
6. Jika s sama dengan 0 maka ke langkah 11, jika tidak maka ke langkah 7.
7. Melakukan perulangan *state k*.
8. Melakukan perulangan *state l*.
9. Menghitung nilai normalisasi dari matriks *epsilon*.
10. Melakukan iterasi hingga selesai dan didapatkan output nilai matriks *epsilon* yang ternormalisasi.
11. Proses matriks *epsilon* selesai.

Diagram alir proses matriks *epsilon* dapat digambarkan pada Gambar 3.14 berikut ini :





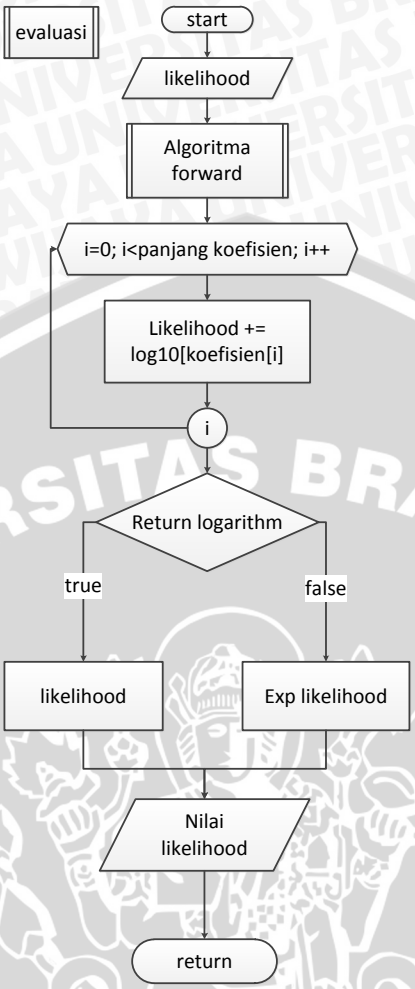
Gambar 3.14 Diagram alir matriks *epsilon*

3.3.4.6 Perancangan Evaluasi

Langkah-langkah pada proses evaluasi adalah:

1. Nilai *likelihood* sebagai data input.
2. Memproses nilai *likelihood* dengan algoritma *forward*.
3. Melakukan perulangan untuk panjang deretan / *sequence*.
4. Menghitung nilai *likelihood*.
5. Jika nilai logaritma tetap maka ambil nilai *likelihood*, jika tidak maka hitung nilai exponen dari nilai *likelihood*.
6. Didapatkan output nilai *likelihood*.
7. Proses evaluasi selesai.

Diagram alir proses evaluasi dapat digambarkan pada Gambar 3.15 berikut ini :



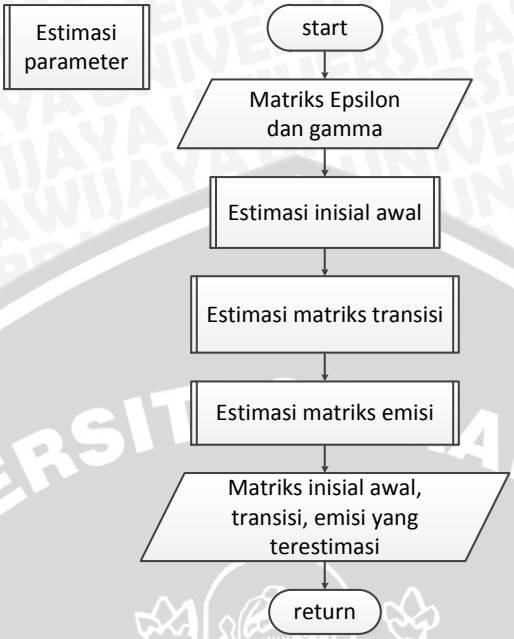
Gambar 3.15 Diagram alir evaluasi

3.3.4.7 Perancangan Re-estimasi Parameter HMM

Langkah-langkah dalam proses re-estimasi parameter HMM adalah :

1. Matriks *epsilon* dan matriks *gamma* sebagai data input.
2. Masuk pada proses re-estimasi inisial awal.
3. Masuk pada proses re-estimasi matriks transisi.
4. Masuk pada proses re-estimasi matriks emisi.
5. Didapatkan output matriks inisial awal, matriks transisi, dan matriks emisi yang telah terestimasi kembali.
6. Proses re-estimasi parameter selesai.

Diagram alir proses re-estimasi parameter dapat digambarkan pada Gambar 3.16 berikut ini :



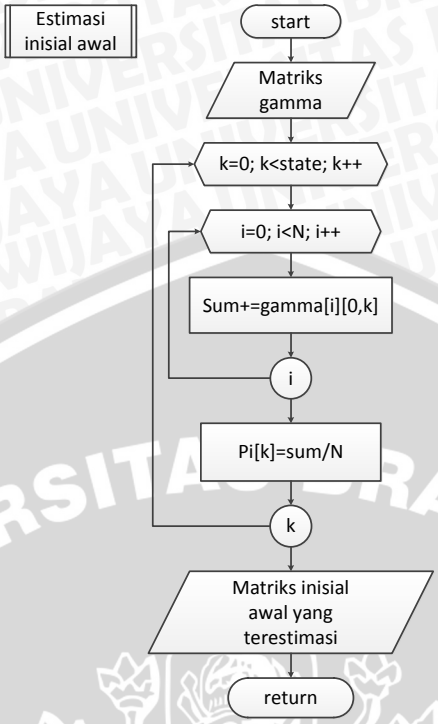
Gambar 3.16 Diagram alir re-estimasi parameter HMM

3.3.4.7.1Perancangan Re-estimasi Matriks Inisial

Langkah-langkah pada proses re-estimasi matriks inisial adalah :

1. Matriks *gamma* sebagai data input.
2. Melakukan perulangan untuk *state k*.
3. Melakukan perulangan untuk jumlah label.
4. Melakukan proses perhitungan matriks inisial dan melakukan iterasi jumlah label hingga selesai.
5. Menghitung nilai normalisasi dari matriks inisial.
6. Melakukan iterasi hingga selesai dan didapatkan nilai matriks inisial yang ternormalisasi.
7. Proses re-estimasi matriks inisial selesai.

Diagram alir proses re-estimasi matriks inisial dapat digambarkan pada Gambar 3.17 berikut ini :



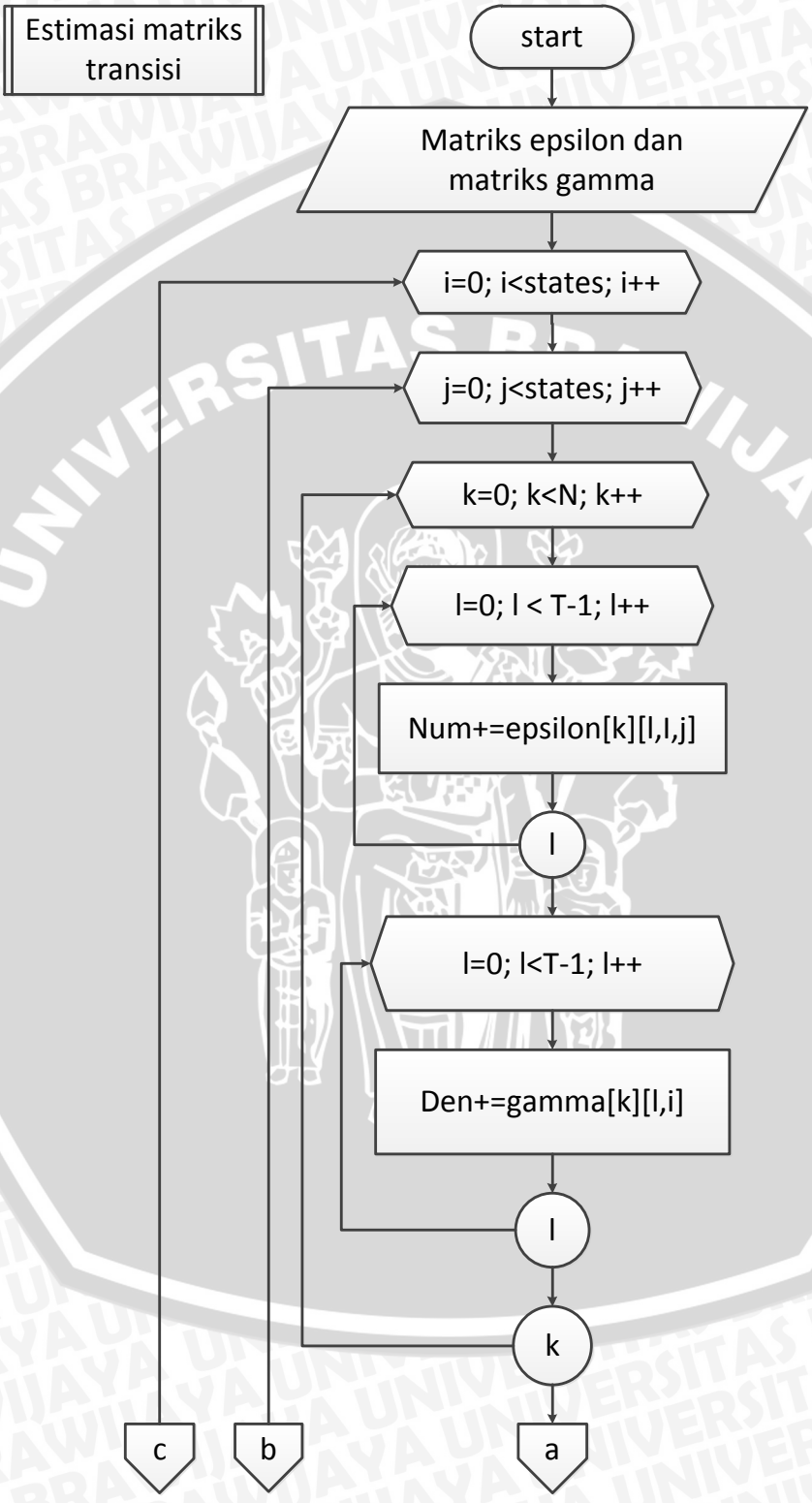
Gambar 3.17 Diagram alir re-estimasi matriks inisial awal

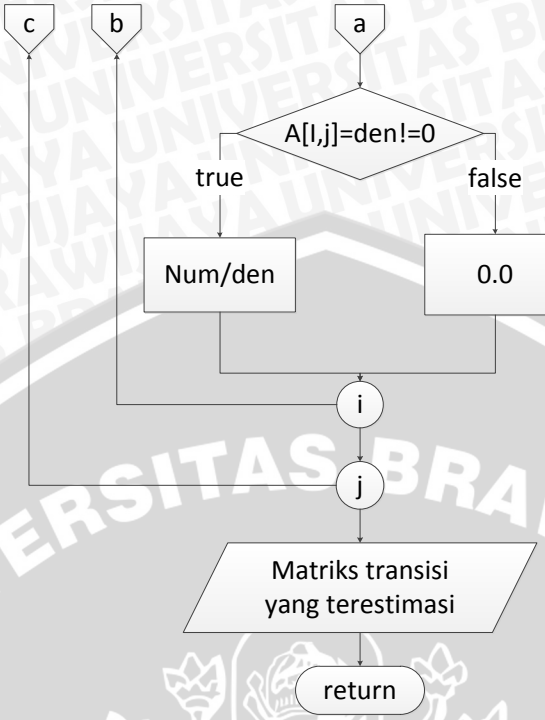
3.3.4.7.2Perancangan Re-estimasi Matriks Transisi

Langkah-langkah pada proses re-estimasi matriks transisi adalah :

1. Matriks *epsilon* dan *gamma* sebagai data input.
2. Melakukan perulangan untuk *state i.*
3. Melakukan perulangan untuk *state j.*
4. Melakukan perulangan untuk jumlah label.
6. Melakukan perulangan untuk panjang deretan / *sequence*.
7. Melakukan perhitungan untuk mendapatkan nilai num.
8. Melakukan perulangan untuk panjang *sequence*.
9. Melakukan perhitungan untuk mendapatkan nilai den.
10. Jika nilai den sama dengan 0 maka matriks transisi bernilai 0, jika tidak maka menghitung matriks transisi.
11. Melakukan iterasi sampai selesai dan didapatkan output matriks transisi yang terestimasi kembali.
12. Proses re-estimasi matriks transisi selesai.

Diagram alir proses re-estimasi matriks transisi dapat digambarkan pada Gambar 3.18 berikut ini :





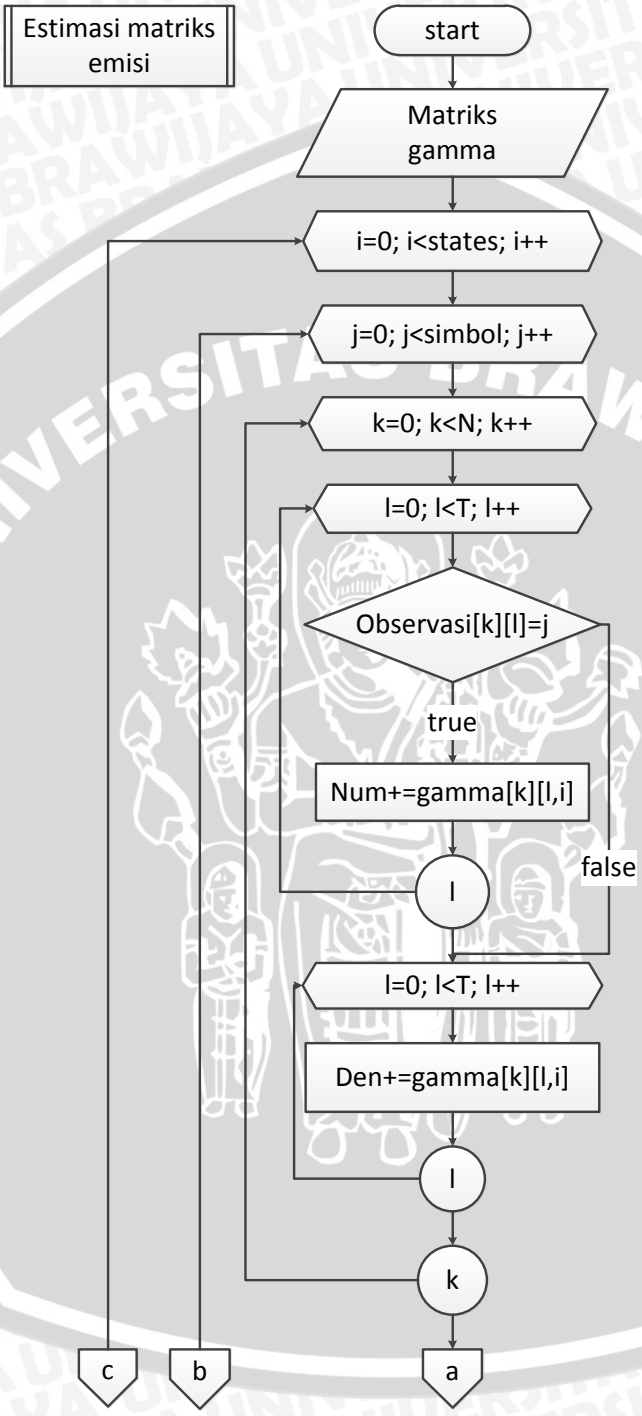
Gambar 3.18 Diagram alir re-estimasi matriks transisi

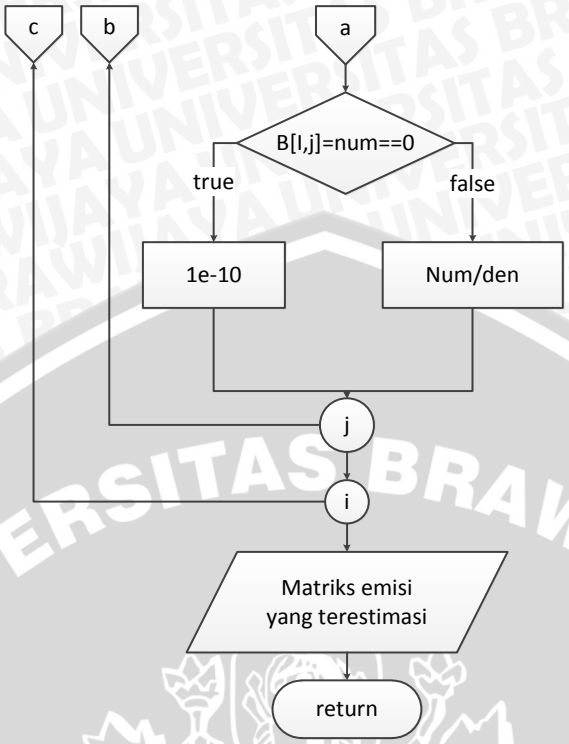
3.3.4.7.3Perancangan Reestimasi Matriks Emisi

Langkah-langkah reestimasi matriks emisi :

1. Matriks *gamma* sebagai data input.
2. Melakukan perulangan untuk *state i.*
3. Melakukan perulangan untuk simbol.
4. Melakukan perulangan untuk jumlah label.
5. Melakukan perulangan untuk panjang deretan / *sequence*.
6. Jika observasi *(k, l)* sama dengan *j* maka hitung nilai num, jika tidak maka ke langkah 7.
7. Melakukan perulangan untuk panjang deretan / *sequence*.
8. Melakukan proses perhitungan nilai den.
9. Jika nilai num == 0 maka nilai matriks transisi adalah 1e-10, jika tidak maka menghitung nilai matriks transisi.
10. Melakukan iterasi hingga selesai dan didapatkan matriks emisi yang terestimasi kembali.
11. Proses re-estimasi matriks emisi selesai.

Diagram alir proses re-estimasi matriks emisi dapat digambarkan pada Gambar 3.19 berikut ini :

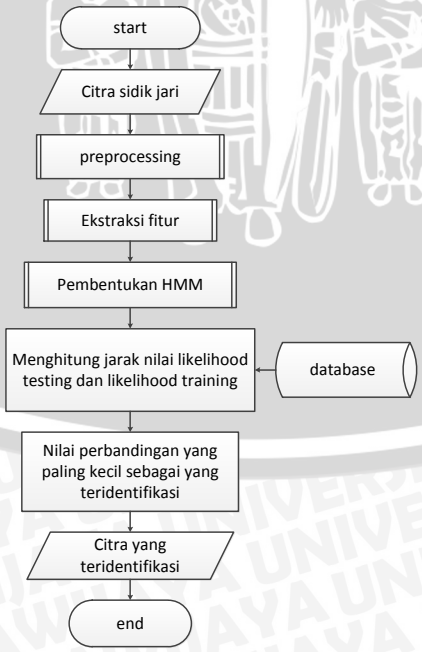




Gambar 3.19 Diagram alir re-estimasi matriks emisi

3.3.5 Perancangan Proses Pengenalan Sidik Jari

Setelah basis data dibuat, maka proses pengenalan sidik jari dilakukan. Gambar 3.3 adalah diagram alir dari proses pengenalan.

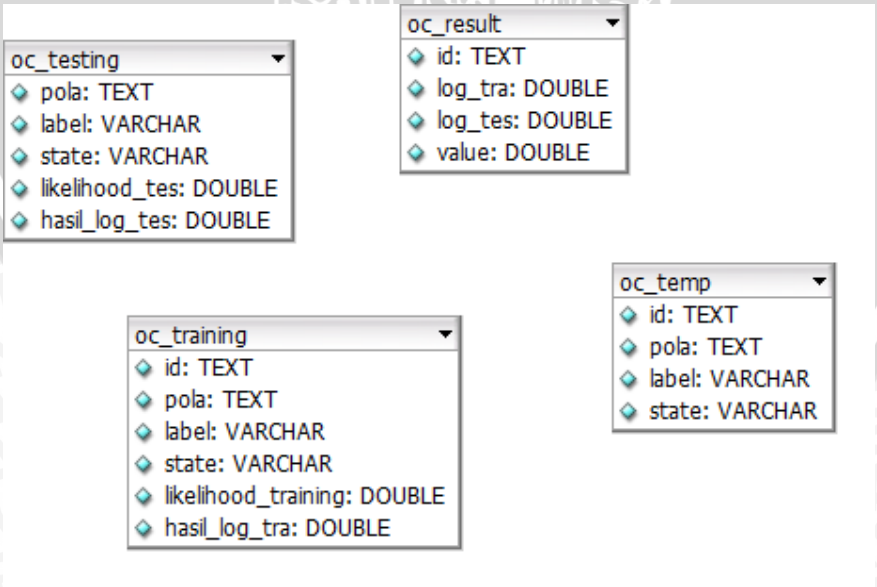


Gambar 3.20 Diagram alir sistem pengenalan sidik jari

Pada proses pengenalan, masukan berupa citra sidik jari akan memasuki proses *preprocessing* yaitu pengubahan citra menjadi citra biner atau disebut binerisasi. Kemudian citra akan dihilangkan *noise*-nya dengan operasi erosi dan tahap terakhir dari *preprocessing* adalah tahap skeletonisasi yang menjadikan citra mempunyai tebal satu piksel. Setelah itu dilakukan ekstraksi fitur terhadap citra yaitu mengambil *bifurcation* pada citra sidik jari. Dari hasil ekstraksi fitur dapat melakukan perhitungan parameter HMM yang meliputi parameter HMM awal, matriks *forward*, matriks *backward*, matriks *gamma*, matriks *epsilon*, proses evaluasi dan proses re-estimasi parameter HMM awal. Setelah didapatkan nilai *likelihood* dari citra *testing* maka akan dihitung jarak diantara citra *testing* dan *training*, citra *training* yang memiliki jarak terkecil dengan citra *testing* akan dijadikan sebagai yang teridentifikasi.

3.3.6 Perancangan Tabel-Tabel Yang Digunakan Dalam Sistem Pengenalan Sidik Jari

Pembuatan tabel-tabel bertujuan untuk memberikan gambaran tabel yang digunakan dan menyimpan data. Struktur tabel yang akan digunakan untuk sistem pengenalan sidik jari pada Gambar 3.21 sebagai berikut :



Gambar 3.21 Tabel-tabel sistem pengenalan sidik jari

3.4 Perhitungan Manual

Pada bab ini akan membahas mengenai contoh perhitungan manual untuk sistem pengenalan sidik jari dengan metode Hidden Markov Model (HMM).

3.4.1 Proses Binerisasi

Pada proses binerisasi mengubah citra *grayscale* menjadi citra biner dengan metode Otsu. Pada Gambar 3.22 adalah contoh citra *grayscale* dengan ukuran 11 x 14 piksel.

255	255	248	140	86	53	63	93	85	110	102
100	98	121	249	174	104	116	77	84	80	100
105	90	71	83	97	130	105	72	115	101	103
110	63	59	75	92	101	136	149	240	62	101
134	128	102	71	62	75	100	103	218	255	255
255	100	131	149	84	74	92	94	117	165	255
255	255	255	255	86	72	81	117	119	160	215
255	255	255	255	255	185	175	91	106	155	199
255	255	255	255	255	255	225	152	149	219	151
240	255	255	255	255	255	255	199	127	101	136
255	255	255	255	255	255	255	190	142	100	117
105	121	202	255	255	255	255	204	144	82	73
97	76	88	122	121	150	182	168	255	174	107
103	87	85	55	69	97	99	70	102	132	205

Gambar 3.22 Citra grayscale

1. Mencari probabilitas dari nilai piksel *i*

   Dari rumus, dapat di hitung probabilitas dari nilai piksel 0 sampai dengan 255. Rumus disusun berdasarkan pada persamaan (2.2) dan contoh manualisasinya adalah pada nilai piksel 55 dan 100 :

   $$P_i = \frac{n_i}{N}$$

   $$P_{55} = \frac{1}{154} = 0{,}006493506$$

   $$P_{100} = \frac{5}{154} = 0{,}032467553$$

2. Mencari nilai rata-rata

Nilai rata-rata digunakan untuk menghitung nilai variansi pada metode otsu didapatkan pada persamaan (2.5) adapun untuk prosesnya sebagai berikut :

$$\mu_T = \sum_{i=1}^{L} i.P_i$$

$$\mu_T = (1 \text{ x } 0) + (2 \text{ x } 0) + \cdots + (53 \text{ x } 0{,}006493506) + (55 \text{ x } 0{,}006493506) + \cdots + (255 \text{ x } \ldots.)$$

$$= 90{,}07142857$$

3. mencari nilai *zerothcm*

Nilai *zerothcm* didapatkan dari persamaan (2.3) dan nilainya akan diambil satu persatu kemudian akan dibandingkan dengan nilai piksel lainnya. Adapun contoh perhitungannya dengan nilai piksel 53 dan 55 :

$$\omega(k) = \sum_{i=1}^{k} P_i$$

$$\omega(53) = 0 + 0 + \cdots + 0{,}006493506 = 0{,}006493506$$

$$\omega(55) = 0 + 0 + \cdots + 0{,}006493506 + 0 + 0{,}006493506$$

$$= 0{,}012987012$$

4. mencari nilai *firstcm*

Sama halnya dengan *firstcm*, nilai *firstcm* didapatkan dari persamaaan (2.4) dan nilainya akan digunakan untuk mencari nilai variansi. Adapun contoh perhitungannya adalah dengan nilai piksel 53 dan 55 :

$$\mu(k) = \sum_{i=1}^{k} i.P_i$$

$$\mu(53) = (1 \text{ x } 0) + (2 \text{ x } 0) + \cdots + (53 \text{ x } 0{,}006493506)$$

$$= 0 + 0 + \cdots + 0{,}344155844 = 0{,}344155844$$

$$\mu(55) = (1 \text{ x } 0) + (2 \text{ x } 0) + \cdots + (53 \text{ x } 0{,}006493506) + (54 \text{ x } 0) + (55 \text{ x } 0{,}006493506)$$

$$= 0 + 0 + \cdots + 0{,}344155844 + 0 + 0{,}357142857$$

$$= 0{,}701298701$$

5. mencari nilai *variance*

Penjelasan mengenai variansi terdapat pada persamaan (2.7) adapun contoh perhitungannya :

$$\sigma_B^2 = \frac{[\mu_T\omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]}$$

$$\sigma_{53}^2 = \frac{[90{,}07142857 * \ 0{,}006493506 \ - 0{,}344155844]^2}{0{,}006493506 \ * \ [1 - 0{,}006493506]}$$

$$= 8{,}982292917$$

$$\sigma_{53}^2 = \frac{[90{,}07142857 * \ 0{,}006493506 \ - 0{,}357142857]^2}{0{,}006493506 \ * \ [1 - 0{,}006493506]}$$

$$= 8{,}039249033$$

6. Membandingkan nilai *variance* antar piksel dan mengambil nilai piksel yang memiliki nilai *variance* yang maksimum untuk dijadikan T

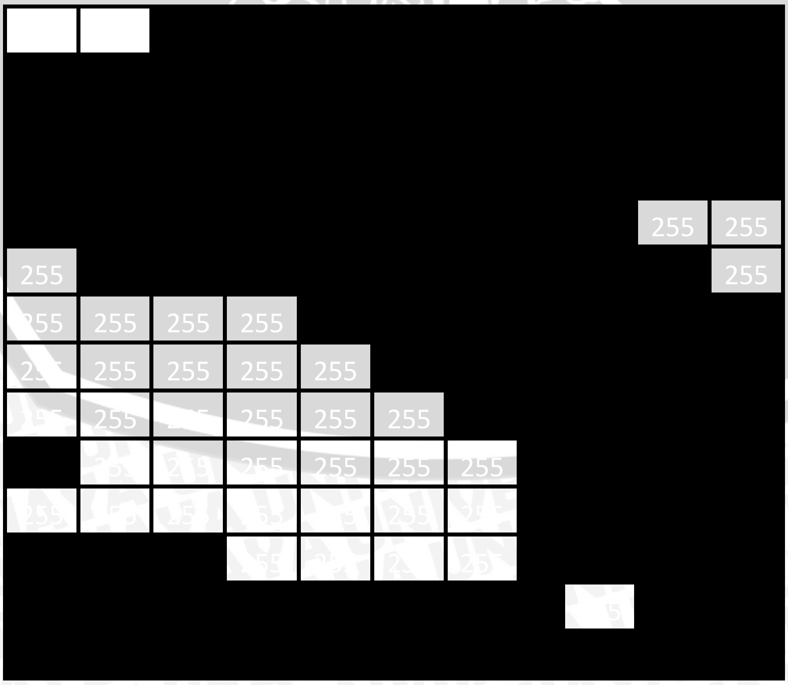
$\sigma_B^2(k^*) = max_{1\le k\le L}\sigma_B^2(k)$

Contoh :

$0 \le \sigma_{53}^2 = 8{,}982292917$ jika iya maka T = 53

Jika tidak maka T = 0

Pada lampiran 1 adalah tabel hasil perhitungan metode Otsu. Dari perhitungan di atas, diperoleh hasil citra biner pada Gambar 3.23 sebagai berikut :



Gambar 3.23 Hasil citra biner dengan metode otsu

3.4.2 Proses Erosi

Dari hasil citra biner pada Gambar 3.23 maka dapat diperbaiki morfologinya dan dihilangkan *noise-noise* yang ada yaitu dengan cara dilakukan erosi. Pada Gambar 3.24 adalah koordinat struktur elemen dan Gambar 3.25 adalah *structuring element* yang akan dipakai dalam proses erosi.

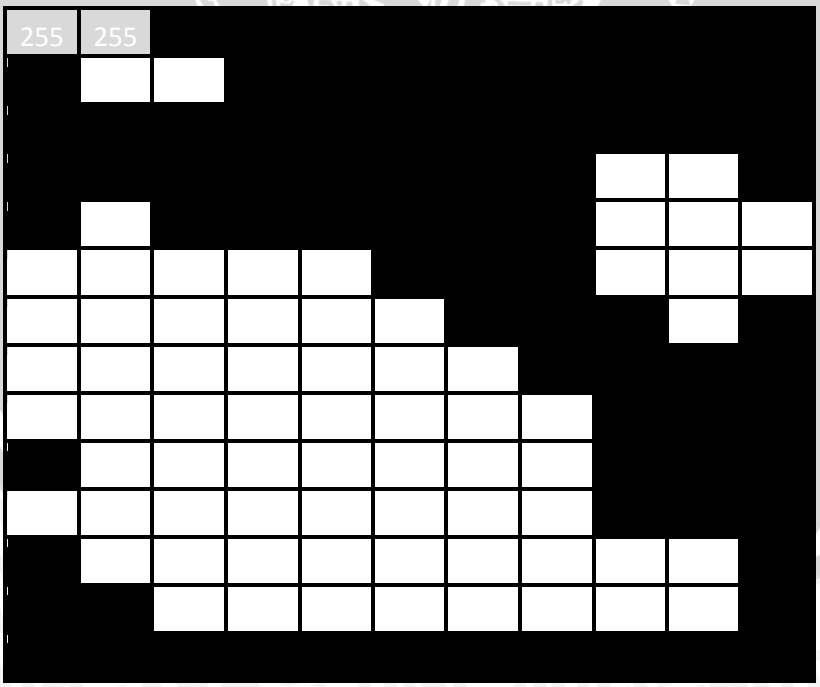
i-1,j-1	i,j-1	i+1,j-1
i-1,j	i,j	i+1,j
i-1,j+1	i,j+1	i+1,j+1

Gambar 3.24 Koordinat struktur elemen

0	0	0
0	0	0
0	0	0

Gambar 3.25 *Structuring element* (*mask*)

Untuk operasi erosi pada citra nilai tengah *(i,j)* diubah menjadi 0 jika semua tetangga terdekatnya memiliki 0, jika salah satu tetangga terdekatnya tidak bernilai 0 maka citra nilai tengah *(i,j)* diubah menjadi 255. Gambar 3.26 adalah hasil dari proses erosi.



Gambar 3.26 Citra hasil erosi

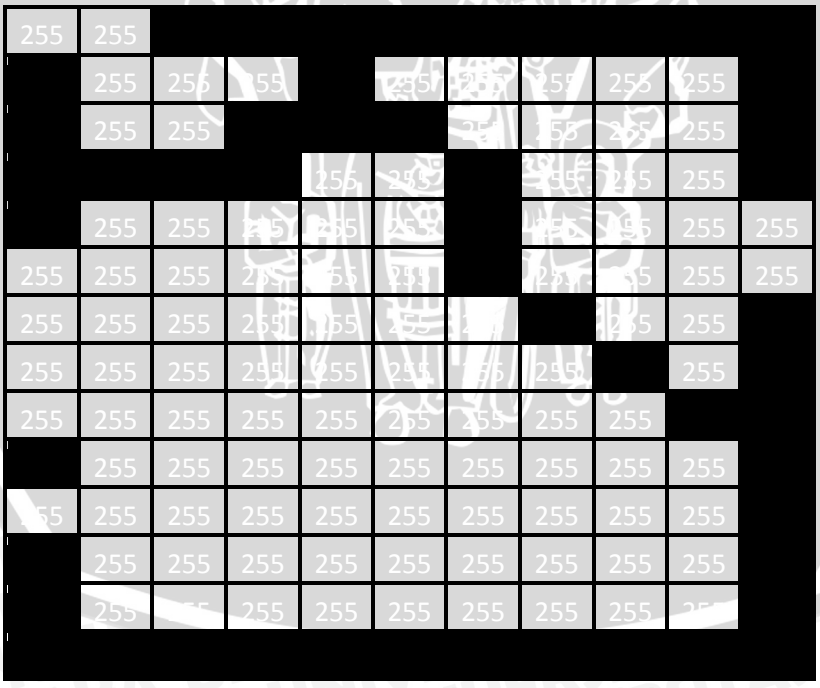
3.4.3 Proses Skeletonisasi

Pada proses skeletonisasi citra akan di jadikan setebal satu piksel dengan ketentuan beikut :

iterasi pertama yaitu jika jumlah tetangga citra *(p2,p3,...,p9)* yang bernilai 0 lebih dari sama dengan 2 dan kurang dari sama dengan 6, jika perpindahan citra 255 ke 0 dari *p2,p3,...,p9* harus berjumlah 1, *p2*p4*p6 =1*, dan *p4*p6*p8=1* maka piksel tersebut ditandai dan dihapus.

Iterasi kedua untuk langkah pertama adalah sama dengan langkah pertama yaitu jika jumlah tetangga citra *(p2,p3,...,p9)* yang bernilai 0 lebih dari sama dengan 2 dan kurang dari sama dengan 6, jika perpindahan citra 255 ke 0 dari p2,p3,…,p9 harus berjumlah 1, *p2*p4*p8 =1, p2*p6*p8=1* maka piksel tersebut ditandai dan dihapus.

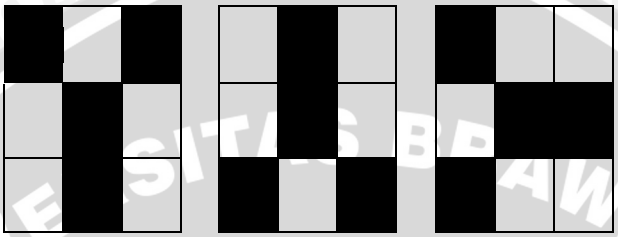
Dua iterasi ini dilakukan secara terus menerus sampai citra tidak mengalami perubahan struktur. Adapun contoh proses skeletonisasi dengan menggunakan metode Zhang-Suen terdapat pada gambar 3.27 sebagai berikut :



Gambar 3.27 Citra yang mengalami proses skeletonisasi

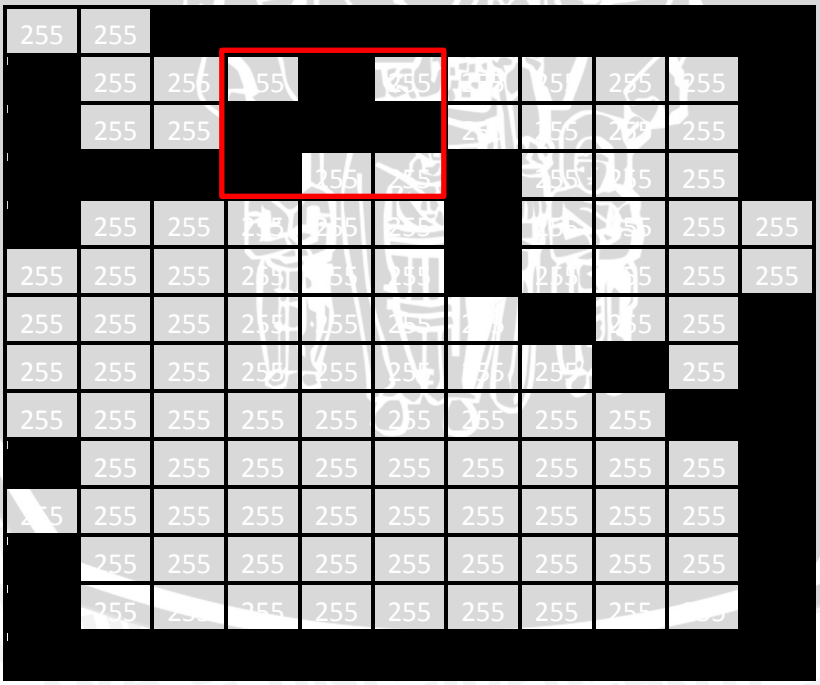
3.4.4 Proses Ekstraksi Fitur

Pada proses ini, semua piksel akan ditelusuri dengan matriks *bifurcation* yang berukuran 3 x 3 dan hasilnya akan disimpan didalam *database*. *Template bifurcation* telah di buat sebanyak 24 buah. Pada Gambar 3.28 adalah sebagian contoh dari *bifurcation*.



Gambar 3.28 Beberapa contoh *bifurcation*

Hasil dari proses ekstraksi fitur seperti pada gambar 3.29 berikut ini.



Gambar 3.29 Citra yang mengalami proses ekstraksi fitur

3.4.5 Proses Parameter HMM

Pada proses ini citra yang sudah di ekstraksi fitur urutan dari *bifurcation*-nya akan disimpan didatabase. Selain pola / sequence / deretan dari *bifurcation* yang akan disimpan adalah nomer id citra, label citra, dan *state* citra. *State* citra yang dipakai adalah 9 karena setiap *bifurcation* dapat berpindah pada matriks 3 x 3 sehingga ada 9 kotak yang dapat ditempati. Label yang digunakan ada 5 yaitu A, B, C, D, E semua label mewakili id citra tersebut. Berikut ini contoh dari proses parameter HMM dengan jumlah *state* 1. Pada Tabel 3.2 adalah salah satu contoh data *training* dan Tabel 3.3 adalah tabel deretan observasi yang ada pada data *training*.

Tabel 3.1 Salah satu contoh data *training*

Pola	Label	state	Id
0,3,10,15,20,17	A	1	012_1_1

diket :

pola	=	menunjukkan nama simbol dari *bifurcation*
label	=	*A*
states	=	1
simbol	=	24
A (matriks transisi)	=	1/state
B (matriks emisi)	=	1/simbol
Pi (matriks inisial awal)	=	array ke - 0 = 1
N (jumlah label)	=	1
T	=	6

Tabel 3.2 Tabel observasi

0	0
1	3
2	10
3	15
4	20
5	17

Untuk melihat nilai matriks HMM yang dibentuk ada pada lampiran 2..

1. Menghitung nilai matriks *forward* untuk *state* HMM dengan menggunakan algoritma *forward*

Dalam menentukan nilai matriks *forward* maka proses akan dibagi menjadi dua yaitu proses inisialisasi dan proses induksi. Contoh perhitungannya adalah sebagai berikut :

a. Inisialisasi

*c[0] += fwd[0,i] = pi[i] * B[0,observations[0]]*
fwd[0,i] = fwd[0,i] / c[0]

fwd[0,0] = *1 x B[0,0]*
= *1 x 0,041667*
= *0,041667*

c[0] = *0 + 0,041667* = *0,041667*

Setelah ditemukan *c[0]* maka proses selanjutnya adalah proses normalisasi dari inisialisasi. Berikut ini proses normalisasinya :

fwd[0,0] = *0,041667 : 0,041667* = *1*

Untuk melihat tabel hasil dari inisialisasi matriks *forward* ada pada lampiran 3.

b. Induksi

p = B[i,observations[t]]
*sum += fwd[t-1,j] * A[i,j]*
*fwd[t,i] = sum * p*
c[t] += fwd [t,i]
fwd[t,i] = fwd[t,i] / c[t]

p = *B[0,observasi[1]]* = *0,041667*
sum = *fwd[0,0] x A[0,0]*
= *1 x 1* = *1*
Sum = *0 + 1* = *1*
fwd[0,1] = *1 x 0,041667* = *0,041667*
c[0] = *0,041667*
c[0] = *0 + 0,041667* = *0,041667*

Setelah ditemukan nilai induksi dari matriks *forward* maka akan dinormalisasi sebagai berikut :

fwd[0,1] = *0,041667 : 0,041667* = *1*

Proses-proses selanjutnya dapat dilihat dalam tabel yang ada pada lampiran 3.

2. Menghitung nilai matriks *backward* untuk *state* HMM dengan menggunakan algoritma *backward*

Dalam menentukan nilai matriks *backward* maka perhitungan dimulai urutan array yang belakang dan terdapat dua proses yaitu inisialisasi dan induksi. Adapun contoh perhitungannya adalah sebagai berikut :

a. Inisialisasi

$$bwd[T-1,i] = 1 / c[T-1]$$

$bwd[5,0] = 1 : c[5]$

$= 1 : 0{,}041667 = 24$

Untuk melihat tabel hasil dari inisialisasi matriks *backward* ada pada lampiran 4.

b. Induksi

$$sum += A[i,j] * B[j,observations[t+1]] * bwd[t+1,j]$$
$$bwd[t,i] += sum / c[t]$$

$sum = A[0,0] \times B[0,observations[5]] \times bwd[5,0]$

$= 1 \times B[0,17] \times bwd[5,0]$

$= 1 \times 0{,}041667 \times 24 = 1$

$sum = 0 + 1 = 1$

$bwd[4,0] = 1 : c[4]$

$= 1 : 0{,}041667 = 24$

Untuk melihat hasil dari perhitungan induksi matriks *backward* terdapat pada lampiran 4.

3. Menentukan dan menghitung nilai matriks *gamma*

Matriks *gamma* digunakan pada waktu re-estimasi parameter HMM. Adapun contoh perhitungan matriks *gamma* adalah sebagai berikut :

$$s += gamma[i][t,k] = fwd[t,k] * bwd[t,k]$$
$$gamma[i][t,k] /= s$$

$gamma[0][0,0] = fwd[0,0] \times bwd[0,0]$

$= 1 \times 24 = 24$

$s = 0 + gamma[0][0,0]$

$= 0 + 24 = 24$

Setelah ditemukan nilai dari matriks gamma maka akan dinormalisasi sebagai berikut.

gamma[0][0,0] = *gamma[0][0,0] : 24*

= *24 : 24* = *1*

Untuk proses selanjutnya dapat dilihat pada lampiran 5.

4. Menentukan dan menghitung nilai matriks *epsilon*

Sama halnya dengan matriks *gamma*, matriks *epsilon* juga digunakan pada waktu re-estimasi parameter HMM. berikut ini adalah contoh perhitungan matriks *epsilon*.

*s += epsilon[i][t,k,l] = fwd[t,k] * A[k,l] * bwd[t+1,l] * B[l,sequence[t+1]]*
epsilon[i][t,k,l] /= s

epsilon[0][0,0,0] = *fwd[0,0] x A[0,0] x bwd[1,0] x B[0,sequence[1]]*

= *1 x 1 x 24 x B[0,3]*

= *1 x 1 x 24 x 0,041667*

= *1*

s = *0 + epsilon[0][0,0,0]*

= *0 + 1* = *1*

Setelah menemukan nilai *epsilon* maka nilai tersebut akan dinormalisasi sebagai berikut.

Epsilon[0][0,0,0] = *epsilon[0][0,0,0] : 1*

= *1 : 1* = *1*

Untuk hasil perhitungan matriks epsilon ada pada lampiran 6.

5. Menentukan dan menghitung nilai *log* dari *sequence* yang ada

newLikelihood += math.log10(scaling[t])

oldlikelihood = *double.minvalue* = *-1.79769e+308*

newlikelihood = *log10(scaling[0])*

= *log10(0,041667)* = *-1,38021*

newlikelihood = *log10(scaling[0]) + ... + log10(scaling[5])*

= *-1,38021 + -1,38021 + ... + -1,38021*

= *-8,28127*

Untuk melihat hasil dari perhitungan *likelihood* baru terdapat pada lampiran 7.

6. Menetapkan nilai *newlikelihood*

newLikelihood = newLikelihood / 1

newlikelihood = *newlikelihood*

= *-8,28127*

7. Cek konvergensi

Mengecek jumlah iterasi sesuai dengan iterasi yang diberikan atau tidak, mengecek *oldlikelihood* sama dengan *newlikelihood* atau *newlikelihood* memiliki *error* yang kecil terhadap *oldlikelihood*. Jika semua benar maka proses dihentikan jika tidak maka akan di reestimasi kembali parameter HMM.

8. Re-estimasi matriks inisial

Matriks inisial digunakan untuk menghitung nilai matriks *forward* setelah parameter HMM terestimasi kembali. Adapun contoh perhitungannya adalah sebagai berikut :

```
sum += gamma[i][0,k]
pi[k] = sum / N
```

sum = *gamma[0][0,0]*

= *1*

sum = *0 + gamma[0][0,0]*

= *0 + 1* = *1*

pi[0] = *1 / 1* = *1*

Untuk hasil re-estimasi matriks inisial terdapat pada lampiran 8.

9. Re-estimasi matriks transisi

Sama halnya dengan matriks inisial, matriks transisi juga digunakan untuk menghitung matriks *forward* dan *backward* setelah parameter HMM terestimasi. Adapun contoh perhitungannya adalah sebagai berikut :

```
num += epsilon[k][l,i,j]
den += gamma[k][l,i]
A[i,j] = (den !=0) ? Num/den : 0
```

num = *epsilon[0][0,0,0]*

= *1*

Untuk perhitungan *num* selanjutnya sama seperti cara diatas dan didapatkan hasil 1.

num = *epsilon[0][0,0,0] + epsilon[0][1,0,0] +*
epsilon[0][2,0,0] + epsilon[0][3,0,0] +
epsilon[0][4,0,0]

$= \quad 1 + 1 + 1 + 1 + 1 \quad = \quad 5$

$den \quad = \quad gamma[0][0,0]$

$= \quad 1$

Untuk perhitungan *den* selanjutnya sama seperti cara diatas dan didapatkan hasil 1.

$den \quad = \quad gamma[0][0,0] + gamma[0][1,0] + gamma[0][2,0] + gamma[0][3,0] + gamma[0][4,0]$

$= \quad 1 + 1 + 1 + 1 + 1 \quad = \quad 5$

$A[0,0] \quad = \quad den != 0$

$= \quad num : den$

$= \quad 5 : 5 \quad = \quad 1$

Untuk perhitungan *num* dan *den* selanjutnya dapat dilihat pada lampiran 8.

10. Re-estimasi matriks emisi

Sama halnya seperti matriks inisial dan matriks transisi, matriks emisi juga digunakan untuk menghitung nilai matriks *forward* dan *backward* setelah parameter HMM terestimasi kembali. Sebelum menghitung nilai *num* maka perlu diperhatikan nilai dari *observasi[k][l] = j*, dimana *j* adalah simbol *bifurcation* yaitu 0 sampai dengan 23 berikut ini adalah nilai observasinya.

observation[0][0] = 0
observation[0][1] = 3
observation[0][2] = 10
observation[0][3] = 15
observation[0][4] = 20
observation[0][5] = 17

jadi nilai matriks emisi bukan *1e-10* pada waktu *(0,0), (0,3), (0,10), (0,15), (0,20), (0,17)* sedangkan selain itu nilainya adalah *1e-10*. Untuk itu maka dihitung nilai matriks yang tidak bernilai *1e-10* dengan perhitungan berikut ini :

```
num += gamma[k][l,i]
den += gamma[k][l,i]
B[i,j] = (num==0) ? 1E-10 :
         num/den
```

num = *gamma[0][0,0]*

= *1*

num = *gamma[0][0,0]*

= *1*

Untuk iterasi selanjutnya jika salah maka akan dihitung nilai den.

den = *gamma[0][1,0]*

= *1*

den = *gamma[0][1,0] + gamma[0][2,0] + gamma[0][3,0] + gamma[0][4,0] + gamma[0][5,0]*

= *1 + 1 + 1 + 1 + 1* = *5*

B[0,0] = *num != 0*

= *1 : 5* = *0,2*

Untuk perhitungan dan hasil matriks emisi terdapat pada lampiran 8.

Setelah didapatkan parameter HMM yang sudah terestimasi kembali maka akan dilanjutkan dengan menghitung dan menentukan nilai matriks *forward*, matriks *backward*, matriks *epsilon*, dan matriks *gamma* serta cek konvergensi. Semua matriks menggunakan cara yang sama seperti cara yang ada diatas dan untuk hasilnya terdapat pada lampiran 8.

11. Evaluasi

Setelah parameter HMM sudah konvergen maka langkah terakhir adalah mengevaluasi hasil dari *likelihood* yang sudah didapatkan pada proses perhitungan sebelumnya. Adapun contoh perhitungannya adalah sebagai berikut :

```
likelihood += math.log10(coefficient[i])
return logarithm ? Likelihood : math.exp(likelihood)
```

likelihood = *log10(coefficient[0])*

= *log10(0,2)* = *-0,69897*

likelihood = *log10(coefficient[0]) + log10(coefficient[1]) + log10(coefficient[2]) + log10(coefficient[3]) + log10(coefficient[4]) + log10(coefficient[5])*

= *-0,69897 + -0,69897 + -0,69897 + -0,69897 + -0,69897 + -0,69897*

= *-4,19382*

Karena nilai tidak kembali ke nilai logaritma maka *likelihood* yang didapat akan di exponenkan sehingga hasilnya adalah sebagai berikut.

Likelihood = *exp(-4,19382)*

= *0,015089*

3.4.6 Proses Pengenalan

Diasumsikan setiap label terdapat 2 data *training* yang sama sehingga 2 x 5 label adalah 10 yang sudah memiliki nilai *likelihood* masing-masing dan hitung jarak kedekatannya dengan data *testing*. Data *training* yang memiliki jarak kedekatan terkecil dengan *testing* akan dijadikan hasil pengenalannya dengan mengambil id dari citra sidik jari data *training*. Pada Tabel 3.4 adalah contoh dari data *training* dan Tabel 3.5 adalah contoh dari data *testing*. Adapun contoh perhitungan pada proses ini sebagai berikut :

Tabel 3.3 Tabel data *training*

No	Pola	Label	State	Id	Hasil_log_tra
1	0,3,10,15,20,17	A	1	012_3_1	-4,91382
2	1,23,20,1,1,2,3,4,5	A	1	012_3_5	-1,94895
3	9,8,3,4,12,11,10,2	B	1	013_4_4	-3,18271
4	23,2,2,2,5,6,9	B	1	013_8_2	-4,12919
5	3,4,10,11,15,17,22,21	C	1	022_3_8	-8,75738
6	23,4,5,9,0,0,1,2,3	C	1	022_2_4	-9,92929
7	8,8,9,10,11,23,22	D	1	027_8_5	-5,19289
8	4,5,8,10,3,8,9,10,11	D	1	027_7_7	-8,83743
9	12,13,10,19,10,22	E	1	076_5_6	-2,29383
10	3,5,7,9,1,11	E	1	076_6_1	-7,39289

Tabel 3.4 Tabel data *testing*

Pola	label	state	Hasil_log_tes
19,3,0,2,8,3,8	E	1	-6,82392

Dengan menggunakan *manhattan distance* akan didapatkan jarak terkecil antara data *training* dan data *testing*.

data testing dengan data training ke-1

|-6,82392 – (-4,91382)| = *1,9101*

data testing dengan data training ke-2

|-6,82392 – (-1,94895)| = 4,87497

data testing dengan data training ke-3

|-6,82392 – (-3,18271)| = 3,64121

data testing dengan data training ke-4

|-6,82392 – (-4,12919)| = 2,69473

data testing dengan data training ke-5

|-6,82392 – (-8,75738)| = 1,93346

data testing dengan data training ke-6

|-6,82392 – (-9,92929)| = 1,93346

data testing dengan data training ke-7

|-6,82392 – (-5,19289)| = 1,63103

data testing dengan data training ke-8

|-6,82392 – (-8,83743)| = 2,01351

data testing dengan data training ke-9

|-6,82392 – (-2,29383)| = 4,53009

data testing dengan data training ke-10

|-6,82392 – (-7,39289)| = 0,56897

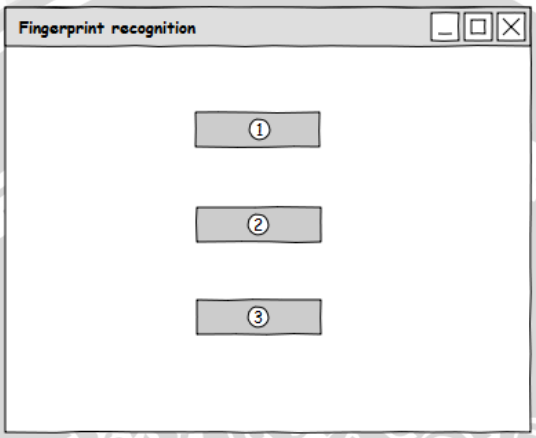
Dari hasil perhitungan diatas maka didapatkan nilai terkecil *0,56897* dengan begitu maka citra diatas dikenali sebagai citra kelas *E* dan dimiliki oleh orang dengan id *076*.

3.5 Perancangan *Interface*

Tahapan implementasi algoritma merupakan tahap implementasi dari metode atau algoritma yang telah dirancang ke dalam suatu bahasa pemrograman. Tahapan ini akan menghasilkan suatu program aplikasi sebagai media representative terhadap hasil dari metode yang diusulkan. Terdapat dua hal dalam tahapan ini yaitu pembuatan kode program yang diusulkan dan pembuatan *interface* program sebagai sarana interaksi sistem dengan pengguna. Implementasi dilakukan dengan bahasa pemrograman C#, sistem managemen basis data menggunakan MySQL, dan pembuatan perancangan *interface* menggunakan

aplikasi yang bernama pencil. Berikut ini perancangan *interface* yang akan digunakan dalam sistem pengenalan sidik jari.

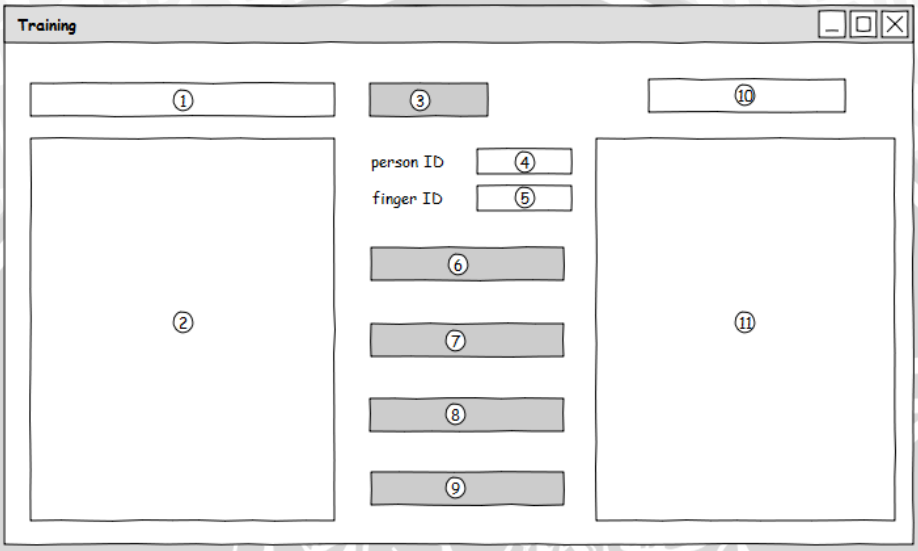
Pada Gambar 3.30 terdapat 3 tombol yaitu tombol *training* (1), tombol *testing* (2), dan tombol *exit* (3). Fungsi tombol *training* adalah untuk menampilkan program *training*. tombol *testing* untuk menampilkan program *testing*. tombol *exit* digunakan untuk keluar dari program pengenalan sidik jari.



Gambar 3.30. Tampilan awal program

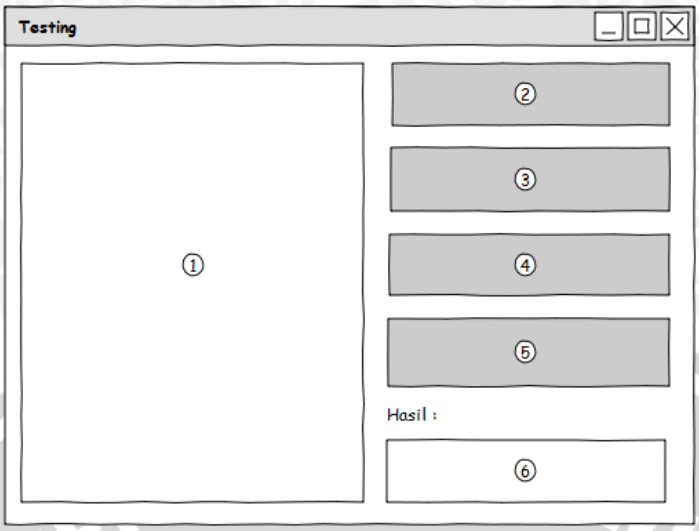
Pada Gambar 3.31 terdapat tombol *open*(3) yang berfungsi untuk membuka citra sidik jari yang akan di t*raining*. Tombol *preprocessing*(6) digunakan untuk mengubah citra sidik jari menjadi citra biner, memperbaiki morfologi citra, dan menjadikan citra sidik jari menjadi citra dengan ketebalan satu piksel. Tombol ekstraksi fitur (7) digunakan untuk mencari ekstraksi fitur dalam citra sidik jari yang terdiri dari *bifurcation* dan *core point*. Tombol parameter HMM (8) digunakan untuk membentuk parameter HMM dari citra sidik jari yang diperoleh selama proses ekstraksi fitur dan nilainya akan dimasukkan ke dalam *database*. Tombol perhitungan nilai observasi (9) digunakan untuk menghitung nilai observasi dari citra sidik jari setelah parameter HMM terbentuk dan nilainya akan dimasukkan ke dalam *database*. Terdapat 4 *textfield* pada program *training*, *textfield* (1) dari alamat gambar saat tombol *open* ditekan maka secara otomatis alamat gambar akan muncul didalam *textfield*, *textfield person ID* (4) dan *Finger ID* (5) digunakan untuk menyimpan *person ID* dan *finger ID* sidik jari *training* dan akan dimasukkan ke dalam *database*.

Textfield status (10) untuk mengetahui apakah proses yang berjalan sudah sukses atau belum jika sudah sukses maka akan muncul tulisan "success" sedangkan jika belum maka akan kosong. Terdapat 2 buah *box* gambar, *box* (2) digunakan untuk menampilkan gambar asli sedangkan *box* (11) digunakan untuk menampilkan gambar hasil proses *preprocessing* dan ekstraksi fitur.



Gambar 3.31. Tampilan program *training*

Pada Gambar 3.32 *textfield* (6) digunakan untuk menampilkan hasil dari pengenalan citra sidik jari. Box (1) untuk menampilkan citra asli dan citra setelah melalui proses *preprocessing* dan ekstraksi fitur. Tombol *open* (2) digunakan untuk membuka fie citra sidik jari yang ingin diuji. Tombol *preprocessing* (3) mengolah citra sidik jari menjadi citra biner, memperbaiki morfologi citra, dan menjadikan citra setebal satu piksel. Tombol ekstraksi fitur (4) digunakan untuk mencari *bifurcation* dan *core point* dari citra yang diuji. Tombol pengenalan (5) digunakan untuk mengenali citra yang diuji dan membandingkan parameter HMM dan nilai observasi dengan citra *training* yang ada didalam *database*.



Gambar 3.32. Tampilan program *testing*

3.6 Skenario Pengujian dan Analisis

Setelah tahapan implementasi algoritma selesai, maka tahapan penelitian ini dilanjutkan dengan melakukan pengujian terhadap sistem yang telah dibuat untuk melihat hasil dari implementasi algoritma. Pengujian dimaksudkan untuk mengetahui apakah penelitian yang dilakukan telah dapat memenuhi tujuan penelitian sebagaimana yang telah direncanakan. Sebagaimana disebutkan diatas, tujuan dari penelitian ini adalah untuk mengimplementasikan *hidden markov model* dalam melakukan pengenalan terhadap citra sidik jari Verifinger_Sample_DB. Analisis dilakukan untuk melihat apakah metode *hidden markov model* yang diusulkan menghasilkan hasil yang baik. Analisis dilakukan berdasarkan hasil perhitungan *hidden markov model* secara kuantitatif. Hasilnya dikatakan baik jika dapat mengenali sidik jari tersebut dengan tepat. Adapun analisis yang akan dilakukan meliputi :

a. Analisis pengaruh jumlah data training terhadap tingkat akurasi proses pengenalan sidik jari

   Perancangan tabel hasil analisis pengaruh jumlah data training terhadap tingkat akurasi yang akan didapat dari sistem pengenalan sidik jari adalah sebagai berikut :

   Tabel 3.5 Tabel Hasil uji coba sistem dengan data training *n*

no	Person_id	Database training	
		Dikenali	Tidak dikenali
1.			
2.			
3.			
Total			