PENGENALAN SIDIK JARI (*FINGERPRINT RECOGNITION*) DENGAN METODE *HIDDEN MARKOV MODEL* (HMM)

SKRIPSI

Laboratorium Komputasi Cerdas Dan Visualisasi

Untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

ERVIN YOHANNES

NIM. 0910680055

PROGRAM STUDI TEKNIK INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

PENGENALAN SIDIK JARI (*FINGERPRINT RECOGNITION*) DENGAN METODE *HIDDEN MARKOV MODEL* (HMM)

SKRIPSI

Laboratorium Komputasi Cerdas Dan Visualisasi

Untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh:

ERVIN YOHANNES

NIM. 0910680055

Menyetujui :

Pembimbing I	**Pembimbing II**
Edy Santoso, S.Si., M.Kom.	**Ahmad Afif Supianto, S.Si., M.Kom.**
NIP. 19740414 200312 1 004	**NIK. 820623 16 1 1 0425**

LEMBAR PENGESAHAN

PENGENALAN SIDIK JARI (*FINGERPRINT RECOGNITION*) DENGAN METODE *HIDDEN MARKOV MODEL* (HMM)

SKRIPSI

Laboratorium Komputasi Cerdas Dan Visualisasi

Untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun oleh:

ERVIN YOHANNES

NIM. 0910680055

Skripsi ini telah diuji dan dinyatakan lulus pada

tanggal 12 Juli 2013

Penguji I	**Penguji II**
Candra Dewi, S.Kom., M.Sc.	**Dian Eka Ratnawati, S.Si., M.Kom.**
NIP. 19771114 200312 2 001	**NIP. 19730619 200212 2 001**

Penguji III

Rekyan Regasari MP, S.T., M.T.
NIK. 770414 06 1 2 0253

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP.19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Juli 2013

Mahasiswa,

Ervin Yohannes

NIM. 0910680055

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Mahaesa karena berkat rahmat-Nya, penulis dapat menyelesaikan proposal skripsi yang berjudul "**Pengenalan Sidik Jari (*Fingerprint Recognition*) Dengan Metode *Hidden Markov Model* (HMM)**".

Penyusunan skripsi ini bertujuan untuk memenuhi sebagian persyaratan memperoleh gelar sarjana komputer. Tak lupa penulis mengucapkan rasa terima kasih kepada :

1. Edy Santoso, S.Si., M.Kom., dan Ahmad Afif Supianto, S.Si., M.Kom., selaku dosen pembimbing selama pelaksanaan skripsi.
2. Ir. Sutrisno, M.T, Ir. Heru Nurwasito, M.Kom., Himawat Aryadita, S.T, M.Sc., dan Edy Santoso, S.Si., M.Kom., selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Ibunda Misti, Ayahanda Petrus Ari Bowo, dan seluruh keluarga atas segenap dukungan dan kasih sayang yang telah diberikan.
4. Drs. Marji, M.T dan Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
5. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
6. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
7. Bekti Widyaningsih, Hendro Pramudyo Saputro, Hanifa Vidya Rizanti, Nina Amalia Dewi, Anisa Aini Arifin, Milani Winangga, Winda Ayu Irianto, Fauziah Mayasari Iskandar, Sufia Adha Putri, Mamluatul Hani'ah, Putu Arya Kurnia, Mohammad Ilham Ubaidillah, Aggy Kubelaborbir dan teman-teman KCV maupun teman-teman TIF 09 yang telah memberikan bantuan baik moril maupun spiritual.

8. Seluruh pihak yang telah membantu kelancaran penulisan skripsi yang tidak dapat saya sebutkan satu persatu.

Yang telah membantu dalam menyelesaikan pembuatan skripsi dan berkat bimbingan beliau kendala ataupun hambatan dalam penyusunan skripsi ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini jauh dari sempurna. Untuk itu penulis mengharapkan kritik dan saran agar penulis dapat melakukan perbaikan terhadap skripsi yang disusun ini. Semoga skripsi ini dapat memberikan manfaat. Amiin.

Malang, Juli 2013

Penulis

ABSTRAK

Ervin Yohannes. 2013. : Pengenalan Sidik Jari (*Fingerprint recognition)* Dengan Metode *Hidden Markov Model* (HMM). Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang.
Dosen Pembimbing: Edy Santoso, S.Si., M.Kom. dan Ahmad Afif Supianto, S.Si., M.Kom.

Pengenalan *biometric* adalah proses mengenali seseorang berdasarkan karakteristik tingkah laku atau anatominya. Salah satu contoh pengenalan *biometric* adalah pengenalan sidik jari. Pengenalan ini banyak dianalisis dengan berbagai metode dan sudah dikembangkan oleh para peneliti. Pada penelitian ini dilakukan pengenalan sidik jari dengan metode *Hidden Markov Model* (HMM) yang menerapkan *computer vision* dan pengenalan pola. Tujuan dari penelitian ini untuk merancang dan mengimplementasikan perangkat lunak yang dapat mengidentifikasi sidik jari manusia. Alur prosesnya terdiri dari binerisasi, erosi, skeletonisasi, ekstraksi fitur dan parameter HMM. Data yang digunakan berjumlah 140 data yang terdiri dari 5 label, dimana 40 data sebagai data *testing* dan sisanya sebagai data *training*. Pengujian yang dilakukan adalah dengan uji coba terhadap jumlah data *training*. Terdapat 5 kali pengujian dilakukan yaitu pengujian terhadap 20 data *training*, 40 data *training*, 60 data *training*, 80 data *training* dan 100 data *training*. Hasil rata – rata akurasi yang didapatkan pada uji coba mencapai hingga 86% yang diperoleh dari 20, 40, 60, 80, dan 100 data *training*. Dari hasil pengujian didapatkan kesimpulan bahwa HMM dapat diimplementasikan kedalam sistem pengenalan sidik jari.

Kata kunci : *Biometrics*, pengenalan sidik jari, *Hidden Markov Model*

ABSTRACT

Ervin Yohannes. 2013. ***: Fingerprint recognition with Hidden Markov Model (HMM) Method. Undergraduate Thesis of Informatics Engineering Study Program, Program on Informatic Technology and Computer Science, University of Brawijaya,*** **Malang.**

Advisor : **Edy Santoso, S.Si., M.Kom.,** ***and*** **Ahmad Afif Supianto, S.Si., M.Kom.**

Biometric recognition is process recognizing person based characteristic behavior or anatomy. One example is fingerprint recognition. This recognition many analyzed with various method and development by researcher. In this research proposed fingerprint recognition with Hidden Markov Model (HMM) that applying computer vision and pattern recognition. The goal of this research for planned and implementation software that can identify human fingerprint. The process is binerization, erotion, skeletonization, fitur extraction, and development parameter HMM. Data used amount 140 data consists of 5 label, in which 40 used to testing and rest used to training. The testing is test to amount training data. There is 5 times testing doing for 20, 40, 60, 80 and 100 training data. Result accuracy is founded achieve 86% is obtained training data amount 20, 40, 60, 80, and 100. The conclusion is HMM can implementation into fingerprint recognition system.

Keywords : Biometrics, fingerprint recognition, Hidden Markov Model

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

DAFTAR *SOURCECODE*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengenalan *biometrics* adalah proses mengenali seseorang berdasarkan karakteristik tingkah laku atau anatominya. *Biometrics* yang bagus harus dapat diukur, unik (berbeda untuk setiap orang) dan tetap atau stabil dari waktu ke waktu [PRA-11]. Pengenalan *biometrics* pada manusia banyak diterapkan ke berbagai jenis pekerjaan seperti pemeliharaan kerja, urusan hukum, autentikasi pegawai, identifikasi pegawai, dan sebagainya. Keuntungan dari pengenalan *biometrics* adalah 1) pengguna tidak butuh mengingat password apapun, 2) pengguna tidak butuh membawa kartu identifikasi apapun, 3) pengguna tidak dapat menyangkal identifikasi *biometric*nya sendiri, dan 4) mengurangi sejumlah besar biaya pada pembuatan kartu ID pribadi atau dokumen yang terhubung [POR-10].

Salah satu contoh pengenalan *biometrics* adalah pengenalan sidik jari. Pengenalan sidik jari (*fingerprint recognition*) adalah masalah yang banyak dianalisis dengan berbagai metode. Metode yang digunakan untuk pengenalan sidik jari sudah banyak dikembangkan oleh para peneliti. Hiew, dkk (2008) menerapkan *Gabor filter* dan *Principle Component Analysis* (PCA) untuk mengenali pengenalan *touch less* sidik jari yang mana gambar diambil menggunakan kamera digital. Jing Luo, dkk (2008) menerapkan *Radial Basis Function Neural Network* untuk mengenali sidik jari. Dale, dkk (2008) melakukan *Discrete Cosinus Transform* (DCT) untuk pengenalan sidik jari dan masih banyak lagi peneliti yang menggunakan metode lain untuk pengenalan sidik jari. Pengenalan sidik jari mempunyai *equal error rate* (EER) yang kurang dari 5% pada dataset *Fingerprint Verification Competition* (FVC) 2002, 2004, dan 2006 [KRI-11]. Sidik jari mempunyai sebuah karakteristik pengenalan yang bernilai dan dapat diteliti dengan mudah. Pengenalan sidik jari harus mempertimbangkan performa dan keamanan. Ekstraksi fitur dalam sidik jari selalu menjadi yang menarik dalam bidang aplikasi sidik jari. Beberapa fitur *fingerprint* biasanya adalah *minutiae*, *moment*, dan *topology* [ZHA-12].

Hidden Markov Model (HMM) adalah teknik pembelajaran statistik yang banyak diterapkan pada model berurutan. HMM telah sukses dalam berbagai jenis bidang seperti *natural language processing* (termasuk pengenalan suara, translasi suara atau parsing), pengenalan pola dan *computer vision* (termasuk pengenalan bentuk, wajah dan pengenalan gerakan, analisis gambar) dan juga perhitungan biologi dan genetik [SUN-10]. HMM adalah mesin kondisi batas *stochastic* yang dihasilkan oleh deretan observasi, simbol atau vektor, dengan memproses sejumlah (*hidden*) kondisi, yang diperoleh dari probabilitas transisi [SUN-10]. HMM memiliki banyak keunggulan dalam bidang pengenalan dan klasifikasi diantaranya adalah beberapa jurnal menemukan bahwa pada pengenalan retina dengan metode HMM memiliki tingkat akurasi hingga 100% pada uji coba dengan jumlah *training* sebanyak 4 model gambar dan ukuran *codebook* 256, dan dengan jumlah *training* sebanyak 8 model gambar dan ukuran *codebook* 64 [YUL-08]. Dalam pengenalan wajah yang mengkombinasikan metode HMM dengan metode 2D-DCT dan metode fraktal masing-masing di peroleh akurasi rata-rata sebesar 72,83 dan 86,15 [SUN-10]. Pada identifikasi keadaan sebuah mata di peroleh akurasi sebesar 91,556% termasuk *eye opening* dan *eye closure* [HUA-12].

Berdasarkan latar belakang tersebut bahwa HMM adalah sebuah metode yang dapat digunakan untuk beberapa pengenalan *biometric* (pengenalan wajah dan retina) dan memperoleh rata-rata akurasi yang tinggi maka penulis mengusulkan judul "Pengenalan Sidik Jari (*Fingerprint Recognition*) Dengan Metode *Hidden Markov Model* (HMM)". Sistem ini menerapkan prinsip *computer vision* dan pengenalan pola.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang maka permasalahan yang akan dibahas antara lain :

1. Bagaimana *Hidden Markov Model* (HMM) mampu digunakan untuk pengenalan sidik jari.
2. Bagaimana peperancangan pengenalan sidik jari dengan metode *Hidden Markov Model* (HMM).

3. Bagaimana implementasi pengenalan sidik jari dengan metode *Hidden Markov Model* (HMM).
4. Bagaimana analisis hasil akurasi dari pengenalan sidik jari dengan metode *Hidden Markov Model* (HMM).

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah :

1. Mampu menggunakan *Hidden Markov Model* (HMM) dalam pengenalan sidik jari.
2. Merancang pengenalan sidik jari dengan metode *Hidden Markov Model* (HMM).
3. Mengimplementasikan pengenalan sidik jari dengan metode *Hidden Markov Model* (HMM).
4. Menganalisis hasil akurasi yang diperoleh dari pengenalan sidik jari dengan metode *Hidden Markov Model* (HMM).

1.4 Batasan Masalah

Pembatasan masalah pada penulisan ini adalah sebagai berikut :

1. Sidik jari yang digunakan meliputi ibu jari, jari kelingking, jari telunjuk, jari tengah dan jari manis.
2. Bentuk image yang akan digunakan untuk *training* dan *testing* adalah .tiff
3. Citra sidik jari yang digunakan didapatkan dari halaman website http://neurotechnology.com/download.
4. Citra sidik jari yang diperoleh memiliki filename xxx_y_z dimana x adalah *person ID*, y adalah *finger ID*, dan z adalah *number of scan*.
5. Ekstraksi fitur yang digunakan pada pengenalan sidik jari adalah bifurcation dengan ukuran matriks 3 x 3.
6. Hasil pengenalan sidik jari yang diambil hanya pada *person ID*.

1.5 Manfaat

Adapun manfaat yang dapat diperoleh dari penelitian ini baik untuk penulis maupun untuk pihak ketiga sebagai berikut :

1. Menerapkan ilmu yang diperoleh selama menempuh studi di Teknik Informatika Universitas Brawijaya.
2. Mengembangkan penelitian interdisipliner.
3. Membantu dalam mendeteksi sidik jari yang dimiliki seseorang.

1.6 Sistematika Penulisan

Sistematika penulisan penelitian untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, antara lain :

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat tujuan, dan sistematika penulisan.

BAB II Kajian Pustaka dan Dasar Teori

Membahas teori dasar dan teori penunjang yang berhubungan dengan citra digital, pengenalan sidik jari (*fingerpint recognition*), *thresholding*, skeletonisasi, morfologi, ekstraksi fitur dan *Hidden Markov Model* (HMM).

BAB III Metode Penelitian dan Peperancangan

Menguraikan tentang metode dan langkah kerja yang dilakukan dalam penulisan tugas akhir yang terdiri dari studi literatur, metode pengambilan data, analisis kebutuhan, peperancangan sistem, implementasi, pengujian dan analisis, pengambilan kesimpulan dan membahas analisis kebutuhan dan peperancangan yang sesuai dengan teori yang ada.

BAB IV Implementasi

Membahas tentang implementasi dari sistem.

BAB V Pengujian dan Analisis

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

BAB VI Penutup

Menguraikan kesimpulan yang diperoleh dari pembuatan dan pengujian aplikasi sistem, serta saran–saran untuk pengembangan lebih lanjut.

1.7 Jadwal Penelitian

Penulisan skripsi ini direncanakan dengan rincian sebagai berikut :

Tabel 1.1 Tabel Jadwal penelitian

NO	PROSES PENULISAN:	BULAN DAN MINGGU KE:																			
		Bulan I				Bulan II				Bulan III				Bulan IV				Bulan V			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi literatur																				
2	Pengambilan data																				
3	Pemodelan sistem																				
4	Peperancangan dan pembangunan sistem																				
5	Pengujian hasil																				
6	Penulisan penelitian																				

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini berisi pembahasan tentang teori dasar yang berhubungan dengan pengembangan sistem pengenalan sidik jari yang dilakukan. Teori dasar yang akan dibahas pada bab ini yaitu penjelasan tentang citra digital, konsep dasar pengenalan sidik jari, konsep dasar *thresholding*, konsep dasar morfologi dan konsep dasar *Hidden Markov Model* (HMM).

2.1 Kajian Pustaka

Kajian pustaka pada penelitian " adalah membahas penelitian sebelumnya yang berjudul "*Fingerprint Recognition By Euclidian Distance*". Penelitian ini membahas tentang pengenalan sidik jari manusia. Metode yang digunakan pada penelitian ini adalah *euclidian distance*. *Euclidian distance* digunakan untuk menghitung jarak antara *core point* dan masing-masing *bifurcation* dalam masing-masing sektor dan sebuah lintasan [POR-10].

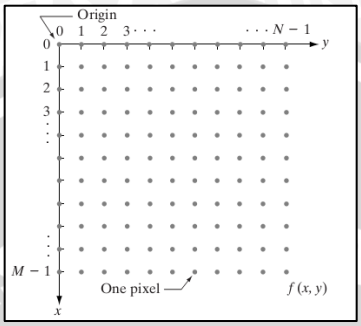
Perbedaan yang dibuat penulis pada penelitian ini adalah pada penggunaan metode yang digunakan yaitu dengan menggunakan metode HMM. Selain itu, itu juga terdapat pada proses ekstraksi fitur yang digunakan yaitu dengan menggunakan *bifurcation* saja tanpa menggunakan *core point*. Dengan adanya perbedaan ini penulis berharap dapat membandingkan dua metode dan membandingkan tingkat akurasi yang diperoleh selama pengujian.

2.2 Citra Digital

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar 2 dimensi yang menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah larik (*array*) yang berisi nilai-nilai *real* maupun komplek yang direpresentasikan dengan deretan bit tertentu.

Suatu citra dapat didefinisikan sebagai fungsi $f(x,y)$ berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik

koordinat *(x,y)* dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai *x,y* dan nilai amplitudo *f* secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital. Gambar 2.1 menunjukkan posisi koordinat citra digital.



Gambar 2.1 Koordinat citra digital

Sumber : [GON-02]

Notasi M x N dapat ditulis kedalam bentuk matrik berikut ini.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) \dots \dots \dots f(0,N-1) \\ f(1,0) & f(1,1) \dots \dots \dots f(1,N-1) \\ \cdot & \cdot \quad\quad\quad \cdot \\ \cdot & \cdot \quad\quad\quad \cdot \\ \cdot & \cdot \quad\quad\quad \cdot \\ f(M-1,0) & f(M-1,1) \dots \dots \dots f(M-1,N-1) \end{bmatrix} \quad (2.1)$$

Nilai pada suatu irisan antara baris dan kolom (pada posisi *x,y*) disebut dengan *picture elements, image elements, pels,* atau *pixels.* Istilah terakhir (*pixel*) paling sering digunakan pada citra digital.

2.3 Pengenalan Sidik Jari (*Fingerprint Recognition*)

Sidik jari bersifat unik untuk setiap individu dan tidak akan berubah seumur hidup kecuali disebabkan oleh kecelakaan seperti luka parah pada jari [MAL-03]. Pola sidik jari dapat dibagi menjadi dua tipe garis, yaitu *ridge* dan *valley*. *Ridge* merupakan garis yang berwarna gelap, sedangkan *valley* adalah daerah antara *ridge* yang berwarna terang. Untuk lebih jelasnya dapat dilihat pada Gambar 2.2 yang diambil dari verifinger_sample_DB.

Gambar 2.2 Contoh sidik jari. Garis berwarna hitam adalah *ridge* sedangkan daerah putih diantaranya adalah *valley*
Sumber : [NEU-12]

Representasi sidik jari yang baik harus memiliki dua properti berikut :

1. *Saliency* : mengandung informasi yang khusus mengenai sidik jari tersebut.
2. *Suitability* : mudah diekstrak, disimpan dalam media yang padat, dan berguna untuk proses verifikasi.

Pola sidik jari saat dianalisis dalam beberapa skala yang berbeda menghasilkan beberapa fitur yang berbeda, yaitu :

a. Pada level global, garis-garis terbentuk gelombang pada sidik jari menggambarkan titik-titik yang disebut Titik Singular. Titik Singular, yang juga disebut *loop* atau delta, adalah titik dimana alur-alur garis yang terdapat pada sidik jari memutar balik. Titik Singular dan tingkat kekasaran alur sidik jari sangat berpengaruh pada pengindeksian dan klasifikasi sidik jari [MAL-03].
b. Pada *level* lokal, teridentifikasi lebih dari 100 karakteristik garis *ridge* yang disebut *minutiae*. Terdapat dua tipe *minutiae* [MAL-03], yaitu :
    1. *Ridge ending* : titik dimana garis ridge memiliki ujung yang kasar.
    2. *Ridge bifurcation* : titik dimana garis ridge mengalami percabangan
c. Analisis pada level yang lebih *detail* berhasil mengidentifikasikan *sweat pores* (pori-pori tempat keluarnya keringat) pada jari. Proses klasifikasi sidik jari menggunakan titik ini hanya dapat dilakukan pada gambar sidik jari yang beresolusi tinggi, misalnya 100 dpi.

Ada sejumlah teknik atau metode yang digunakan untuk pengenalan sidik jari dan beberapa teknik ini didiskusikan sebagai berikut.

a. *Gabor Based Technique*

Aguilar, G., dkk (2008) mengombinasikan *Fast Fourier Transform* (FFT) dengan *gabor filter* untuk pengenalan sidik jari dengan presisi 94.1 persen.

b. Metode *Neural Network*

Astrov, I., dkk (2008) menggunakan *Three-Rate Hybrid Kohonen Neural Network* (TRHKNN) untuk mengenali sidik jari. Ju Cheng Yang, Adrian Lim Hooi Jin, dkk (2007) mengusulkan *back-propagation neural network* untuk mengidentifikasi sidik jari. K. Umamaheswari, dkk (2007) mengusulkan klasifikasi dan pengenalan sidik jari dengan menggunakan *neural network* dan data mining dengan presisi 97.4 persen.

c. *Wavelet-Based Feature*

Ada banyak peneliti yang mengaplikasikan *Fast Fourier Transform* (FFT) untuk mengenali sidik jari dengan presisi sekitar 90 persen [LEO-09].

d. Teknik *Euclidian Distance*

Pradeep M.P. dan Jin Q. Z., dkk (2005) mengusulkan sebuah sistem pengenalan sidik jari dengan menggunakan teknik *Euclidian distance* dengan rata-rata presisi tinggi.

2.4 *Thresholding*

Citra pada umumnya terdiri dari *pixel-pixel* dengan variasi nilai RGB (komponen warna merah/*red*, hijau/*green*, dan biru/*blue*). Citra jenis ini disebut sebagai citra berwarna. Citra abu-abu (*grayscale*) adalah citra dengan satu macam warna yakni abu-abu, dengan variasi derajat keabuan. Proses mengubah citra warna menjadi citra *grayscale* digunakan dalam *image processing* untuk menyederhanakan elemen matriks dari suatu citra. Citra berwarna terdiri dari 3 *layer* matriks yaitu R-*layer*, G-*layer*, B-*layer*. Untuk mengubah citra berwarna menjadi abu-abu maka 3 *layer* tersebut diubah menjadi 1 *layer* matrik *grayscale*. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan.

Thresholding merupakan proses pemisahan piksel-piksel berdasarkan derajat keabuan yang dimilikinya. Piksel yang memiliki derajat keabuan lebih

kecil dari nilai batas yang ditentukan akan diberikan nilai 0, sementara piksel yang memiliki derajat keabuan yang lebih besar dari batas akan diubah menjadi bernilai 1. Metode ini digunakan untuk mengubah data *image* menjadi data biner dengan tujuan agar proses selanjutnya menjadi lebih mudah [YUL-08]. Dalam penelitian ini digunakan metode Otsu untuk binerisasi.

Metode Otsu menghitung nilai ambang (*T*) secara otomatis berdasarkan gambar masukan. Pendekatan yang digunakan oleh metode Otsu adalah dengan melakukan analisis diskriminan yaitu menentukan suatu variabel yang dapat membedakan antara dua atau lebih kelompok yang muncul secara alami. Analisis diskriminan akan memaksimumkan *variable* tersebut agar dapat memisahkan objek dengan latar belakang. Misalkan nilai ambang yang akan dicari dinyatakan dengan *k.* Nilai *k* berkisar antara 1 sampai dengan *L*, dengan *L* = 255, probabilitas untuk piksel *i* dinyatakan dengan :

$$P_i = \frac{n_i}{N} \quad (2.2)$$

Keterangan :

P_i = probabilitas untuk piksel i

n_i = jumlah piksel dengan tingkat keabuan i

N = banyaknya piksel pada gambar

Nilai momen kumulatif ke-nol, momen kumulatif ke-satu, dan nilai rata-rata berturut-turut dapat dinyatakan sebagai berikut :

$$\omega(k) = \sum_{i=1}^{k} P_i \quad (2.3)$$

Keterangan :

$\omega(k)$ = momen kumulatif ke 0

k = nilai ambang berkisar antara 1 sampai dengan 255

P_i = probabilitas untuk piksel i

$$\mu(k) = \sum_{i=1}^{k} i.P_i \quad (2.4)$$

Keterangan :

$\mu(k)$ = momen kumulatif ke 1

k = nilai ambang berkisar antara 1 sampai dengan 255

i = nilai ambang antara 1 sampai dengan k

P_i = probabilitas untuk piksel i

$\mu_T = \sum_{i=1}^{L} i.P_i$ (2.5)

Keterangan :

μ_T = nilai rata-rata

L = 255

i = nilai ambang berkisar antara 1 sampai dengan 255

P_i = probabilitas untuk piksel i

Nilai ambang k dapat ditentukan dengan memaksimumkan persamaan :

$\sigma_B^2(k^*) = max_{1 \le k \le L} \sigma_B^2(k)$ (2.6)

Keterangan :

σ_B = variansi

k = nilai ambang antara 1 sampai dengan 255

$\sigma_B^2 = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]}$ (2.7)

Keterangan :

σ_B = variansi

$\omega(k)$ = momen kumulatif ke 0

$\mu(k)$ = momen kumulatif ke 1

μ_T = nilai rata-rata

Untuk menghitung nilai ambang (*T*), pertama adalah dengan membuat histogram dari gambar yang dimaksud yaitu dengan menambah nilai *k* jika terdapat piksel yang memiliki nilai sama dengan *level* (*L*), dalam hal ini *level* (*L*) adalah *level* untuk gambar dengan format *grayscale* mulai dari 0 (nol sampai dengan 255. Selanjutnya adalah menghitung nilai probabilitas untuk setiap *level* (*L*) dengan membandingkan antara jumlah piksel pada *level* dengan total piksel pada gambar atau dapat dituliskan seperti pada persamaan (2.2). Kemudian untuk setiap *level* dihitung jumlah nilai momen kumulatif ke 0 dengan persamaan (2.3), momen kumulatif ke -1 dengan persamaan (2.4) dan nilai rata-rata dengan persamaan (2.5) yang selanjutnya dihitung nilai varian dengan persamaan (2.7). Jika seluruh *level* telah dihitung, selanjutnya dicari nilai varian yang paling tinggi untuk digunakan sebagai nilai ambang (*T*) [ARD-11].

2.5 Skeletonisasi

Skeletonisasi merupakan salah satu cara yang dapat digunakan untuk mereduksi objek atau citra awal menjadi suatu representasi yang lebih sederhana dan lebih lengkap namun sifat-sifat objek yang umum seperti topologi maupun karakteristik ukuran dari citra asli masih dapat digambarkan. Kerangka (*skeleton*) mengekspresikan konektivitas struktural dari suatu komponen utama sebuah objek dengan hanya menampilkan citra objek tersebut selebar satu piksel.

Thinning merupakan metode dasar dalam skeletonisasi. Proses ini bekerja dengan teknik pengulangan, dimana teknik ini mengekstrak kerangka dari sebuah objek sebagai hasilnya. Pada setiap tahapnya, proses *thinning* akan mengikis piksel-piksel yang ada dimulai dari piksel yang terletak pada bagian terluar dari objek dan dilakukan berulang-ulang sampai proses *thinning* tersebut tidak mungkin dapat dilakukan lagi, sehingga yang tertinggal hanyalah kerangka (*skeleton*) dari objek tersebut.

Proses penghapusan ini dilakukan dengan hati-hati agar tidak mengubah topologi dari objek. Kerangka tersebut yang selanjutnya merepresentasikan bentuk dari sebuah objek dalam jumlah piksel kecil. Algoritma *thinning* hanya dapat diterapkan pada *binary image* saja yaitu citra hitam putih yang hanya terdiri dari piksel level 0 dan 1, atau 0 dan 255 [JAM-08]. Dalam proses skeletonisasi digunakan algoritma Zhang-Suen.

Algoritma *thinning* paralel cepat Zhang-suen diusulkan pada tahun 1984. Yang mana diketahui objek ditandai sebagai 1, dan *background point* sebagai 0. Titik batas adalah 1, 8-hubungan yang berdekatan setidaknya memiliki satu titik ditandai dengan 0. Menurut algoritma ini, titik batas dibutuhkan untuk di proses dalam prosedur berikut ini :

a. Dalam titik batas dipusatkan 8-hubungan yang berdekatan, titik pusat adalah *p1* dan titik ketetanggaannya searah jarum jam yang dinotasikan *p2,p3,...,p9*. Diantara titik-titik ini, *p2* ada diatas titik pusat *p1* (seperti ditunjukkan dalam Gambar 2.3). Pertama-tama, semua titik yang memenuhi persyaratan dibawah ini dipilih (ditandai/dihapus dijadikan nilai *background*) :

   1. $2 \leq B(p1) \leq 6;$

2. *A(p1)=1*
3. *p2*p4*p6=0;*
4. *p4*p6*p8=0;*

p9	p2	p3
p8	p1	p4
p7	p6	p5

Gambar 2.3 Diagram 8-ketetanggaan

B(p1) adalah jumlah titik terdekat yang bukan nol, dan *A(p1)* adalah jumlah perubahan dari 0 ke 1 dalam deret *p2,p3,....p9*. setelah semua titik di periksa maka semua titik yang ditandai dihilangkan/dijadikan warna *background*.

b. Sama seperti langkah (a), persyaratan (3) diubah yaitu *p2*p4*p8=0* dan persyaratan (4) adalah *p2*p6*p8=0*. Titik-titik yang ditandai dihapus setelah diperiksa.

Dua langkah diatas adalah sebuah iterasi. Sampai tidak ada titik-titik yang memenuhi persyaratan, sehingga titik-titik yang tersisa akan membentuk wilayah kerangka [WEI-12].

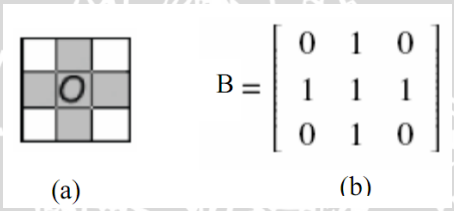
2.6 Morfologi

Matematika morfologi merepresentasikan citra objek dua dimensi sebagai suatu himpunan matematika dalam ruang *Euclidian* E^2, dimana dapat berupa ruang kontinu R^2 atau ruang diskrit Z^2 [SRI-10]. Dulu sebuah citra dipandang sebagai suatu fungsi intensitas terhadap posisi *(x,y),* sedangkan dengan pendekatan morfologi, suatu citra dipandang sebagai himpunan. Sebuah objek citra A dapat direpresentasikan dalam bentuk himpunan dari posisi-posisi *(x,y)* yang bernilai 1 atau 0 dimana nilai-nilai tersebut menunjukkan tingkat *grayscale* setiap posisi. Nilai 1 untuk gray level warna putih dan nilai 0 untuk gray level warna hitam.

Prinsip dasar dari matematika morfologi adalah penggunaan *structuring element* yaitu bentuk dasar dari suatu objek yang digunakan untuk menganalisis struktur geometri dari objek lain yang lebih besar dan kompleks [SER-82]. Tujuannya adalah untuk memperoleh informasi mengenai bentuk dari suatu citra dengan mengatur bentuk dari ukuran suatu *structuring element.*

2.7 *Structuring Element*

Structuring element dapat diibaratkan dengan *mask* pada pemrosesan citra biasa (bukan secara morfologi). *Structuring element* juga memiliki titik poros (disebut juga titik origin atau titik asal atau titik acuan). Di bawah ini adalah contoh *structuring element* dengan titik poros di (0,0) ditunjukkan dengan huruf "O" (Gambar 3).



Gambar 2.4 Contoh *Structuring Element* (a) titik "O" adalah titik poros, (b) representasi biner strel

Sumber : [GON-02]

Bentuk *structuring element* pada Gambar 3(a) dapat direpresentasikan dalam bentuk matriks biner seperti pada Gambar 3(b) di mana angka "1" dan angka "0" menunjukkan nilai gray *level*. Dalam morfologi, yang menjadi kunci penting adalah pemilihan *structuring element*. *Structuring element* memiliki dua komponen yang penting yaitu bentuk dan ukuran dimana keduanya mempengaruhi hasil pengujian. Pemilihan bentuk *structuring element* juga mempengaruhi citra hasil operasi morfologi.

Morfologi mempunyai dua operator dasar, yaitu Dilasi (*dilation*) dan Erosi (*erosion*) yang biasa digunakan untuk mengekstrak komponen yang diinginkan dalam sebuah citra. Berdasarkan dua operator tersebut, dapat diturunkan dua operator lainnya yang berguna untuk menghaluskan batas subinterval komponen

yang telah diekstrak, yaitu Pembukaan (*opening*) dan Penutupan (*closing*) [GON-02].

Dalam penggunaannya, Morfologi selalu melibatkan sebuah citra dengan komponen *I* (citra) dan elemen penyusun *E* (*structuring element*). Operator-operator Morfologi tersebut adalah sebagai berikut [GON-02]:

Dilasi : $I \oplus E = \{z \mid (\hat{E}_z) \cap I \neq \emptyset\}$

Erosi : $I \ominus E = \{z \mid (E)_z \subseteq I\}$

Opening : $I \circ E = (I \ominus E) \oplus E$

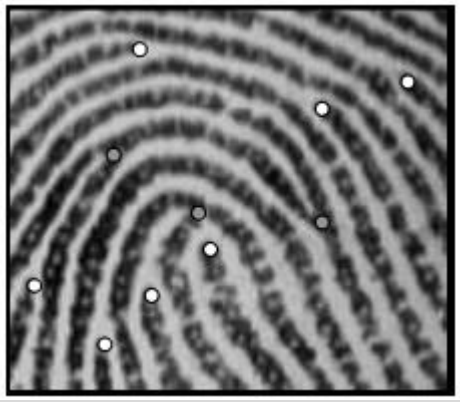
Closing : $I \bullet E = (I \oplus E) \ominus E$

$(E)_z$ merupakan translasi dari komponen *I* terhadap titik *z*, sedangkan $(\hat{E}_z)$ adalah refleksinya. Dilasi digunakan untuk memperbesar komponen yang diinginkan dengan cara menambahkan seluruh tepinya dengan elemen penyusun *E*. Erosi digunakan untuk mengikis komponen yang diinginkan dengan cara mengurangi seluruh tepinya dengan elemen penyusun *E*.

2.8 Ekstraksi Fitur

Didalam sebuah gambar sidik jari, *ridge* (juga dinamakan daerah garis) adalah area gelap sedangkan lembah (*valley*) adalah area terang (seperti pada Gambar 2.5) *ridges* dan lembah sering berjalan paralel; kadang-kadang mereka bercabang dan kadang – kadang mereka putus. Pada *level* lokal, fitur penting lainnya, dinamakan *minutiae* dapat ditemukan dalam pola sidik jari. *Minutiae* menunjuk pada cara yang beragam jenis dimana *ridges* dapat terputus-putus. Untuk contoh, sebuah *ridge* tiba-tiba datang ke sebuah *end* (putus), atau dapat dibagi kedalam dua *ridges* (*bifurcation*) (Gambar 2.5) [JAI-07].

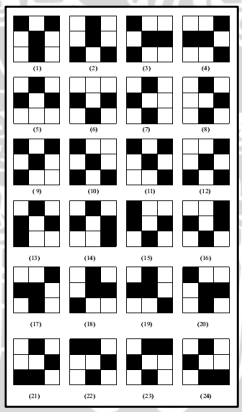
Sebuah *ridge bifurcation* adalah titik dalam gambar dimana *ridge* bercabang dua. Fitur ini unik untuk setiap sidik jari dan digunakan untuk pengenalan sidik jari. Syarat untuk memperoleh akurasi titik *bifurcation* adalah memperoleh gambar biner dimana semua *ridges* dari sidik jari telah benar-benar menipis ke garis pusatnya. Gambar tipis seharusnya dijadikan selebar satu piksel tanpa ketidaksinambungan karena penipisan [PAT-05].



Gambar 2.5 *Termination* (putih) dan *bifurcation* (*gray*) *minutiae* dalam contoh sidik jari.

Sumber : [JAI-07]

Bifurcation dibentuk menjadi matriks berukuran 3 x 3. Jumlah dari *bifurcation* sebanyak 24 buah seperti yang ada pada Gambar 2.6. Tujuan dari ekstraksi fitur adalah untuk mengambil fitur-fitur yang ada pada citra sidik jari yang berguna sebagai pembeda citra sidik jari satu dengan yang lainnya.



Gambar 2.6 3x3 *pixel mask* dan 24 kemungkinan pola *bifurcation*

Sumber : [KAU-08]

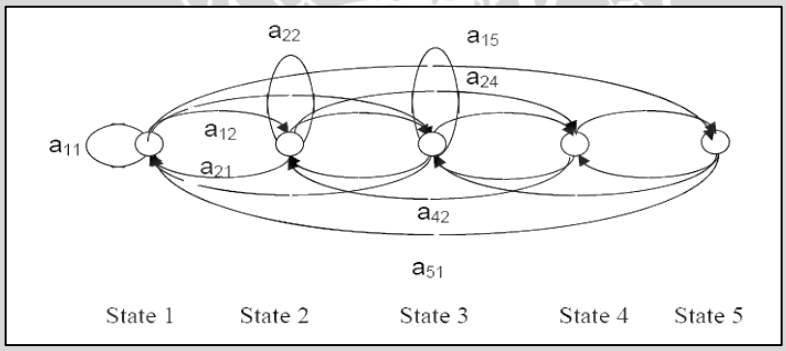
2.9 Metode *Template Matching*

Template matching adalah sebuah teknik dalam pengolahan citra digital untuk menemukan bagian-bagian kecil dari gambar yang cocok dengan *template* gambar. Metode *template matching* merupakan salah satu metode yang digunakan untuk menjelaskan bagaimana otak manusia mengenali kembali bentuk-bentuk

atau pola-pola [LEK-11]. Cara kerja *template matching* ini adalah melakukan *pattern recognition* pada karakter atau objek yang ingin dikenali dan membandingkan antara input *pattern* dengan *template* yang disimpan. *Template matching* yang disebut juga dengan *pattern matching* atau *matrix matching* merupakan suatu proses membandingkan suatu objek karakter yang biasanya disebut sebagai *glyph* dengan sejumlah *template*. *Glyph* yang telah diekstrak akan dicocokkan dengan *template matching* yang telah disimpan dalam database *template* [ELI-08].

2.10 *Hidden Markov Model* (HMM)

Hidden markov model merupakan pemodelan probabilitas suatu sistem dengan mencari parameter-parameter markov yang tidak diketahui untuk memperoleh analisis sistem tersebut. Metode *hidden markov model* (HMM) mampu menangani perubahan statistik dari gambar, dengan memodelkan elemen-elemen menggunakan probabilitas. Salah satu aplikasinya adalah pada *image processing*, HMM memiliki tiga parameter utama yang harus dicari nilainya terlebih dahulu, ketiga parameter tersebut sebagai berikut :



Gambar 2.7 Contoh matriks transisi

Sumber : [SEP-12]

Parameter A dalam HMM dinyatakan dalam sebuah matriks dengan ukuran *M x M* dengan *M* adalah jumlah state yang ada. Pada Gambar 2.7 ada 5 (lima *state* sehingga setiap *state* memiliki 5 hubungan transisi, maka parameter *A* dapat dituliskan dalam bentuk matriks seperti pada gambar berikut :

$$A = a_{ij} = \begin{bmatrix} a11 & a12 & a13 & a14 & a15 \\ a21 & a22 & a23 & a24 & a25 \\ a31 & a32 & a33 & a34 & a35 \\ a41 & a42 & a43 & a44 & a45 \\ a51 & a52 & a53 & a54 & a55 \end{bmatrix} \quad (2.8)$$

$$a_{ij} = P\left[q_{t+1} = S_j \middle| q_1 = S_i\right] \qquad 1 \leq i, j \leq N \quad (2.9)$$

Keterangan :

a_{ij} = probabilitas transisi dari state i ke state j

P = peluang/probabilitas

q_{t+1} = kondisi sesudah q_t

q_1 = kondisi saat ini

Sj = state ke j

Si = state ke i

Parameter B disebut sebagai probabilitas *state*, merupakan proses kemunculan suatu *state* dalam deretan seluruh *state* yang ada. Parameter B dalam HMM dituliskan dalam bentuk matriks kolom dengan ukuran $M \ x \ 1$, dimana M merupakan jumlah seluruh *state* yang ada. Misalnya terdapat 5 buah *state* dalam suatu kondisi, maka matriks B yang terbentuk ditunjukkan oleh persamaan berikut :

$$B = \begin{bmatrix} b1 \\ b2 \\ b3 \\ b4 \\ b5 \end{bmatrix} \quad (2.10)$$

$$b_j(k) = P\left[v_k at\ t \middle| q_t = S_j\right] \quad 1 \leq j \leq N, \quad 1 \leq k \leq M \quad (2.11)$$

Keterangan :

$b_j(k)$ = observasi simbol probabilitas distribusi matriks

P = peluang/probabilitas

v_k = probabilitas simbol dengan kluster index k dalam *state* j

q_t = kondisi pada waktu tertentu yang berhubungan dengan q_1

Sj = *state* ke j

Parameter π, disebut sebagai parameter awal, merupakan probabilitas kemunculan suatu *state* di awal. Sama halnya dengan parameter B, parameter π juga dituliskan dalam bentuk matriks kolom dengan ukuran $M \ x \ 1$, dimana M

adalah jumlah *state* nya, jadi jika terdapat 5 (lima) bauah *state* maka parameter π yang dihasilkan akan ditunjukkan seperti pada gambar berikut.

$$\pi = \begin{bmatrix} c1 \\ c2 \\ c3 \\ c4 \\ c5 \end{bmatrix} \quad (2.12)$$

keterangan :

π = parameter awal / inisial awal

c1,c2,...,c5 = nilai inisial dari state ke-1 sampai ke-5

Hidden markov model dapat dituliskan sebagai $\lambda = (A,B,\pi)$. Dengan diketahuinya *N,M,A,B*, dan π, *hidden markov model* dapat menghasilkan urutan observasi $O = O_1O_2...O_T$ dimana masing-masing observasi O_t adalah simbol *V*, dan *T* adalah jumlah urutan observasi. *Hidden markov model* dapat dituliskan sebagai $\lambda = (A,B,\pi)$. Perhitungan yang efisien dari $P(O|\lambda)$, yaitu probabilitas urutan observasi apabila diberikan urutan observasi $O = O_1O_2...O_T$ dan sebuah model $\lambda = (A,B,\pi)$ [SEP-12].

Sebuah *hidden markov model* dikarakteristikkan dengan parameter berikut :

N, jumlah *state* dalam model

M, jumlah simbol pengamatan yang dimiliki setiap *state*

A = matriks transisi, himpunan distribusi kemungkinan perpindahan *state*

B = matriks emisi, himpunan distribusi kemungkinan symbol pengamatan pada *state* j

Π = inisial awal, himpunan distribusi kemungkinan *state* awal

Pelatihan model, diberikan sekumpulan sequence {*Oi*}, biasanya dihitung dengan menggunakan Baum-Welch re-estimation, dapat untuk menentukan parameter - parameter (A,B,Π) yang memaksimalkan probabilitas $P(\{Oi|\lambda\})$. Prosedur pelatihan dihentikan setelah konvergen dari *likelihood*. Langkah evaluasi yaitu menghitung probabilitas $P(O|\lambda)$, diberikan sebuah model λ dan sebuah deretan *O* untuk dievaluasi, menggunakan algoritma *forward* [BIC-03]. Algoritma forward dan backward digunakan untuk menghitung probabilitas dari urutan nilai observasi yang diberikan oleh HMM dan untuk evaluasi parameter HMM. Algoritma Baum-

Welch digunakan untuk melatih parameter HMM jika diberikan dataset barisan-barisan tertentu agar dapat menemukan himpunan transisi *state* yang paling mungkin beserta probabilitas *output*-nya. Berikut ini penjelasan mengenai algoritma yang digunakan dalam HMM [MUL-08]:

a. Persoalan dalam evaluasi dapat diselesaikan dengan menggunakan algoritma yang dinamakan prosedur maju-mundur (forward-backward prosedur). Pertama, dijelaskan prosedur forward, diasumsikan variable maju $\alpha t(i)$ didefinisikan sebagai :

$$\propto t(i) = P(O1, O2, \dots, Ot, qt = Si \mid \lambda \quad (2.13)$$

Yaitu kemungkinan rangkaian pengamatan parsial hingga waktu t dan berada pada state Si pada waktu t, jika diberikan model λ, maka $\alpha t(i)$ dapat dihitung dengan induksi sebagai berikut :

1. Inisialisasi

$$\propto t(i) = \pi i bi(O1), 1 \leq i \leq N \quad (2.14)$$

2. Induksi

$$\propto_{t+1} (j) = [\sum_{i=1}^{N} \propto_t (i) a_{ij}] b_j(O_{t+1}) \;\; 1 \leq t \leq T-1, \;\; 1 \leq j \leq N \quad (2.15)$$

3. Terminasi

$$P(O|\lambda) = \sum_{i=1}^{N} \propto_T (i) \quad (2.16)$$

Langkah pertama merupakan inisialisasi, langkah induksi merupakan langkah yang paling utama pada prosedur forward. $P(O|\lambda)$ dapat dicari dengan menjumlahkan variabel maju dengan $t=T$ dari semua state. $P(O|\lambda)$ merupakan probabilitas model menghasilkan rangkaian pengamatan $O = O1, O2, \dots, OT$.

b. Dengan cara yang sejenis dapat didefinisikan variabel mundur $\beta t(i)$ sebagai berikut :

$$\beta t(i) = P(Ot + 1, Ot + 2, \dots, OT | qt = Si, \lambda) \quad (2.17)$$

Yaitu kemungkinan rangkaian pengamatan dari $t+1$ hingga T jika diberikan state Si pada waktu t dan model λ. $\beta t(i)$ dapat diselesaikan sebagaimana $\alpha t(i)$.

1. Inisialisasi

$$\beta T(i) = 1, 1 \leq i \leq N \quad (2.18)$$

2. Induksi

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)\, t = T-1, T-2, \ldots, 1$$

$$1 \leq i \leq N \quad (2.19)$$

Variabel mundur akan digunakan pada proses estimasi nilai parameter-parameter HMM.

c. Persoalan evaluasi merupakan persoalan yang paling sulit , bisa dikatakan sebagai persoalan pelatihan yang digunakan untuk menghasilkan parameter model *A, B,* dan π optimal sehingga dapat dengan baik merepresentasikan rangkaian observasi yang terjadi. Kriteria optimal adalah memaksimalkan probabilitas rangkaian pengamatan, *P(O/λ)*, dengan diberikan model, *λ(A,B,π)*. Tidak ada pendekatan analitik untuk permasalah ini, namun terdapat prosedur iteratif seperti metode Baum-Welch yang dapat digunakan. Untuk mendeskripsikan formula pelatihan secara matematis, diasumsikan *ξt(i,j)* sebagai probabilitas berada pada *state i* pada waktu *t*, dan *state j* pada waktu *t+1*, diberikan model rangkaian pengamatan :

$$\xi t(i,j) = P(qt = Si, qt+1 = Sj | O, \lambda) \quad (2.20)$$

Dengan menggunakan definisi variabel maju dan mundur, persamaan diatas dapat ditulis dalam bentuk :

$$\xi_t(i,j) = \frac{(\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j))}{(P(O|\lambda))} \quad (2.21)$$

$$\xi_t(i,j) = \frac{(\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j))}{(\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j))} \quad (2.22)$$

P(O/λ) merupakan nilai probabilitas model λ memberikan sequence *O*. Biasanya *P(O/λ)* sering diberi nilai 1, nilai harapan model λ memberikan sequence *O*. Menjumlahkan *ξt(i,j)* pada $1 \leq t \leq T-1$ menghasilkan jumlah perpindahan dari *i* ke *j* yang diharapkan. Untuk kebutuhan pelatihan,

didefinisikan probabilitas berada di *state i* pada waktu *t*, diberikan rangkaian pengamatan dan model sebagai *γt(i)*.

$$\gamma_t(i) = \frac{(\propto_t(i)\beta_t(i))}{(P(O|\lambda))} = \frac{(\propto_t(i)\beta_t(i))}{(\sum_{i=1}^{N} \propto_t(i)\beta_t(i))} \quad (2.23)$$

Selanjutnya, *γt(i)* dan *ξt(i,j)* dapat dihubungkan dengan menjumlahkan *ξt(i,j)* untuk semua j :

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \quad (2.24)$$

Menjumlahkan, *γt(i)* sepanjang waktu memberikan jumlah state i dikunjungi. Jika waktu *T* tidak dimasukkan, dengan kata lain menjumlahkan sepanjang *1 ≤ t ≤ T-1*, ini memberikan jumlah perpindahan dari state i.

$$\sum_{t=1}^{T-1} \gamma_t(i) = jumlah\ perpindahan\ yang\ diharapkan\ Si \quad (2.25)$$

$$\sum_{j=1}^{N} \xi_t(i,j) = jumlah\ perpindahan\ yang\ diharapkan\ dari\ Si\ ke\ Sj \quad (2.26)$$

Menggunakan formula di atas, dapat didefinisikan formula untuk melakukan estimasi terhadap nilai-nilai parameter HMM

$$\overline{a_{ij}} = \frac{(jumlah\ perpindahan\ yang\ diharapkan\ dari\ Si\ ke\ Sj)}{(jumlah\ perpindahan\ yang\ diharapkan\ dari\ Si)}$$

$$\overline{a_{ij}} = \frac{\left(\sum_{j=1}^{N} \xi_t(i,j)\right)}{\left(\sum_{t=1}^{T-1} \gamma_t(i)\right)} \quad (2.27)$$

$$\overline{b_j}(k) = \frac{(jumlah\ frekuensi\ yang\ diharapkan\ pada\ state\ j\ dan\ menghasilkan\ simbol\ Vk)}{(jumlah\ frekuensi\ pada\ state\ j)}$$

$$\overline{b_j}(k) = \frac{\left(\sum_{t=1}^{T-1} \gamma_t(i)\right)}{\sum_{s.t.Ot=vk} \left(\sum_{t=1}^{T-1} \gamma_t(i)\right)} \quad (2.28)$$

2.11 *Manhattan Distance*

Nama Manhattan sendiri diambil dari daerah Manhattan suatu daerah kecil di Kota New York, yang memiliki jalan yang berbentuk kisi-kisi segiempat. Jarak antara dua lokasi yang berada di setiap kisi-kisi daerah Manhattan dapat diukur berdasarkan jalur horizontal dan vertikal yang terbentuk diantara kisi-kisi jalan

tersebut. Perhitungan jarak antara dua lokasi dapat dihitung dengan menggunakan perhitungan Phytagoras terhadap total jalur horizontal dan jalur vertikal yang terbentuk. Manhattan distance adalah formula untuk menghitung jarak antara dua titik. Perhitungan Manhattan distance untuk mencari jarak minimal dari dua buah titik *(x1 , y1)* dan *(x2 , y2)* dapat dilakukan dengan menghitung nilai */x2 - x1/ + /y2 - y1*| [WIB-12].

BAB III
METODE PENELITIAN DAN PEPERANCANGAN

Didalam bab metode penelitian dan peperancangan akan dibahas mengenai studi literatur yang digunakan, analisis kebutuhan, peperancangan 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

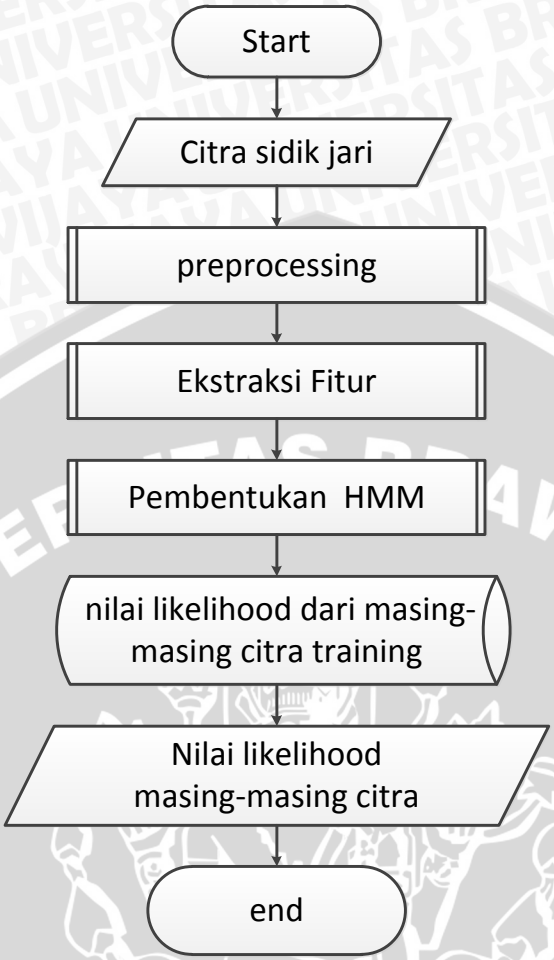
Pada bagian ini akan dipaparkan mengenai langkah-langkah yang akan dijalankan untuk melakukan pelatihan dan pengenalan citra sidik jari dari Verifinger_Sample_DB.

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 :



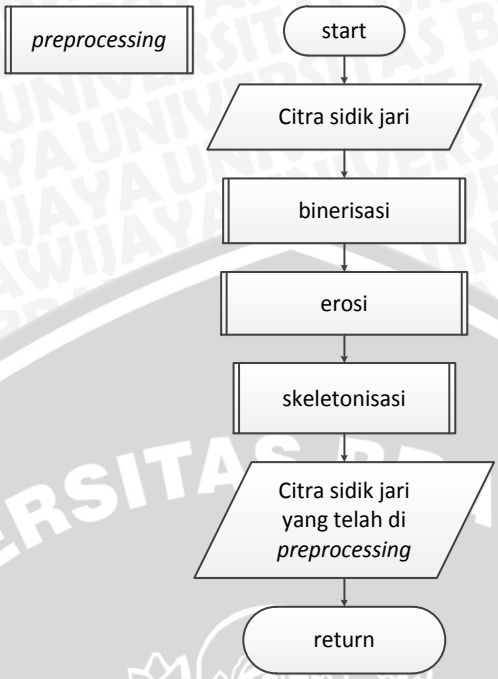
Gambar 3.1 Diagram alir *training*

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*

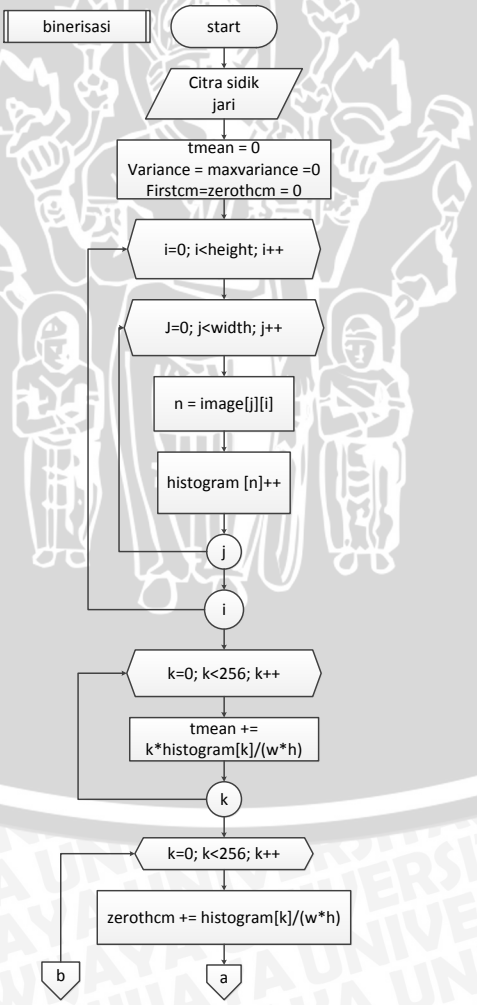
3.3.2.1 Perancangan Binerisasi

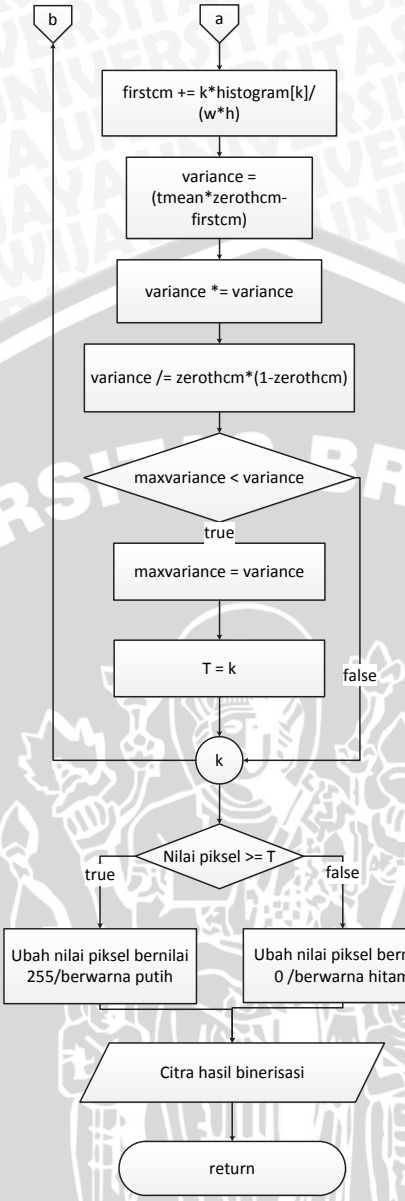
Langkah-langkah yang ada dalam proses binerisasi adalah sebagai berikut :

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

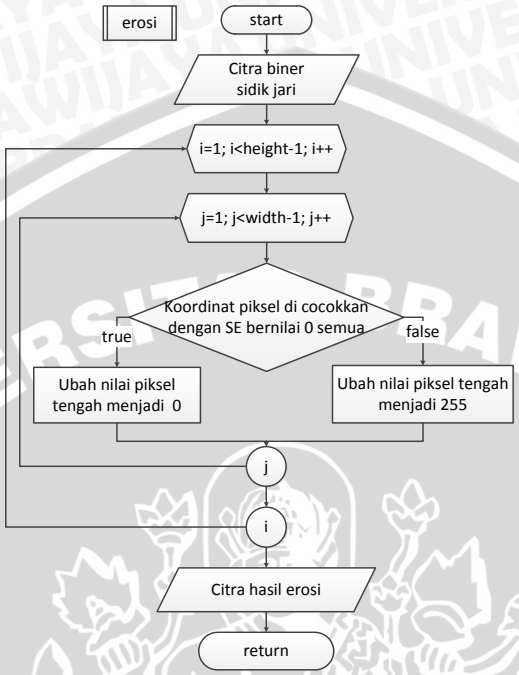
3.3.2.2 Perancangan Erosi

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

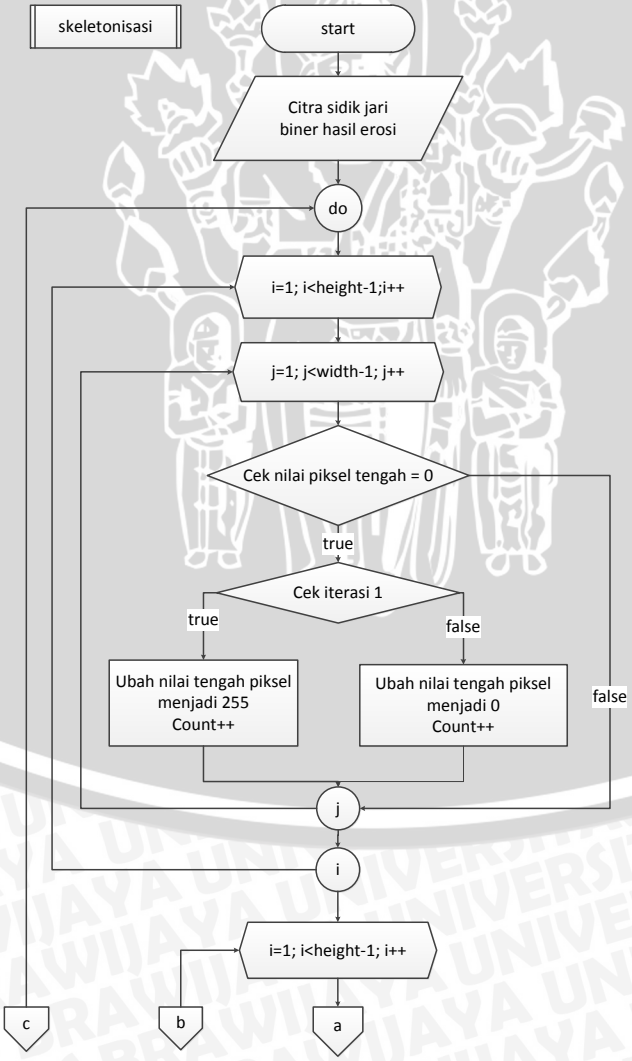
3.3.2.3 Perancangan Skeletonisasi

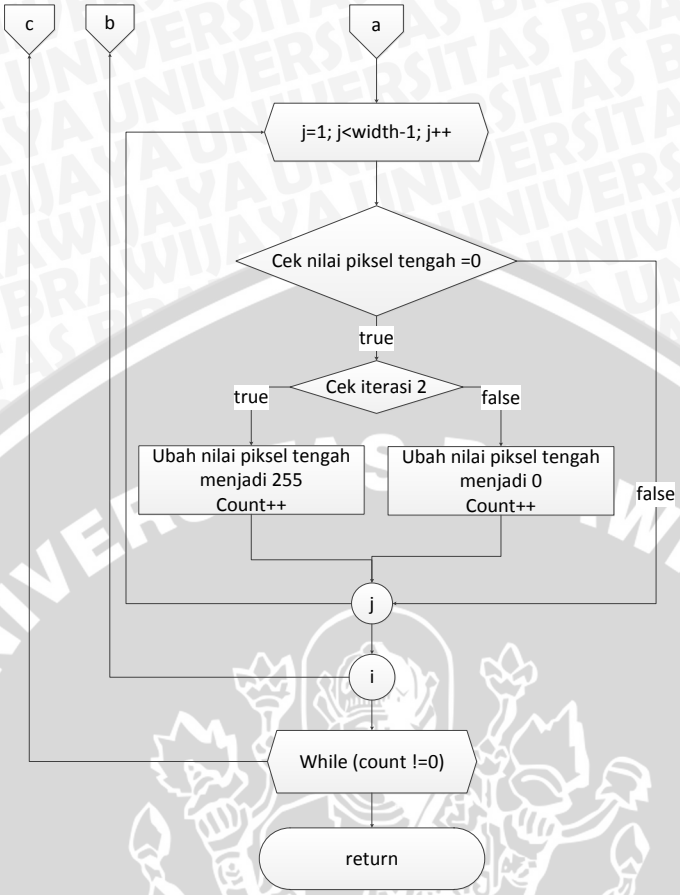
Langkah – langkah yang ada dalam proses skeletonisasi sebagai berikut :

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 :





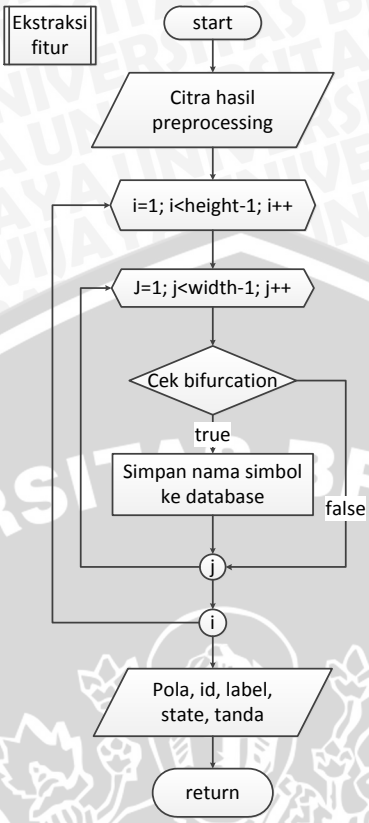
Gambar 3.5 Diagram alir proses skeletonisasi dengan algoritma Zhang-Suen

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

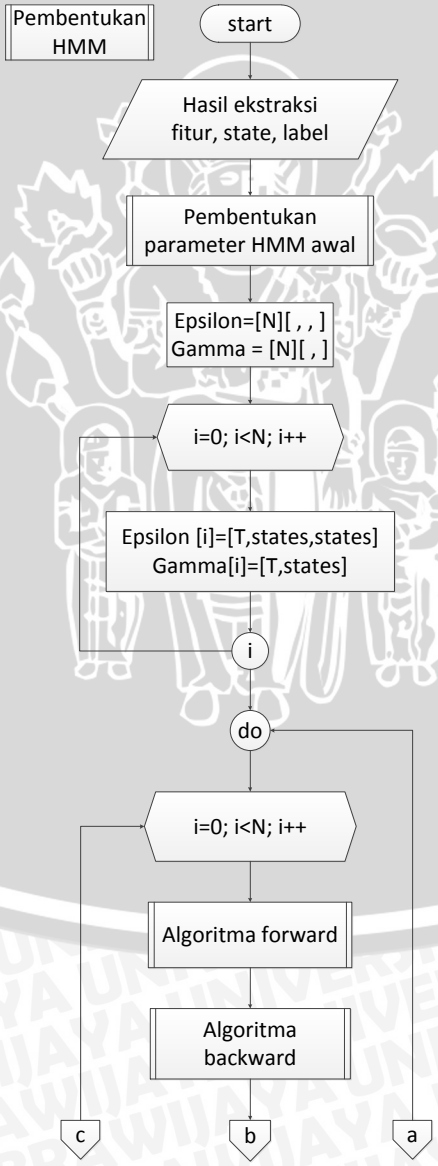
3.3.4 Perancangan Parameter HMM

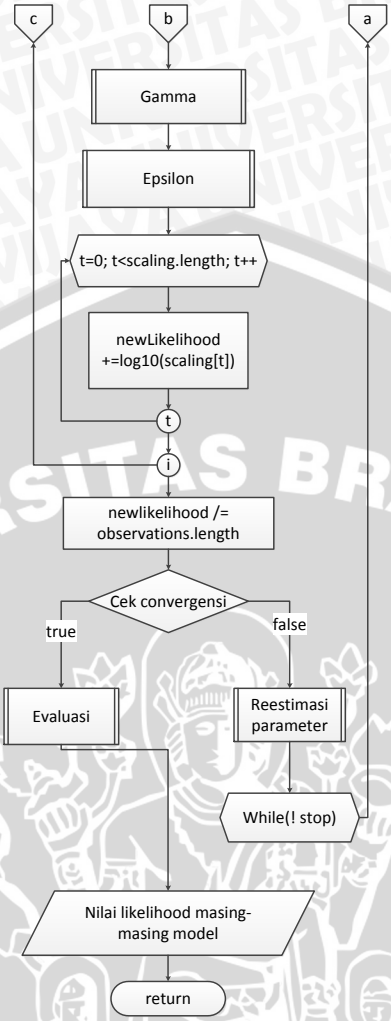
Langkah-langkah dalam proses parameter HMM adalah sebagai berikut :

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 :

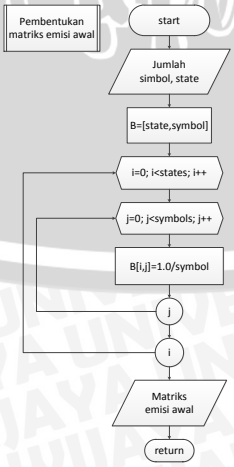
Gambar 3.8 Diagram alir parameter HMM awal

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 :



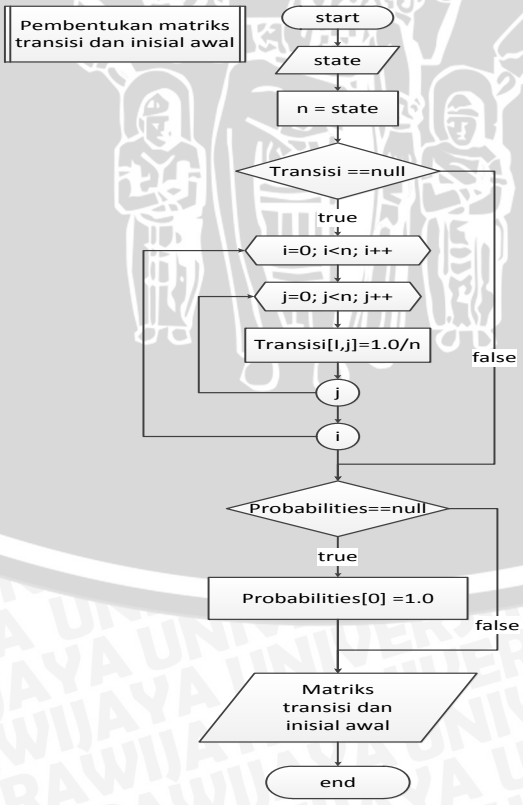
Gambar 3.9 Diagram alir matriks emisi awal

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 :



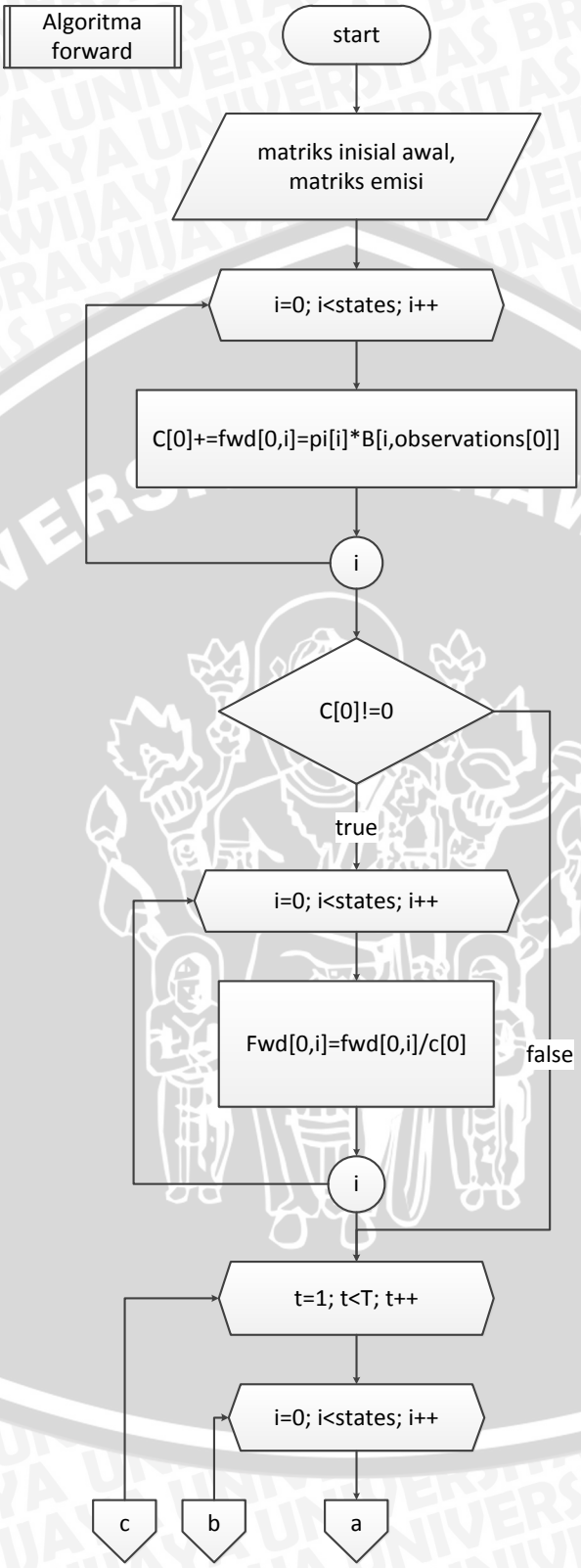
Gambar 3.10 Diagram alir matriks inisial dan transisi awal

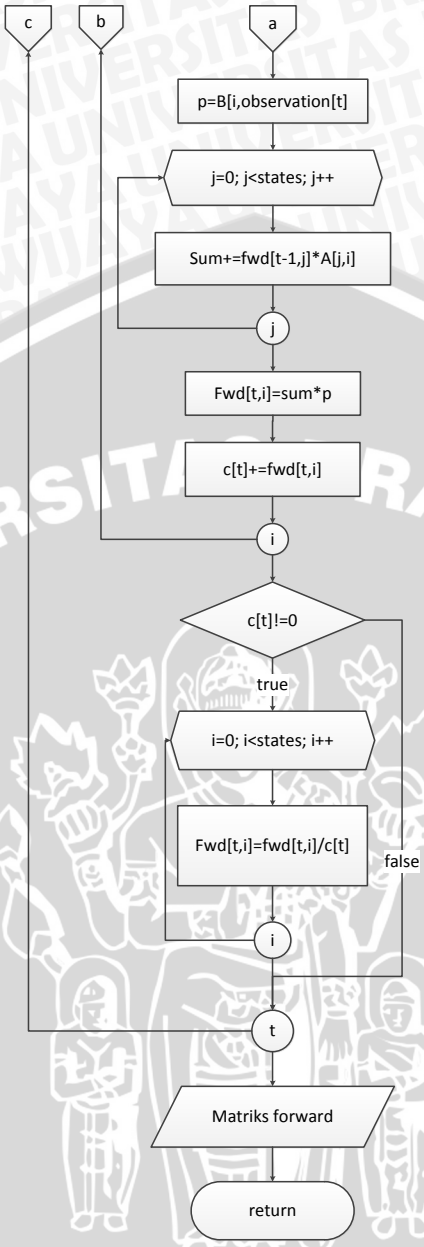
3.3.4.2 Perancangan Algoritma *Forward*

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*

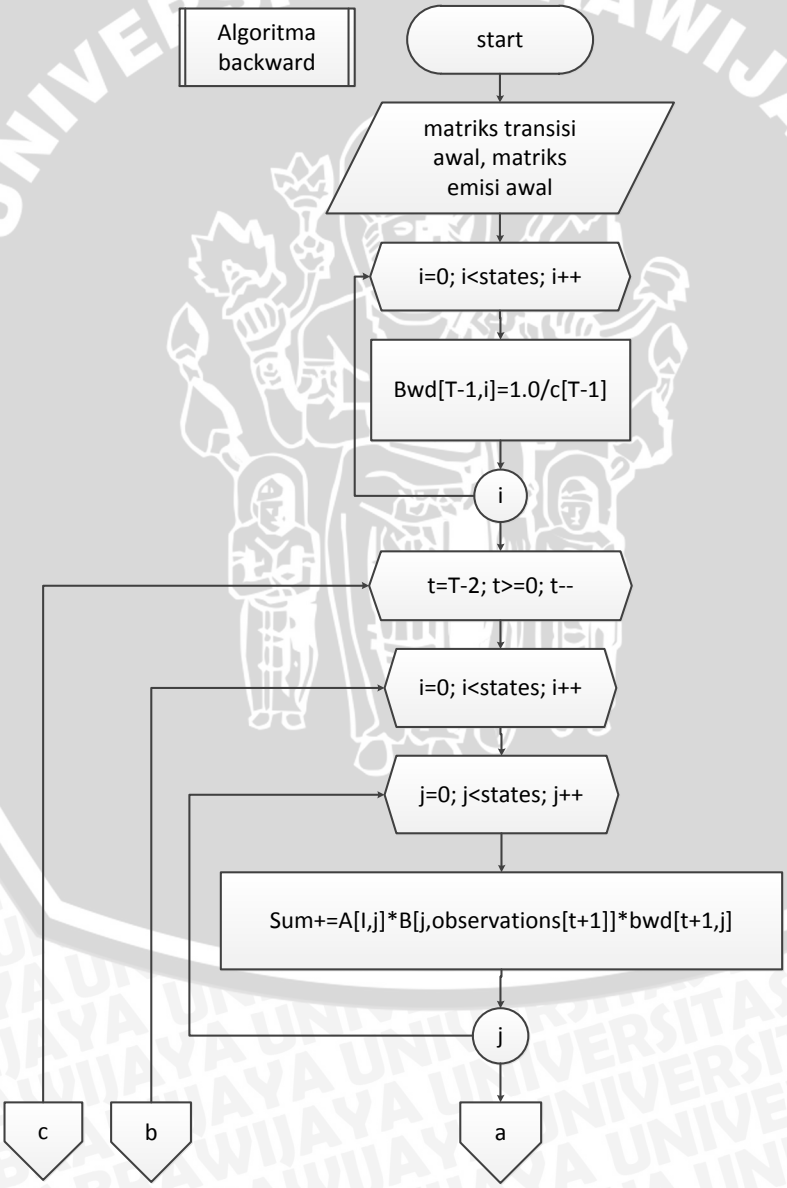
3.3.4.3 Perancangan Algoritma *Backward*

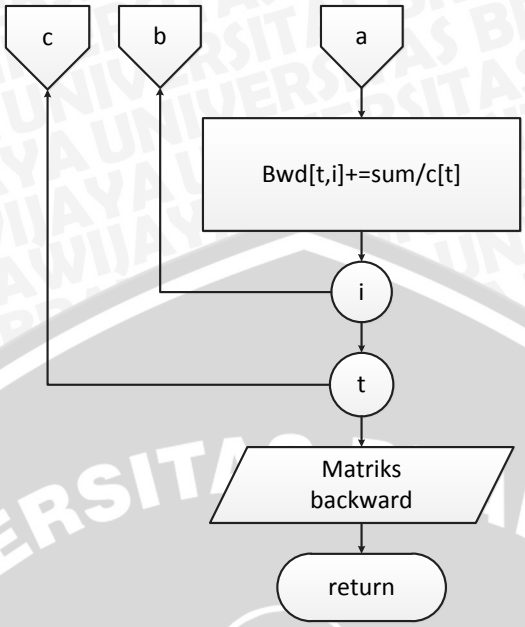
Langkah-langkah dalam proses algoritma *backward* adalah :

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 :





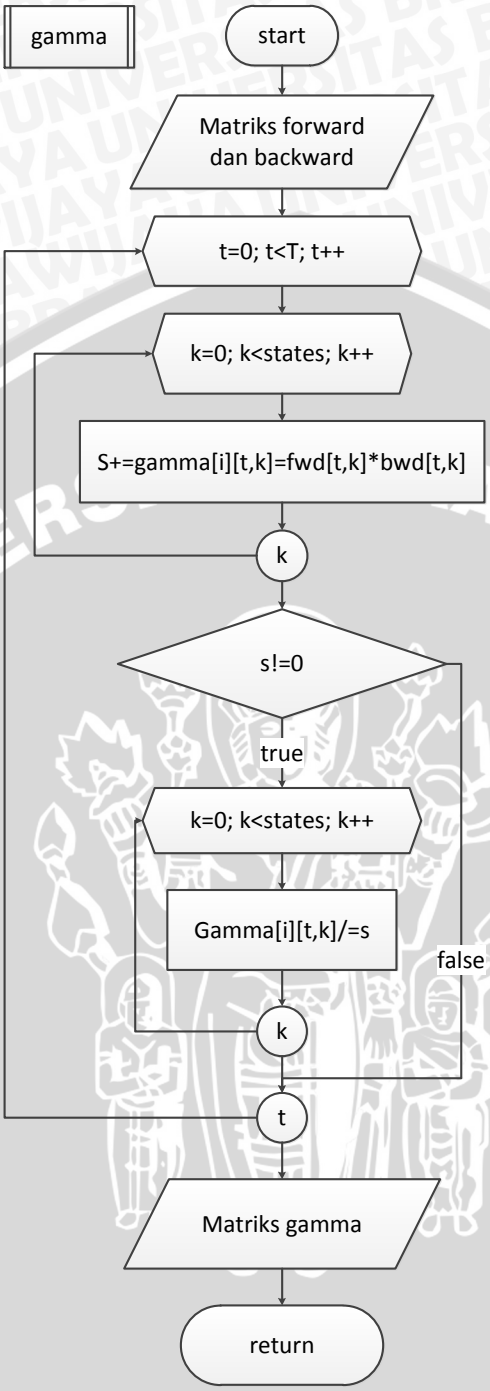
Gambar 3.12 Diagram alir algoritma *backward*

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*

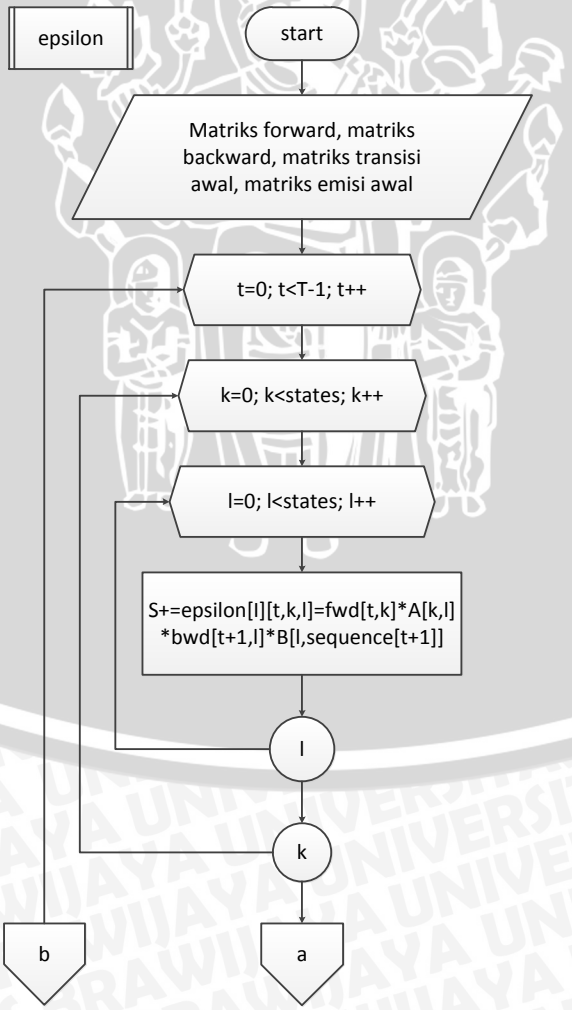
3.3.4.5 Perancangan Matriks *Epsilon*

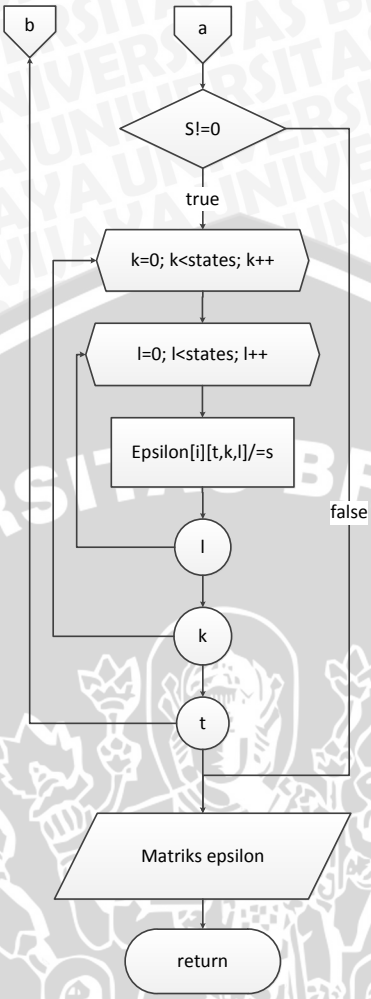
Langkah-langkah pada proses matriks *epsilon* adalah :

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 :





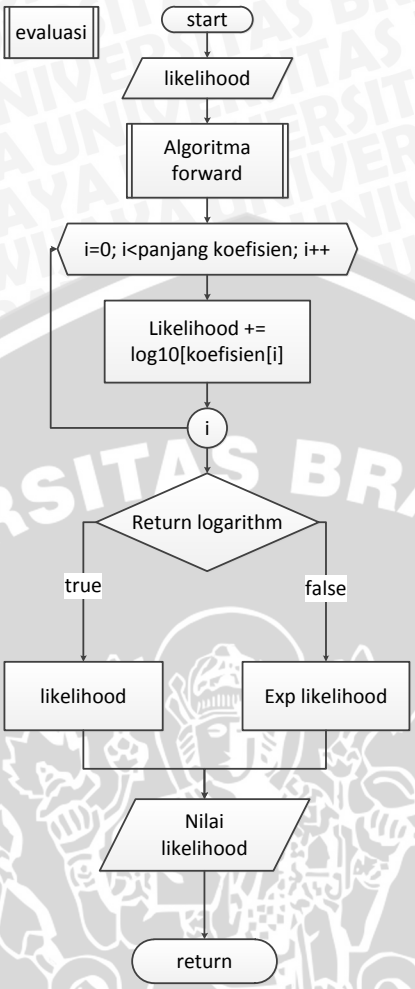
Gambar 3.14 Diagram alir matriks *epsilon*

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 :



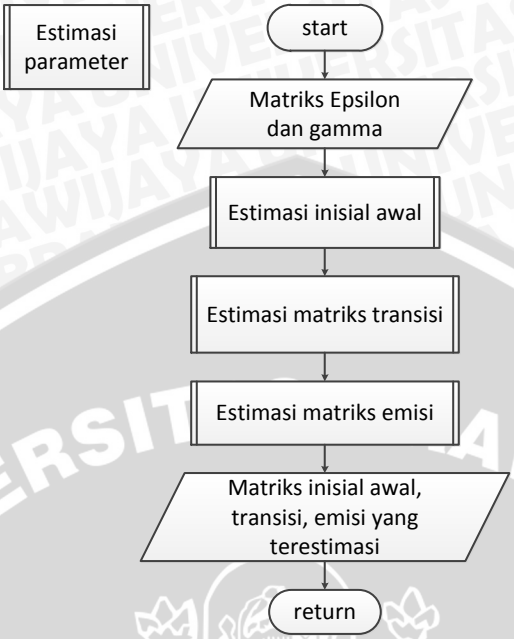
Gambar 3.15 Diagram alir evaluasi

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 :



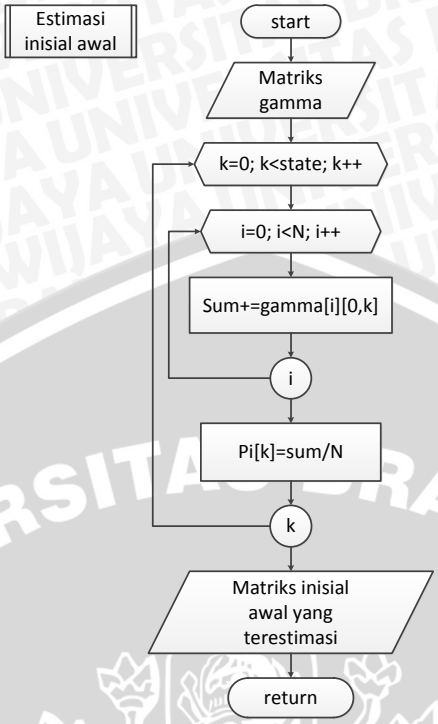
Gambar 3.16 Diagram alir re-estimasi parameter HMM

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 :



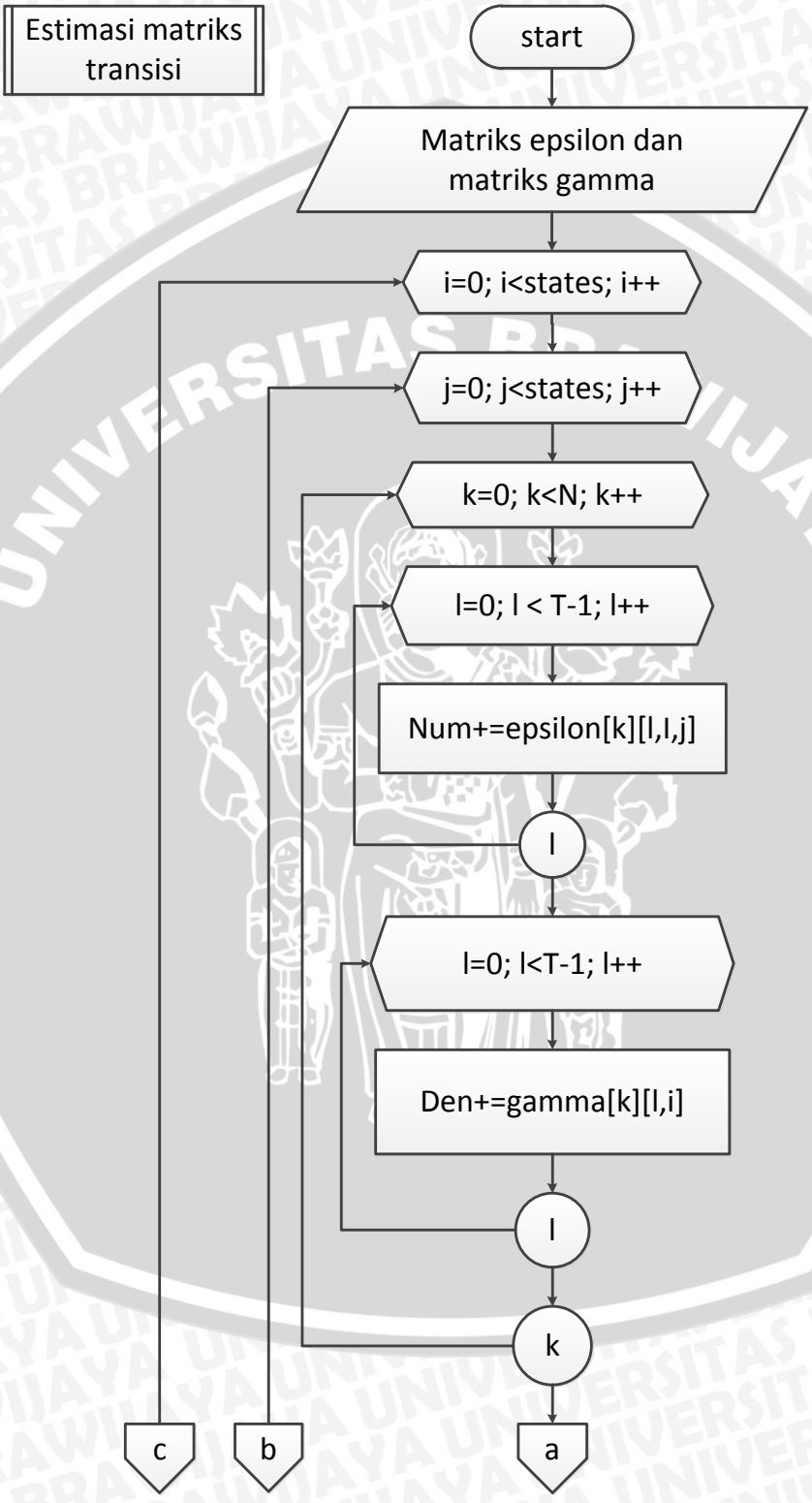
Gambar 3.17 Diagram alir re-estimasi matriks inisial awal

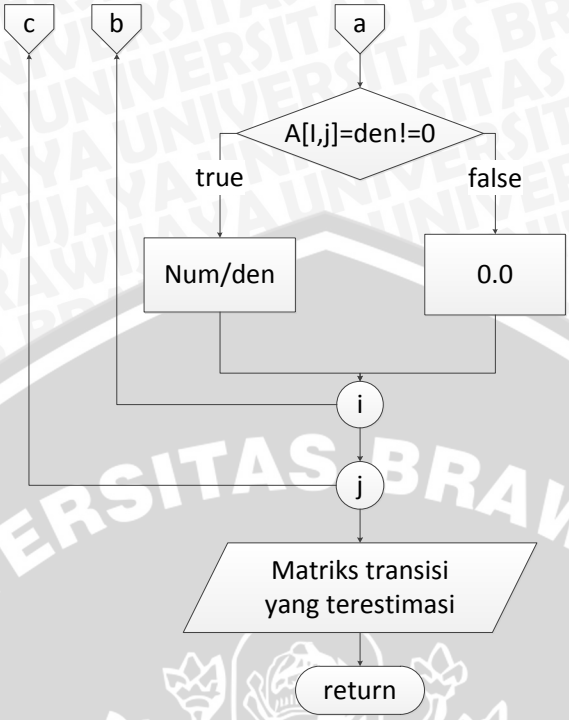
3.3.4.7.2Perancangan Re-estimasi Matriks Transisi

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 :





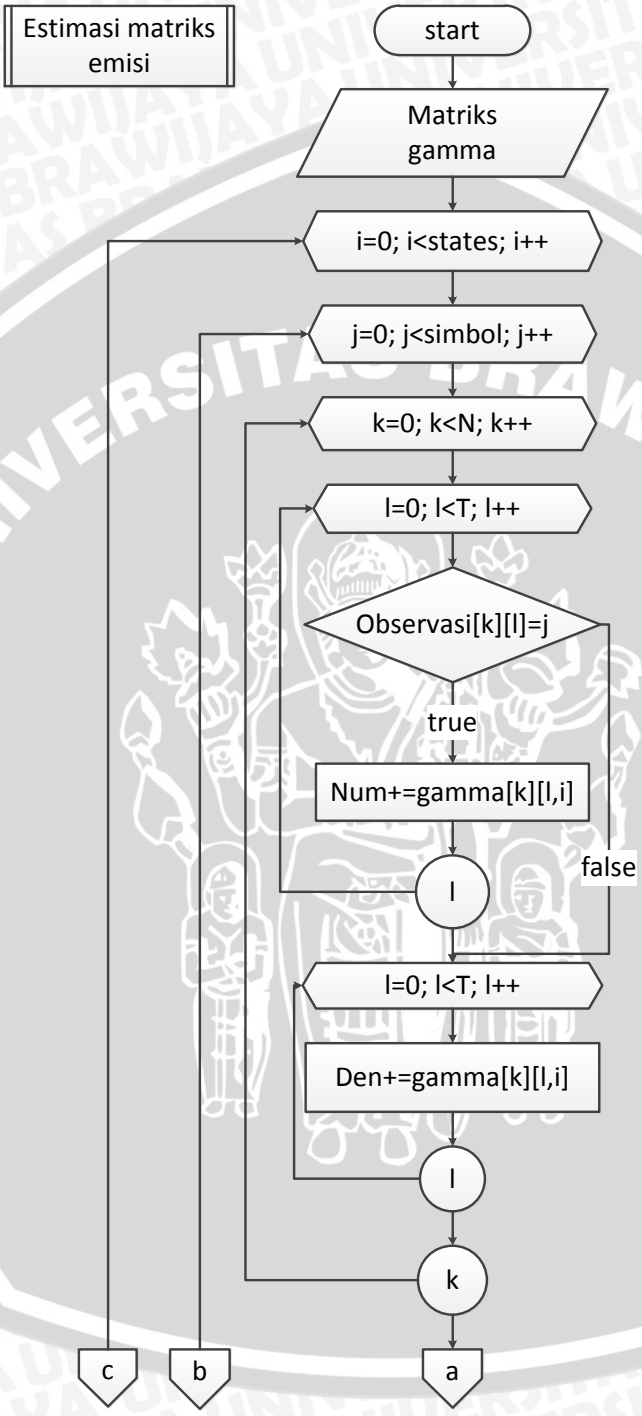
Gambar 3.18 Diagram alir re-estimasi matriks transisi

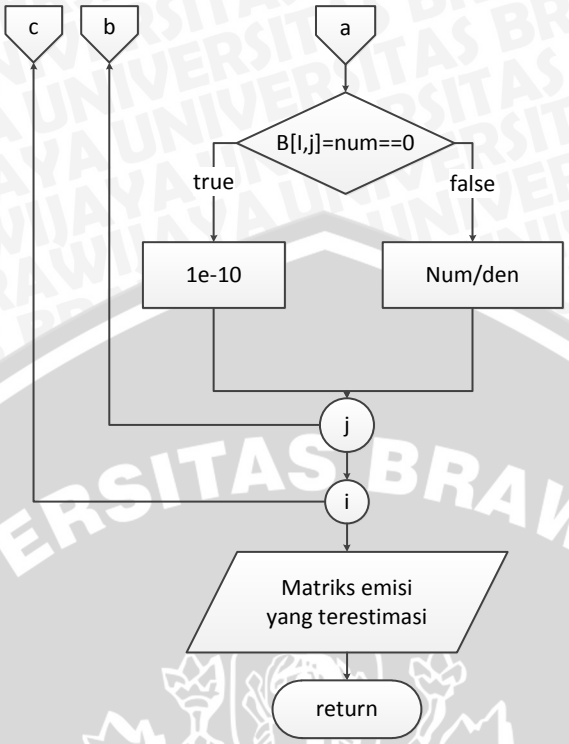
3.3.4.7.3Perancangan Reestimasi Matriks Emisi

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 :

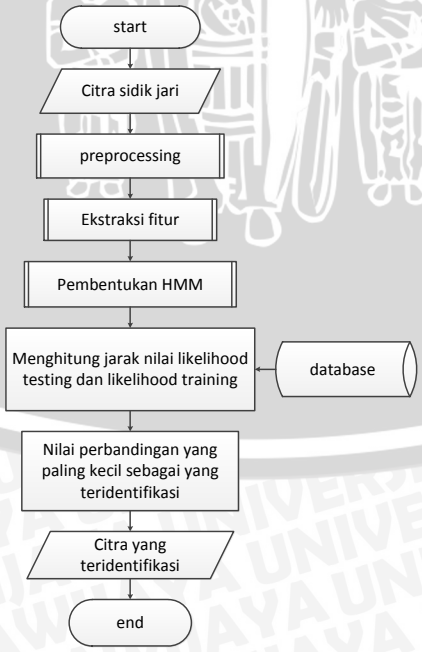




Gambar 3.19 Diagram alir re-estimasi matriks emisi

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.

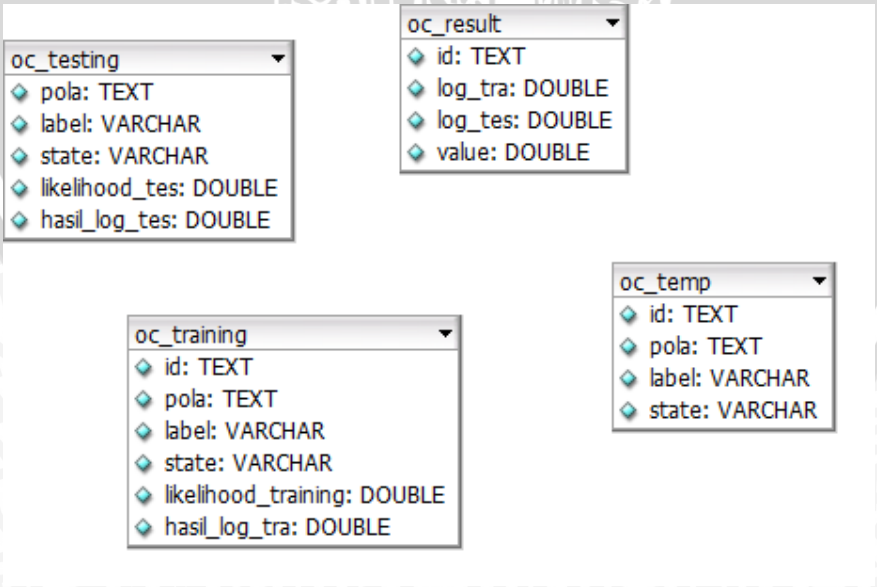


Gambar 3.20 Diagram alir sistem pengenalan sidik jari

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

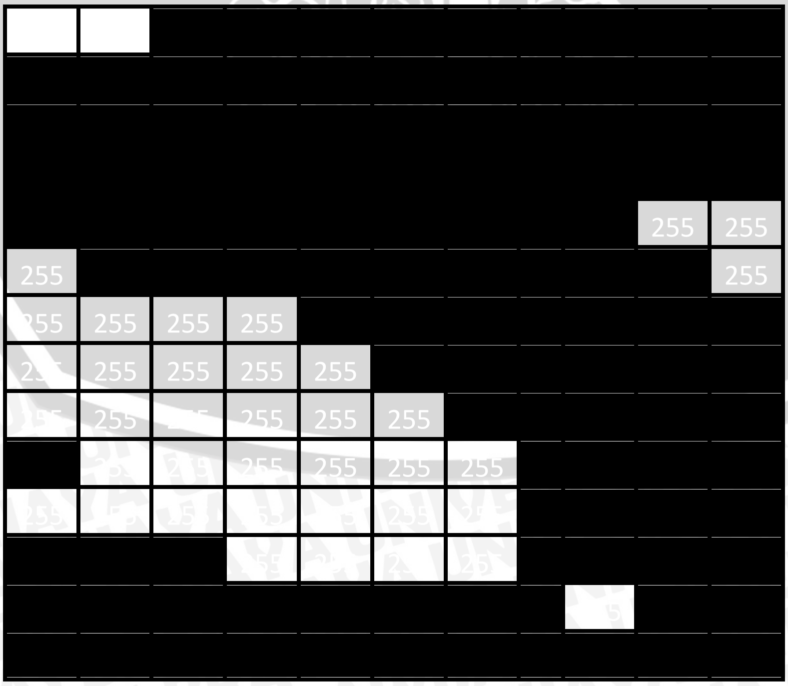
$\sigma_B^2(k^*) = max_{1 \le k \le L} \sigma_B^2(k)$

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.

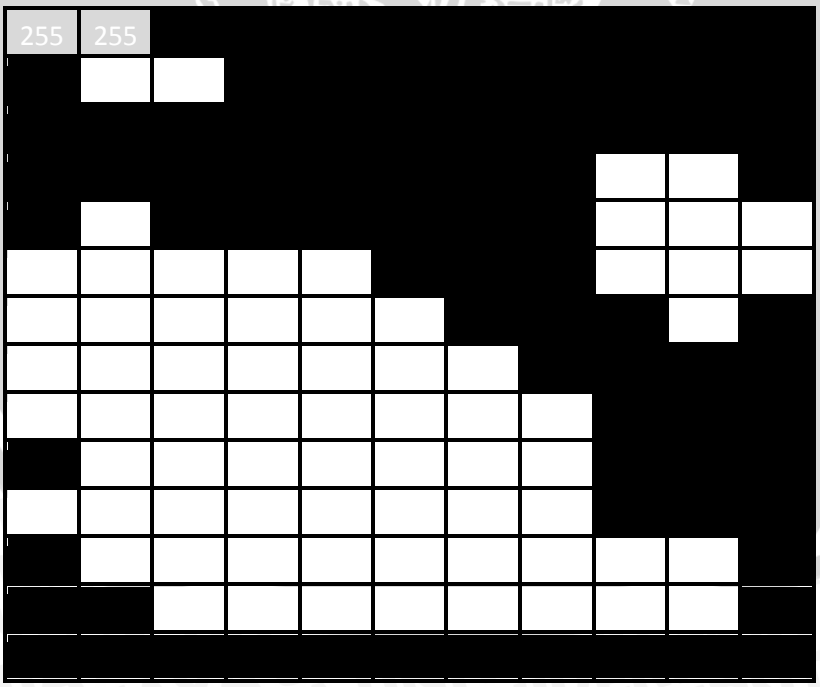
i-1,j-1	i,j-1	i+1,j-1
i-1,j	i,j	i+1,j
i-1,j+1	i,j+1	i+1,j+1

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

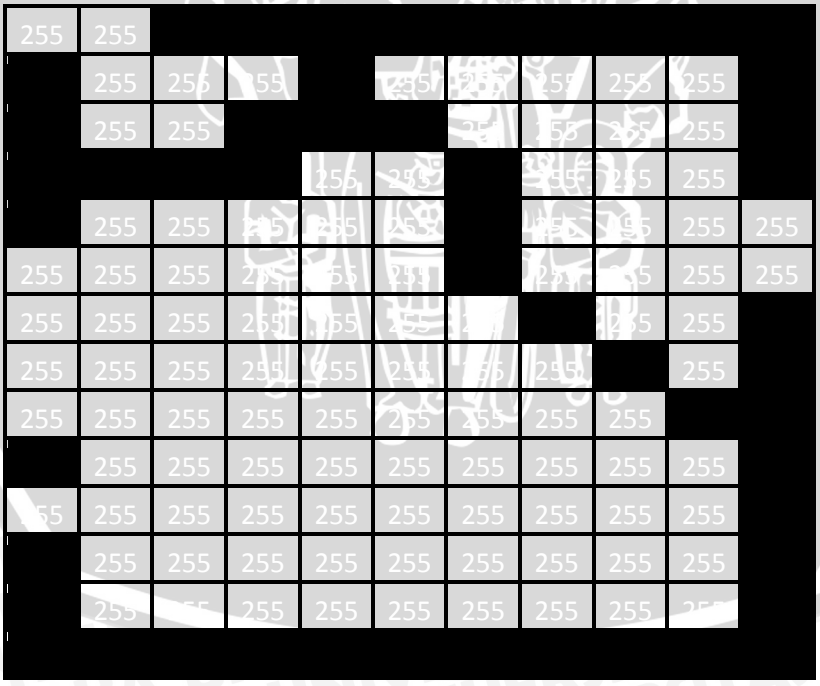
3.4.3 Proses Skeletonisasi

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.

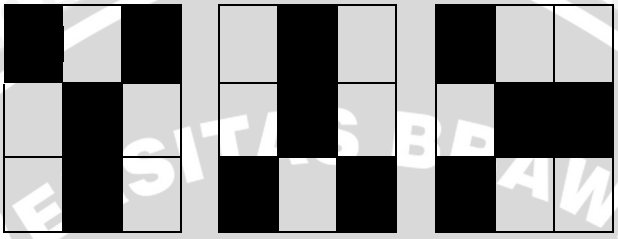
Dua iterasi ini dilakukan secara terus menerus sampai citra tidak mengalami perubahan struktur. Adapun contoh proses skeletonisasi dengan menggunakan metode Zhang-Suen terdapat pada gambar 3.27 sebagai berikut :



Gambar 3.27 Citra yang mengalami proses skeletonisasi

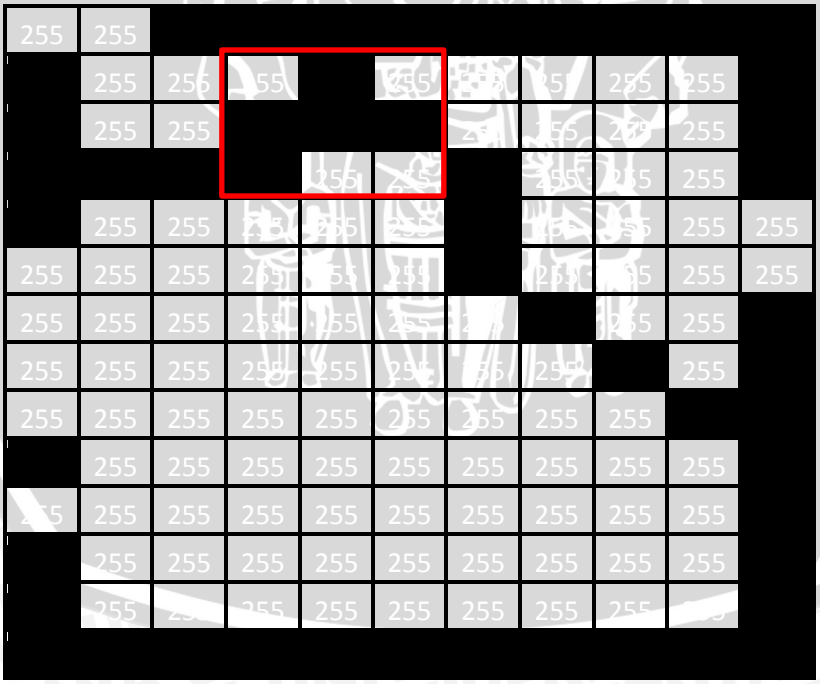
3.4.4 Proses Ekstraksi Fitur

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

$$c[0] \mathrel{+}= fwd[0,i] = pi[i] * B[0,observations[0]]$$
$$fwd[0,i] = fwd[0,i] / c[0]$$

$fwd[0,0] = 1 \times B[0,0]$

$= 1 \times 0{,}041667$

$= 0{,}041667$

$c[0] = 0 + 0{,}041667 = 0{,}041667$

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

$$p = B[i,observations[t]]$$
$$sum \mathrel{+}= fwd[t-1,j] * A[i,j]$$
$$fwd[t,i] = sum * p$$
$$c[t] \mathrel{+}= fwd\,[t,i]$$
$$fwd[t,i] = fwd[t,i] / c[t]$$

$p = B[0,observasi[1]] = 0{,}041667$

$sum = fwd[0,0] \times A[0,0]$

$= 1 \times 1 = 1$

$Sum = 0 + 1 = 1$

$fwd[0,1] = 1 \times 0{,}041667 = 0{,}041667$

$c[0] = 0{,}041667$

$c[0] = 0 + 0{,}041667 = 0{,}041667$

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

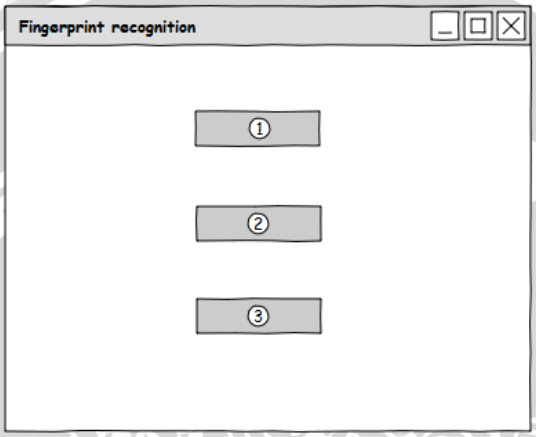
Dari hasil perhitungan diatas maka didapatkan nilai terkecil *0,56897* dengan begitu maka citra diatas dikenali sebagai citra kelas *E* dan dimiliki oleh orang dengan id *076*.

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.

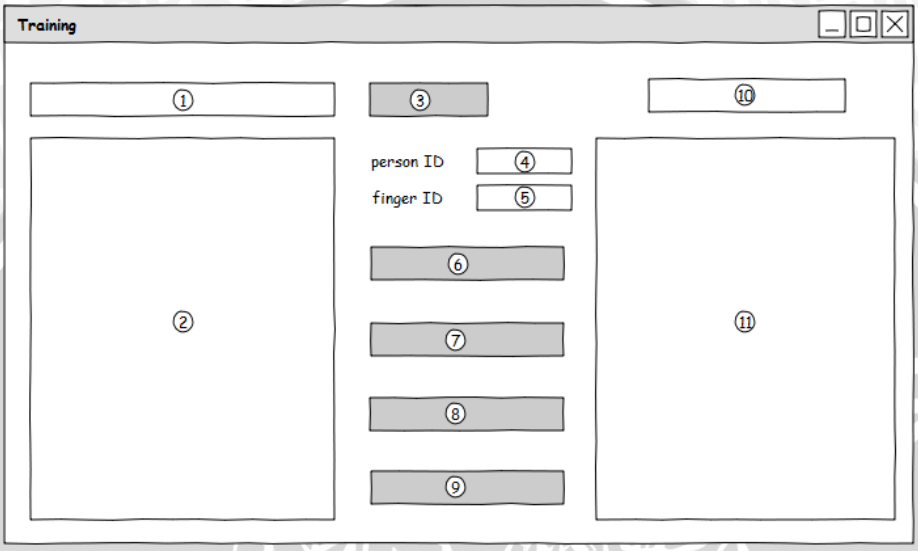
Pada Gambar 3.30 terdapat 3 tombol yaitu tombol *training* (1), tombol *testing* (2), dan tombol *exit* (3). Fungsi tombol *training* adalah untuk menampilkan program *training*. tombol *testing* untuk menampilkan program *testing*. tombol *exit* digunakan untuk keluar dari program pengenalan sidik jari.



Gambar 3.30. Tampilan awal program

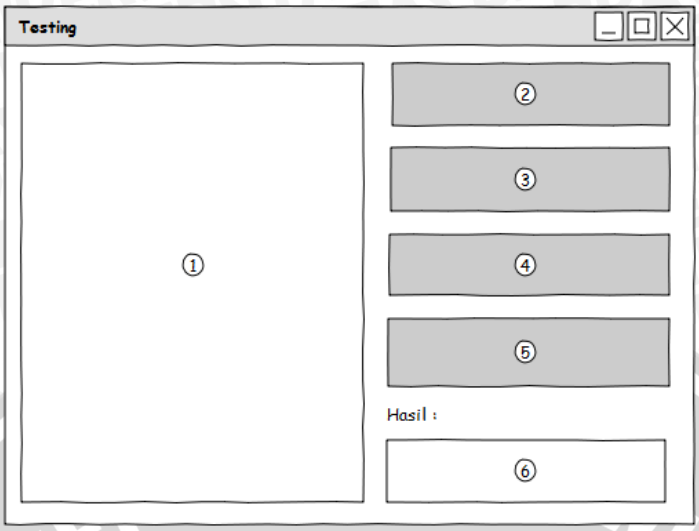
Pada Gambar 3.31 terdapat tombol *open*(3) yang berfungsi untuk membuka citra sidik jari yang akan di t*raining*. Tombol *preprocessing*(6) digunakan untuk mengubah citra sidik jari menjadi citra biner, memperbaiki morfologi citra, dan menjadikan citra sidik jari menjadi citra dengan ketebalan satu piksel. 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			

BAB IV
IMPLEMENTASI

Pada bab implementasi akan dibahas mengenai implementasi sistem yang dibuat baik itu perangkat keras maupun perangkat lunak, batasan-batasan implementasi, implementasi program dari proses binerisasi, proses erosi, proses skeletonisasi, sampai dengan parameter HMM, implementasi *database* dan implementasi *interface*.

4.1 Implementasi Sistem

Perangkat lunak pengenalan sidik jari dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

4.1.1 Implementasi Perangkat Keras

Implementasi perangkat keras yang dipakai dalam proses pembuatan sisitem dijelaskan pada tabel berikut ini :

Tabel 4.1 Implementasi perangkat keras komputer

Notebook Toshiba Satellite L635	
Nama *Hardware*	Spesifikasi
Processor	Intel (R) Core (TM) i5 M460 2.53 GHz
Memory (RAM)	2 GB
Harddisk	Seagate Momentus 5400.6 SATA 3Gb/s 500 GB
Motherboard	Toshiba Notebook Intel Motherboard
Graphic Card	ATI Mobility Radeon HD 5470
Monitor	13.3" Widescreen LED Backlit Display
Webcam	Toshiba Web Camera 1,3 MP

4.1.2 Implementasi Perangkat Lunak

Implementasi perangkat lunak yang dipakai dalam proses pembuatan sistem dijelaskan pada tabel berikut ini :

Tabel 4.2 Implementasi perangkat lunak komputer

Notebook Toshiba Satellite L635	
Nama *software*	Spesifikasi
Sistem operasi	Microsoft Windows 7 Ultimate 32-bit
Versi *DirectX*	DirectX 11

Bahasa pemrograman	C#
Integrated Development Environment	Microsoft Visual Studio 2008
Database Management System	MySQL XAMPP 1.7.7

4.2 Batasan – Batasan Implementasi

Beberapa batasan dalam mengimplementasikan perangkat lunak pengenalan sidik jari adalah sebagai berikut :

1. Perangkat lunak pengenalan sidik jari dirancang dan dijalankan dengan menggunakan *Desktop Application*.
2. *File* yang digunakan memiliki format **.tiff*.
3. *Database Management System* yang digunakan adalah MySQL.
4. Setiap *file* mempunyai *filename* xxx_y_z dimana x adalah *person ID*, y adalah *finger ID*, dan z adalah *number of scan*
5. Ukuran *width* dan *height* citra adalah 504 x 408 piksel.

4.3 Implementasi Program

Terdapat beberapa proses dalam program baik dalam program *training* maupun program *testing*. Adapun proses-prosesnya adalah sebagai berikut :

4.3.1 Implementasi Proses Binerisasi

Proses binerisasi ini bertujuan untuk mengubah citra awal yang berwarna grayscale menjadi citra berwarna hitam dan putih. Proses implementasi proses binerisasi seperti yang ditunjukkan pada *Sourcecode* 4.1 di bawah ini :

Sourcecode 4.1 Implementasi proses binerisasi

```
1   red = green = blue = new int[bmp.Width, bmp.Height];
2   for (int i = 0; i < bmp.Width; i++)
3   {
4   for (int j = 0; j < bmp.Height; j++)
5   {
6   red[i, j] = bmp.GetPixel(i, j).R;
7   green[i, j] = bmp.GetPixel(i, j).G;
8   blue[i, j] = bmp.GetPixel(i, j).B;
9   }
10  }
11
12  for (int i = 0; i < bmp.Width; i++)
13  for (int j = 0; j < bmp.Height; j++)
```

```
{
p = red[i, j];
histogram[p]++;
}
box1_img_tes.Image = bmp;
bmp = new Bitmap(box1_img_tes.Image);

for (k = 0; k < 256; k++)
tmean += k * histogram[k] / (bmp.Width * bmp.Height);
for (k = 0; k < 256; k++)
{
zerothcm += histogram[k] / (bmp.Width * bmp.Height);
firstcm += k * histogram[k] / (bmp.Width * bmp.Height);
variance = (tmean * zerothcm - firstcm);
variance *= variance;
variance /= zerothcm * (1 - zerothcm);
if (maxvariance < variance)
{
maxvariance = variance;
t = k;
}
}

for (int i = 0; i < bmp.Width; i++)
for (int j = 0; j < bmp.Height; j++)
{
p = red[i, j];
if (p >= t) bmp.SetPixel(i, j, Color.FromArgb(255, 255,
255));
else bmp.SetPixel(i, j, Color.FromArgb(0, 0, 0));
}
```

Penjelasan dari *Sourcecode* 4.1 proses binerisasi sebagai berikut ini :

Baris 1-11 adalah proses memasukkan ukuran citra kedalam array kemudian ditelusuri.

Baris 13-20 adalah proses pembuatan histogram.

Baris 22-39 adalah metode otsu dengan mencari nilai variance tertinggi dan diambil nilai k untuk dijadikan threshold.

Baris 41-49 adalah proses merubah gambar menjadi bernilai 0 dan 255.

4.3.2 Implementasi Proses Erosi

Proses Erosi bertujuan untuk menghilangkan noise-noise pada citra dan untuk memperbaiki struktur citra agar terlihat lebih baik. Implementasi proses erosi ditunjukkan pada *Sourcecode* 4.2 berikut ini :

Sourcecode 4.2 Implementasi proses erosi

```
private Bitmap erosi(Bitmap bmp)
{
red = green = blue = new int[bmp.Width, bmp.Height];
```

```
4   for (int i = 1; i < bmp.Width - 1; i++)
5   {
6   for (int j = 1; j < bmp.Height - 1; j++)
7   {
8   red[i, j] = bmp.GetPixel(i, j).R;
9   green[i, j] = bmp.GetPixel(i, j).G;
10  blue[i, j] = bmp.GetPixel(i, j).B;
11  }
12  }
13
14  for (int c = 0; c < 1; c++)
15  {
16  bmp1 = new Bitmap(bmp);
17  for (int i = 1; i < bmp.Width - 1; i++)
18  for (int j = 1; j < bmp.Height - 1; j++)
19  {
20  if (red[i - 1, j - 1] == 0 && red[i - 1, j] == 0 && red[i -
21  1, j + 1] == 0 && red[i, j - 1] == 0 && red[i, j] == 0 &&
22  red[i, j + 1] == 0 && red[i + 1, j - 1] == 0 && red[i + 1,
23  j] == 0 && red[i + 1, j + 1] == 0)
24  bmp1.SetPixel(i, j, Color.FromArgb(0, 0, 0));
25  else bmp1.SetPixel(i, j, Color.FromArgb(255, 255, 255));
26  }
27  bmp = new Bitmap(bmp1);
28  box1_img_tes.Image = (Image)bmp;
29  }
30  return bmp;
31  }
```

Penjelasan dari *Sourcecode* 4.2 proses erosi sebagai berikut ini :

Baris 3-13 adalah proses mengambil dan memasukkan nilai piksel citra kedalam array.

Baris 15-36 adalah inti dari proses erosi yaitu proses pengecekan antara nilai piksel 8-ketetanggaan dengan nilai SE.

4.3.3 Implementasi Proses Skeletonisasi

Proses skeletonisasi adalah proses mengubah citra menjadi setebal 1 piksel. Proses ini bertujuan untuk memudahkan dalam mencari ekstraksi fitur dan nama metode yang digunakan dalam skeletonisasi adalah metode Zhang-Suen. Adapun proses implmentasi dari proses skeletonisasi ditunjukkan pada *Sourcecode* 4.3 berikut ini :

Sourcecode 4.3 Implementasi proses skeletonisasi

```
1   private void skeletonisasi_tes_Click(object sender,
2   EventArgs e)
3   {
4   int count = 0;
5   bmp1 = new Bitmap(bmp);
6   do
```

```
{
count = 0;
for (int j = 1; j < bmp.Height - 1; j++)
for (int i = 1; i < bmp.Width - 1; i++)
{
if (bmp.GetPixel(i, j).R == 0)
{
if (cekbp(i, j) == true && ap(i, j) == true && cek1(i, j) ==
true && cek2(i, j) == true)
{
bmp1.SetPixel(i, j, Color.White);
count++;
}
if (cekbp(i, j) == true && ap(i, j) == true && cek3(i, j) ==
true && cek4(i, j) == true)
{
bmp1.SetPixel(i, j, Color.White);
count++;
}
}
}
bmp = new Bitmap(bmp1);
}
while (count != 0);
box1_img_tes.Image = (Image)bmp;
}
private bool cekbp(int x, int y)
{
int jumlahbp = 0;
for (int i = x - 1; i <= x + 1; i++)
for (int j = y - 1; j <= y + 1; j++)
{
if (i != x && j != y)
if (bmp.GetPixel(i, j).R != 255)
jumlahbp++;
}
if (jumlahbp >= 2 && jumlahbp <= 6)
return true;
else return false;
}

private bool ap(int x, int y)
{
int perubahan = 0;
if (bmp.GetPixel(x, y - 1).R == 255 && bmp.GetPixel(x + 1, y
- 1).R == 0)
perubahan++;
if (bmp.GetPixel(x + 1, y - 1).R == 255 && bmp.GetPixel(x +
1, y).R == 0)
perubahan++;
if (bmp.GetPixel(x + 1, y).R == 255 && bmp.GetPixel(x + 1, y
+ 1).R == 0)
perubahan++;
if (bmp.GetPixel(x + 1, y + 1).R == 255 && bmp.GetPixel(x, y
+ 1).R == 0)
perubahan++;
if (bmp.GetPixel(x, y + 1).R == 255 && bmp.GetPixel(x - 1, y
```

```
64  + 1).R == 0)
65  perubahan++;
66  if (bmp.GetPixel(x - 1, y + 1).R == 255 && bmp.GetPixel(x -
67  1, y).R == 0)
68  perubahan++;
69  if (bmp.GetPixel(x - 1, y).R == 255 && bmp.GetPixel(x, y -
70  1).R == 0)
71  perubahan++;
72  if (bmp.GetPixel(x - 1, y - 1).R == 255 && bmp.GetPixel(x -
73  1, y).R == 0)
74  perubahan++;
75  if (perubahan == 1)
76  return true;
77  else return false;
78  }
79
80  private bool cek1(int x, int y)
81  {
82  if (bmp.GetPixel(x, y - 1).R == 0 && bmp.GetPixel(x + 1,
83  y).R == 0 && bmp.GetPixel(x, y + 1).R == 0 ||
84  bmp.GetPixel(x, y - 1).R == 255 && bmp.GetPixel(x + 1, y).R
85  == 255 && bmp.GetPixel(x, y + 1).R == 255)
86  return false;
87  else return true;
88  }
89
90  private bool cek2(int x, int y)
91  {
92  if (bmp.GetPixel(x + 1, y).R == 0 && bmp.GetPixel(x, y +
93  1).R == 0 && bmp.GetPixel(x - 1, y).R == 0 ||
94  bmp.GetPixel(x + 1, y).R == 255 && bmp.GetPixel(x, y + 1).R
95  == 255 && bmp.GetPixel(x - 1, y).R == 255)
96  return false;
97  else return true;
98  }
99
100 private bool cek3(int x, int y)
101 {
102 if (bmp.GetPixel(x, y - 1).R == 0 && bmp.GetPixel(x + 1,
103 y).R == 0 && bmp.GetPixel(x - 1, y).R == 0 ||
104 bmp.GetPixel(x, y - 1).R == 255 && bmp.GetPixel(x + 1, y).R
105 == 255 && bmp.GetPixel(x - 1, y).R == 255)
106 return false;
107 else return true;
108 }
109
110 private bool cek4(int x, int y)
111 {
112 if (bmp.GetPixel(x, y - 1).R == 0 && bmp.GetPixel(x, y +
113 1).R == 0 && bmp.GetPixel(x - 1, y).R == 0 ||
114 bmp.GetPixel(x, y - 1).R == 255 && bmp.GetPixel(x, y + 1).R
115 == 255 && bmp.GetPixel(x - 1, y).R == 255)
116 return false;
117 else return true;
118 }
```

Penjelasan *Sourcecode* 4.3 proses skeletonisasi sebagai berikut :

Baris 1-33 adalah proses utama dalam skeletonisasi.

Baris 35-48 adalah proses pengecekan jumlah 255 lebih dari sama dengan 2 atau kurang dari sama dengan 6 jika benar maka true jika salah false

Baris 50-80 adalah pengecekan perpindahan dari 255 ke 0 jika lebih dari satu maka true jika tidak maka false

Baris 82-91 adalah pengecekan nilai P2*P4*P6=0

Baris 93-103 adalah pengecekan nilai P4*P6*P8=0

Baris 105-114 adalah pengecekan nilai P2*P4*P8=0

Baris 116-126 adalah pengeceken nilai P2*P6*P8=0

4.3.4 Implementasi Proses Ekstraksi Fitur

Proses ekstraksi fitur bertujuan untuk mengambil berbagai macam fitur yang ada pada semua citra sidik jari. Dalam proses ini ekstraksi fitur menggunakan percabangan (*bifurcation*) sejumlah 24 kemungkinan *bifurcation*. Adapun proses implementasi untuk proses ekstraksi fitur ditunjukkan pada *Sourcecode* 4.4 berikut ini :

Sourcecode 4.4 Implementasi proses ekstraksi fitur

```
1   private void EF_testing_Click(object sender, EventArgs e)
2   {
3   red = green = blue = new int[bmp.Width, bmp.Height];
4   for (int i = 1; i < bmp.Height - 1; i++)
5   {
6   for (int j = 1; j < bmp.Width - 1; j++)
7   {
8   red[j, i] = bmp.GetPixel(j, i).R;
9   green[j, i] = bmp.GetPixel(j, i).G;
10  blue[j, i] = bmp.GetPixel(j, i).B;
11  }
12  }
13
14  bmp1 = new Bitmap(bmp);
15  String pola = "";
16  int count = 0;
17  for (int i = 1; i < bmp.Height - 1; i++)
18  for (int j = 1; j < bmp.Width - 1; j++)
19  {
20  if (cek_bif(0, 255, 0, 255, 0, 255, 255, 0, 255, j, i) == 9)
21  {
22  pola += "0,";
23  rectangle(bmp1, j, i, 3, 3);
24  count++;
25  }
```

```
26  else if (cek_bif(255, 0, 255, 255, 0, 255, 0, 255, 0, j, i)
27  == 9)
28  {
29  pola += "1,";
30  rectangle(bmp1, j, i, 3, 3);
31  count++;
32  }
33  else if (cek_bif(0, 255, 255, 255, 0, 0, 0, 255, 255, j, i)
34  == 9)
35  {
36  pola += "2,";
37  rectangle(bmp1, j, i, 3, 3);
38  count++;
39  }
40  else if (cek_bif(255, 255, 0, 0, 0, 255, 255, 255, 0, j, i)
41  == 9)
42  {
43  pola += "3,";
44  rectangle(bmp1, j, i, 3, 3);
45  count++;
46  }
47  else if (cek_bif(255, 0, 255, 0, 255, 0, 255, 255, 255, j,
48  i) == 9)
49  {
50  pola += "4,";
51  rectangle(bmp1, j, i, 3, 3);
52  count++;
53  }
54  else if (cek_bif(255, 255, 255, 0, 255, 0, 255, 0, 255, j,
55  i) == 9)
56  {
57  pola += "5,";
58  rectangle(bmp1, j, i, 3, 3);
59  count++;
60  }
61  else if (cek_bif(255, 0, 255, 0, 255, 255, 255, 0, 255, j,
62  i) == 9)
63  {
64  pola += "6,";
65  rectangle(bmp1, j, i, 3, 3);
66  count++;
67  }
68  else if (cek_bif(255, 0, 255, 255, 255, 0, 255, 0, 255, j,
69  i) == 9)
70  {
71  pola += "7,";
72  rectangle(bmp1, j, i, 3, 3);
73  count++;
74  }
75  else if (cek_bif(0, 255, 0, 255, 0, 255, 0, 255, 255, j, i)
76  == 9)
77  {
78  pola += "8,";
79  rectangle(bmp1, j, i, 3, 3);
80  count++;
81  }
82  else if (cek_bif(0, 255, 255, 255, 0, 255, 0, 255, 0, j, i)
```

```
83  == 9)
84  {
85  pola += "9,";
86  rectangle(bmp1, j, i, 3, 3);
87  count++;
88  }
89  else if (cek_bif(0, 255, 0, 255, 0, 255, 255, 255, 0, j, i)
90  == 9)
91  {
92  pola += "10,";
93  rectangle(bmp1, j, i, 3, 3);
94  count++;
95  }
96  else if (cek_bif(255, 255, 0, 255, 0, 255, 0, 255, 0, j, i)
97  == 9)
98  {
99  pola += "11,";
100 rectangle(bmp1, j, i, 3, 3);
101 count++;
102 }
103 else if (cek_bif(255, 0, 255, 0, 255, 0, 0, 255, 255, j, i)
104 == 9)
105 {
106 pola += "12,";
107 rectangle(bmp1, j, i, 3, 3);
108 count++;
109 }
110 else if (cek_bif(255, 0, 255, 0, 255, 0, 255, 255, 0, j, i)
111 == 9)
112 {
113 pola += "13,";
114 rectangle(bmp1, j, i, 3, 3);
115 count++;
116 }
117 else if (cek_bif(0, 225, 255, 0, 255, 0, 255, 0, 255, j, i)
118 == 9)
119 {
120 pola += "14,";
121 rectangle(bmp1, j, i, 3, 3);
122 count++;
123 }
124 else if (cek_bif(255, 255, 0, 0, 255, 0, 255, 0, 255, j, i)
125 == 9)
126 {
127 pola += "15,";
128 rectangle(bmp1, j, i, 3, 3);
129 count++;
130 }
131 else if (cek_bif(255, 255, 0, 0, 0, 255, 255, 0, 255, j, i)
132 == 9)
133 {
134 pola += "16,";
135 rectangle(bmp1, j, i, 3, 3);
136 count++;
137 }
138 else if (cek_bif(255, 0, 255, 255, 0, 0, 0, 255, 255, j, i)
139 == 9)
```

```
{
pola += "17,";
rectangle(bmp1, j, i, 3, 3);
count++;
}
else if (cek_bif(255, 0, 255, 0, 0, 255, 255, 255, 0, j, i)
== 9)
{
pola += "18,";
rectangle(bmp1, j, i, 3, 3);
count++;
}
else if (cek_bif(0, 255, 255, 255, 0, 0, 255, 0, 255, j, i)
== 9)
{
pola += "19,";
rectangle(bmp1, j, i, 3, 3);
count++;
}
else if (cek_bif(255, 0, 255, 255, 255, 0, 0, 0, 255, j, i)
== 9)
{
pola += "20,";
rectangle(bmp1, j, i, 3, 3);
count++;
}
else if (cek_bif(0, 0, 255, 255, 255, 0, 255, 0, 255, j, i)
== 9)
{
pola += "21,";
rectangle(bmp1, j, i, 3, 3);
count++;
}
else if (cek_bif(255, 0, 0, 0, 255, 255, 255, 0, 255, j, i)
== 9)
{
pola += "22,";
rectangle(bmp1, j, i, 3, 3);
count++;
}
else if (cek_bif(255, 0, 255, 0, 255, 255, 255, 0, 0, j, i)
== 9)
{
pola += "23,";
rectangle(bmp1, j, i, 3, 3);
count++;
}
}
pola = pola.Substring(0, pola.Length - 1);
database(pola);
box1_img_tes.Image = (Image)bmp1;
}

private int cek_bif(int b1, int b2, int b3, int b4, int b5,
int b6, int b7, int b8, int b9, int x, int y)
{
int benar = 0;
```

```
197  if (red[x - 1, y - 1] == b1)
198  benar++;
199  if (red[x, y - 1] == b2)
200  benar++;
201  if (red[x + 1, y - 1] == b3)
202  benar++;
203  if (red[x - 1, y] == b4)
204  benar++;
205  if (red[x, y] == b5)
206  benar++;
207  if (red[x + 1, y] == b6)
208  benar++;
209  if (red[x - 1, y + 1] == b7)
210  benar++;
211  if (red[x, y + 1] == b8)
212  benar++;
213  if (red[x + 1, y + 1] == b9)
214  benar++;
215  return benar;
216  }
217
218  private Bitmap rectangle(Bitmap bmp, int x, int y, int
219  jarakx, int jaraky)
220  {
221  int batasy1 = y - jaraky;
222  int batasy2 = y + jaraky;
223  int batasx1 = x - jarakx;
224  int batasx2 = x + jarakx;
225
226  if (batasx1 < 0) batasx1 = 0;
227  if (batasx2 > bmp.Width) batasx2 = bmp.Width;
228  if (batasy1 < 0) batasy1 = 0;
229  if (batasy2 > bmp.Height) batasy2 = bmp.Height;
230
231  for (int i = batasy1; i < batasy2; i++)
232  for (int j = batasx1; j < batasx2; j++)
233                  {
234  if (i == batasy1 || i == batasy2 - 1 || j == batasx1 || j ==
235  batasx2 - 1)
236  bmp.SetPixel(j, i, Color.Red);
237  }
238  return bmp;
239  }
```

Penjelasan mengenai *Sourcecode* 4.4 proses ekstraksi fitur sebagai berikut :

Baris 3-13 adalah memasukkan nilai piksel ke dalam array

Baris 15-193 adalah pengecekan *bifurcation* yang ada pada citra dan memasukkan nama simbol ke dalam *database*.

Baris 195-218 adalah menginisialisasikan 8-ketetanggaan

Baris 220-241 adalah proses pemberian tanda merah pada citra yang terdapat *bifurcation.*

4.3.5 Implementasi Parameter HMM

Dalam proses parameter HMM meliputi matriks emisi, matriks transisi, matriks inisial, proses algoritma *forward*, algoritma *backward*, matriks *gamma*, matriks *epsilon*, evaluasi hasil, dan re-estimasi parameter HMM.

4.3.5.1 Implementasi Matriks Emisi

Matriks emisi digunakan untuk menjalankan algoritma *forward* dan *backward* baik proses inisialisasi maupun induksi. *Sourcecode* 4.5 adalah matriks emisi ditunjukkan sebagai berikut :

Sourcecode 4.5 matriks emisi

```
1   public HiddenMarkovModel(int symbols, int states)
2   : this(symbols, states, HiddenMarkovModelType.Ergodic)
3   {
4   }
5
6   public HiddenMarkovModel(int symbols, int states,
7   HiddenMarkovModelType type)
8   : this(null, null, states, type)
9   {
10  if (symbols <= 0)
11  {
12  throw new ArgumentOutOfRangeException("symbols", "Number of
13  symbols should be higher than zero.");
14  }
15
16  this.symbols = symbols;
17  this.B = new double[states, symbols];
18
19  for (int i = 0; i < states; i++)
20  for (int j = 0; j < symbols; j++)
21  B[i, j] = 1.0 / symbols;
22  }
```

Penjelasan *Sourcecode* 4.5 implementasi matriks emisi sebagai berikut :

Baris 1-5 adalah konstruktor dari fungsi.

Baris 7-25 adalah pengecekan simbol kurang dari 0 atau tidak dan menghitung nilai matriks emisi.

4.3.5.2 Implementasi Matriks Transisi Dan Inisial

Matriks transisi dan inisial juga digunakan untuk menjalankan algoritma *forward* dan *backward* baik proses inisialisasi maupun induksi. *Sourcecode* 4.6 adalah matriks transisi dan inisial ditunjukkan sebagai berikut :

Sourcecode 4.6 matriks transisi dan inisial

```
1   private HiddenMarkovModel(double[,] transitions, double[]
2   probabilities, int? states, HiddenMarkovModelType type)
3   {
4   #region Number of states N
5
6   int n = states.Value;
7   this.states = n;
8
9   #endregion
10
11  #region Transitions Matrix A
12
13  if (type == HiddenMarkovModelType.Ergodic)
14  {
15  transitions = new double[n, n];
16  for (int i = 0; i < n; i++)
17  for (int j = 0; j < n; j++)
18  transitions[i, j] = 1.0 / n;
19  }
20
21  this.A = transitions;
22
23  #endregion
24
25  #region Initial Probabilities pi
26
27  if (probabilities == null)
28  {
29  probabilities = new double[n];
30  probabilities[0] = 1.0;
31  }
32  this.pi = probabilities;
33
34  #endregion
35
36  }
```

Penjelasan *Sourcecode* 4.6 implementasi matriks transisi dan inisial sebagai berikut :

Baris 6-7 adalah menentukan nilai n adalah *state*.

Baris 13-21 adalah menghitung nilai matriks transisi.

Baris 27-32 adalah menentukan nilai matriks inisial.

4.3.5.3 Implementasi Algoritma *Forward*

Algoritma *forward* digunakan untuk membentuk matriks *forward* yang digunakan dalam menghitung matriks *epsilon* dan *gamma*. *Sourcecode* 4.7 adalah algoritma *forward* yang ditunjukkan sebagai berikut :

Sourcecode 4.7 Algoritma *forward*

```
1   private double[,] forward(int[] observations, out double[]
```

```
2     c)
3     {
4     int T = observations.Length;
5     double[] pi = Probabilities;
6     double[,] A = Transitions;
7
8     double[,] fwd = new double[T, States];
9     c = new double[T];
10
11    // 1. Initialization
12    for (int i = 0; i < States; i++)
13    c[0] += fwd[0, i] = pi[i] * B[i, observations[0]];
14
15    if (c[0] != 0) // Scaling
16    {
17    for (int i = 0; i < States; i++)
18    fwd[0, i] = fwd[0, i] / c[0];
19    }
20
21    // 2. Induction
22    for (int t = 1; t < T; t++)
23    {
24    for (int i = 0; i < States; i++)
25    {
26    double p = B[i, observations[t]];
27    double sum = 0.0;
28    for (int j = 0; j < States; j++)
29    sum += fwd[t - 1, j] * A[j, i];
30    fwd[t, i] = sum * p;
31
32    c[t] += fwd[t, i]; // scaling coefficient
33    }
34
35    if (c[t] != 0) // Scaling
36    {
37    for (int i = 0; i < States; i++)
38    fwd[t, i] = fwd[t, i] / c[t];
39    }
40    }
41
42    return fwd;
43    }
```

Penjelasan *Sourcecode* 4.7 implementasi algoritma *forward* sebagai berikut :

Baris 3-8 adalah menentukan batasan dan variable yang dipakai dalam algoritma *forward*.

Baris 12-20 adalah proses inisialisasi dan normalisasi hasil inisialisasi dari algoritma *forward*.

Baris 24-43 adalah proses induksi dan normalisasi hasil induksi dari algoritma *forward*.

4.3.5.4 Implementasi Algoritma *Backward*

Sama halnya dengan algoritma *forward*, algoritma *backward* juga digunakan untuk menghitung nilai dari matriks *gamma* dan *epsilon*. *Sourcecode* 4.8 untuk algoritma *backward* ditunjukkan sebagai berikut :

Sourcecode 4.8 Algoritma *backward*

```
1   private double[,] backward(int[] observations, double[] c)
2   {
3   int T = observations.Length;
4   double[] pi = Probabilities;
5   double[,] A = Transitions;
6
7   double[,] bwd = new double[T, States];
8
9   // 1. Initialization
10  for (int i = 0; i < States; i++)
11  bwd[T - 1, i] = 1.0 / c[T - 1];
12
13  // 2. Induction
14  for (int t = T - 2; t >= 0; t--)
15  {
16  for (int i = 0; i < States; i++)
17  {
18  double sum = 0;
19  for (int j = 0; j < States; j++)
20  sum += A[i, j] * B[j, observations[t + 1]] * bwd[t + 1, j];
21  bwd[t, i] += sum / c[t];
22  }
23  }
24
25  return bwd;
26  }
```

Penjelasan *Sourcecode* 4.8 implementasi algoritma *backward* sebagai berikut :

Baris 3-7 adalah variabel dan memberi batasan yang dipakai dalam algoritma *backward*.

Baris 9-11 adalah menghitung nilai inisialisasi dari algoritma *backward*.

Baris 14-24 adalah menghitung nilai induksi dari algoritma *backward*.

4.3.5.5 Implementasi Nilai Matriks *Gamma*

Nilai matriks *gamma* digunakan untuk estimasi kembali parameter HMM. *Sourcecode* 4.9 untuk menghitung nilai matriks *gamma* ditunjukkan sebagai berikut :

Sourcecode 4.9 Implementasi nilai matriks *gamma*

```
1   for (int t = 0; t < T; t++)
2   {
```

```
3     double s = 0;
4
5     for (int k = 0; k < States; k++)
6     s += gamma[i][t, k] = fwd[t, k] * bwd[t, k];
7
8     if (s != 0) // Scaling
9     {
10    for (int k = 0; k < States; k++)
11    gamma[i][t, k] /= s;
12    }
13    }
```

Penjelasan *Sourcecode* 4.9 implementasi nilai matriks *gamma* sebagai berikut :

Baris 6 adalah proses perhitungan nilai matriks *gamma*.

Baris 12 adalah proses perhitungan normalisasi nilai matriks *gamma*.

4.3.5.6 Implementasi Nilai Matriks *Epsilon*

Sama halnya dengan nilai matriks *gamma*, nilai matriks *epsilon* juga digunakan untuk estimasi kembali parameter HMM. *Sourcecode* 4.10 untuk menghitung nilai matriks *epsilon* sebagai berikut :

Sourcecode 4.10 Implementasi nilai matriks *epsilon*

```
1     // Calculate epsilon values for next computations
2     for (int t = 0; t < T - 1; t++)
3     {
4     double s = 0;
5
6     for (int k = 0; k < States; k++)
7     for (int l = 0; l < States; l++)
8     s += epsilon[i][t, k, l] = fwd[t, k] * A[k, l] * bwd[t + 1,
9     l] * B[l, sequence[t + 1]];
10
11    if (s != 0) // Scaling
12    {
13    for (int k = 0; k < States; k++)
14    for (int l = 0; l < States; l++)
15    epsilon[i][t, k, l] /= s;
16    }
17    }
```

Penjelasan *Sourcecode* 4.10 implementasi nilai matriks *epsilon* sebagai berikut :

Baris 8-9 adalah proses perhitungan nilai matriks *epsilon*.

Baris 16 adalah proses normalisasi nilai matriks *epsilon*.

4.3.5.7 Implementasi Evaluasi Hasil

Jika kondisi konvergen tercapai maka hasil *likelihood* akan dievaluasi. *Sourcecode* 4.11 untuk mengevaluasi hasil ditunjukkan sebagai berikut :

Sourcecode 4.11 Implementasi evaluasi hasil

```
1   public double Evaluate(int[] observations, bool logarithm)
2   {
3   if (observations == null)
4   throw new ArgumentNullException("observations");
5
6   if (observations.Length == 0)
7   return 0.0;
8
9
10  // Forward algorithm
11  double likelihood = 0;
12  double[] coefficients;
13
14  // Compute forward probabilities
15  forward(observations, out coefficients);
16
17  for (int i = 0; i < coefficients.Length; i++)
18  likelihood += Math.Log10(coefficients[i]);
19
20  // Return the sequence probability
21  return logarithm ? likelihood : Math.Exp(likelihood);
22  }
```

Penjelasan *Sourcecode* 4.11 implementasi evaluasi hasil sebagai berikut :

Baris 3-8 adalah proses pengecekan observasi dan panjang observasi.

Baris 12-16 adalah menginisialisasikan variabel dan mendapatkan nilai koefisien dari matriks *forward*.

Baris 18-23 adalah perhitungan nilai evaluasi dan pengecekan nilai *likelihood* yang diperoleh dari hasil evaluasi.

4.3.5.8 Implementasi Re-estimasi Nilai Matriks Inisial

Reestimasi matriks inisial bertujuan untuk menghitung kembali nilai matriks *forward* dan *backward* yang belum mencapai kondisi konvergen. *Sourcecode* 4.12 untuk re-estimasi matriks inisial ditunjukkan sebagai berikut :

Sourcecode 4.12 Implementasi re-estimasi nilai matriks inisial

```
1   // 3.1 Re-estimation of initial state probabilities
2   for (int k = 0; k < States; k++)
3   {
4   double sum = 0;
5   for (int i = 0; i < N; i++)
6   sum += gamma[i][0, k];
```

```
7       pi[k] = sum / N;
8       }
```

Penjelasan *Sourcecode* 4.12 implementasi re-estimasi nilai matriks inisial sebagai berikut :

Baris 1-8 adalah proses menentukan nilai matriks inisial.

4.3.5.9 Implementasi Re-estimasi Nilai Matriks Transisi

Sama halnya dengan matrik inisial, matriks transisi diestimasi kembali untuk menghitung nilai matriks *forward* dan *backward* agar mencapai kondisi konvergen. *Sourcecode* 4.13 untuk re-estimasi matriks transisi ditunjukkan sebagai berikut :

Sourcecode 4.13 Implementasi reestimasi nilai matriks transisi

```
1       // 3.2 Re-estimation of transition probabilities
2       for (int i = 0; i < States; i++)
3       {
4       for (int j = 0; j < States; j++)
5       {
6       double den = 0, num = 0;
7
8       for (int k = 0; k < N; k++)
9       {
10      int T = observations[k].Length;
11
12      for (int l = 0; l < T - 1; l++)
13      num += epsilon[k][l, i, j];
14
15      for (int l = 0; l < T - 1; l++)
16      den += gamma[k][l, i];
17      }
18
19      A[i, j] = (den != 0) ? num / den : 0.0;
20      }
21      }
```

Penjelasan *Sourcecode* 4.13 implementasi re-estimasi nilai matriks transisi sebagai berikut :

Baris 2-16 adalah proses perhitungan nilai num dan den.

Baris 19-20 adalah proses pengecekan nilai den dan proses menentukan nilai matriks transisi.

4.3.5.10Implementasi Re-estimasi Nilai Matriks Emisi

Sama halnya dengan matriks inisial dan transisi, matriks emisi diestimasi kembali untuk menghitung nilai matriks *forward* dan *backward* agar mencapai

kondisi konvergen. *Sourcecode* 4.14 untuk reestimasi nilai matriks emisi ditunjukkan sebagai berikut :

Sourcecode 4.14 Implementasi reestimasi nilai matriks emisi

```
1   for (int i = 0; i < States; i++)
2   {
3   for (int j = 0; j < Symbols; j++)
4   {
5   double den = 0, num = 0;
6
7   for (int k = 0; k < N; k++)
8   {
9   int T = observations[k].Length;
10
11  for (int l = 0; l < T; l++)
12  {
13  if (observations[k][l] == j)
14  num += gamma[k][l, i];
15  }
16
17  for (int l = 0; l < T; l++)
18  den += gamma[k][l, i];
19  }
20
21  B[i, j] = (num == 0) ? 1e-10 : num / den;
22  }
23  }
```

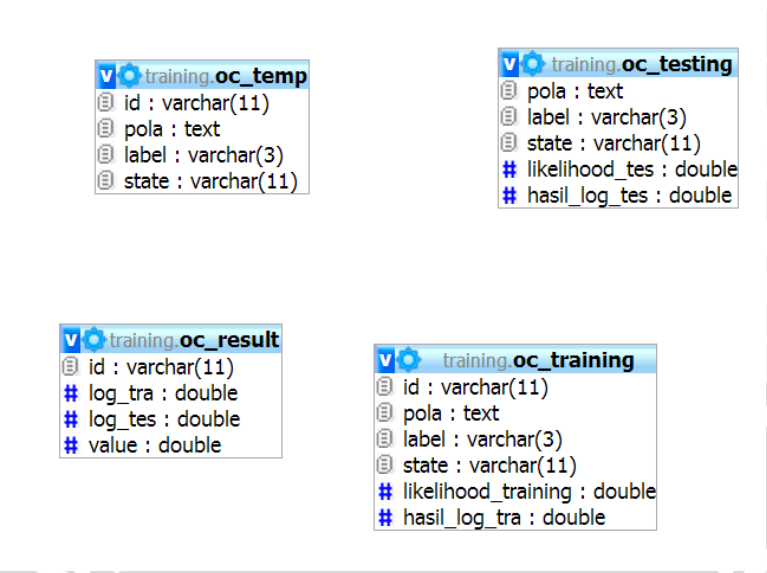
Penjelasan *Sourcecode* 4.14 implementasi re-estimasi nilai matriks emisi sebgai berikut :

Baris 1-20 adalah proses perhitungan nilai den dan num serta proses pengecekan nilai observasi pada kolom k dan baris l sesuai dengan nilai j.

Baris 22-23 adalah pengecekan nilai num dan proses menentukan nilai matriks emisi.

4.4 Implementasi Tabel Dalam Sistem Pengenalan Sidik Jari

Untuk implementasi database yang digunakan dalam sistem pengenalan sidik jari ditunjukkan pada Gambar 4.1 sebagai berikut :



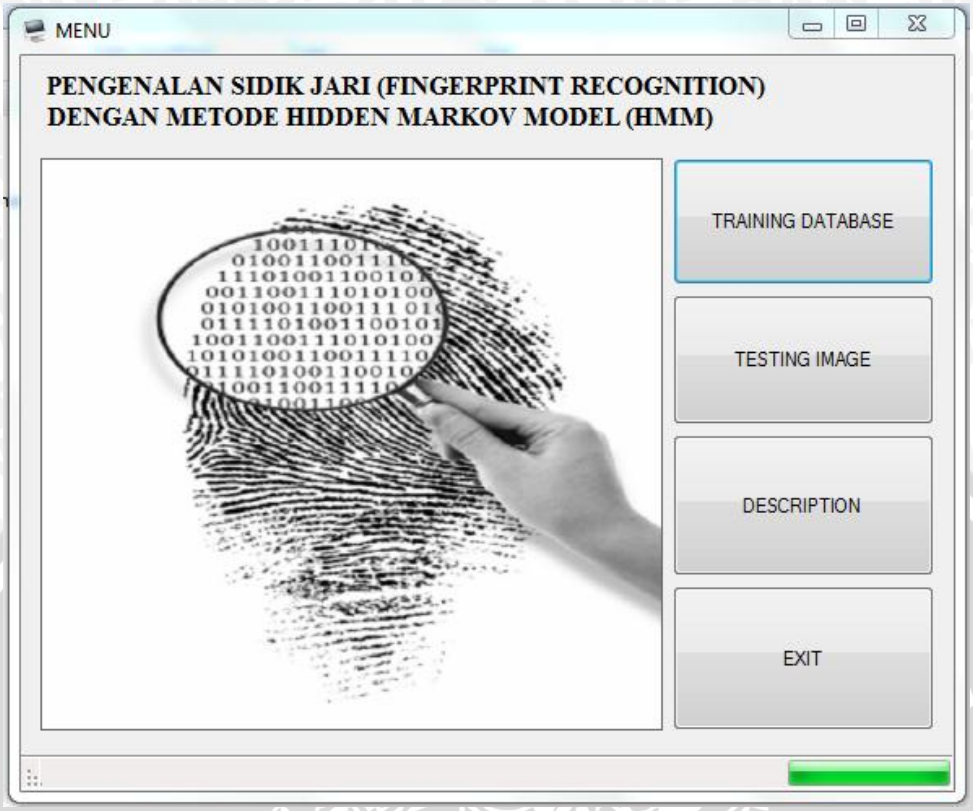
Gambar 4.1 Implementasi tabel dalam sistem

4.5 Implementasi Interface

Implementasi interface terdiri dari 4 bagian utama, yaitu :

a. Form utama

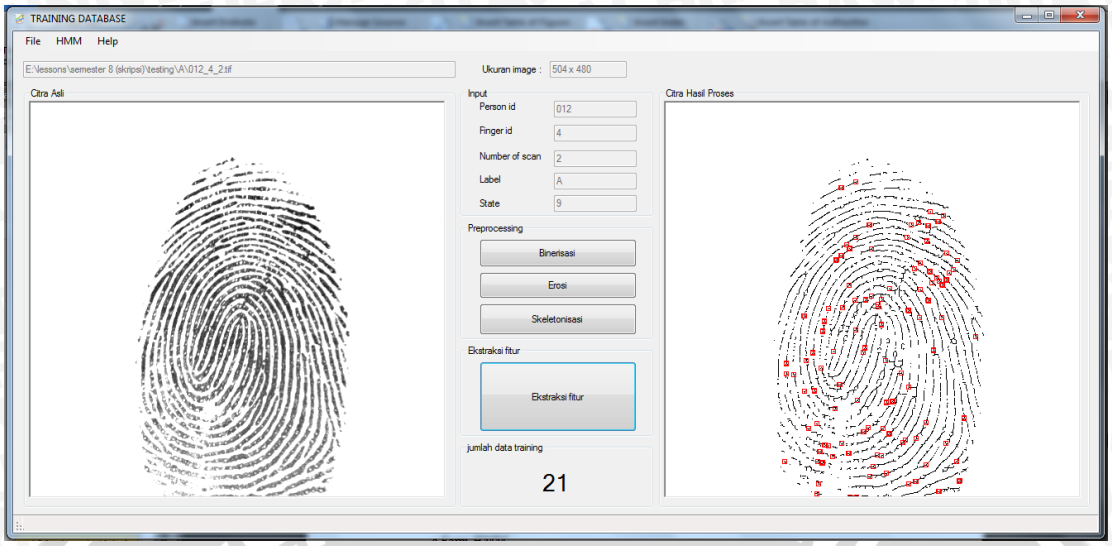
Form ini menampilkan seluruh *menu* yang ada pada pengenalan sidik jari. Adapun *menu* dalam form ini antara lain : *menu training database* yang digunakan untuk melatih citra, *menu testing image* digunakan untuk mengetahui hasil citra yang diuji, *menu description* berisi penjelasan singkat mengenai pengenalan sidik jari dengan metode *hidden markov model* (HMM), dan *menu exit* yang digunakan untuk keluar dari aplikasi. Adapun tampilannya ditunjukkan pada Gambar 4.2 berikut ini :



Gambar 4.2 Implementasi form utama

b. Form *Training*

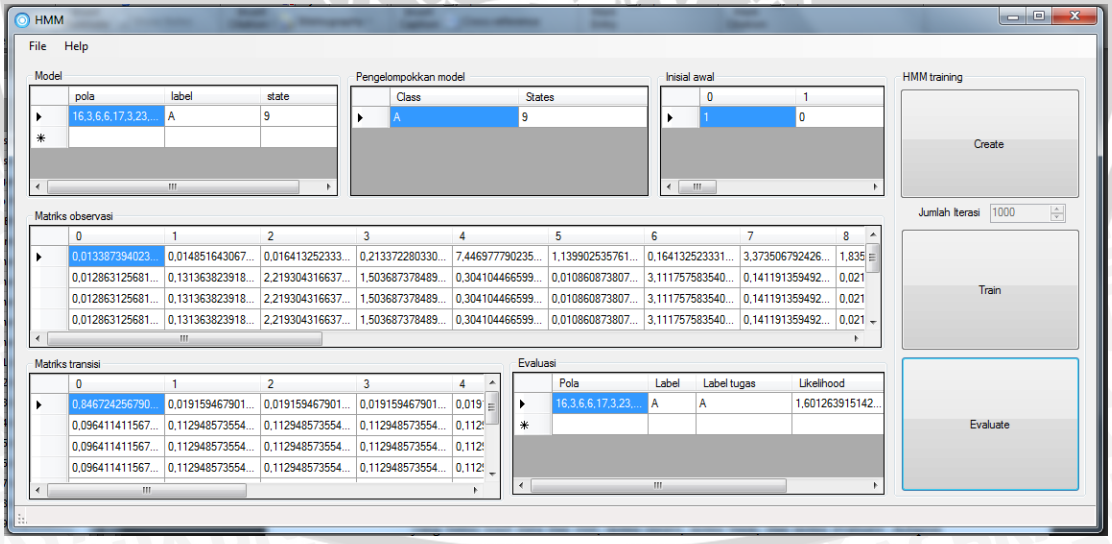
Form ini memasukkan hasil ekstraksi fitur yang ditemukan, label, *state.* dan id dari citra yang dilatih ke dalam *database*. Adapun *menu* yang terdapat dalam form *training* adalah *menu file* yang didalamnya terdapat *menu open* dan *exit*, *menu* HMM yang didalamnya berisi parameter dan hapus *database*. *Menu* binerisasi, *menu* erosi, *menu* skeletonisasi, *menu* ekstraksi fitur. Adapun tampilan form *training* ditunjukkan pada Gambar 4.3 berikut ini :



Gambar 4.3 Implementasi form *training*

c. Form HMM

Form ini memasukkan menghitung parameter HMM yang ada pada citra dan menghitung nilai *likelihood*. *Menu* yang terdapat dalam form ini adalah *menu file* yang berisi *load* data dan *exit*, *menu create*, *menu train*, dan *menu evaluate*. Adapun tampilan form HMM ditunjukkan pada Gambar 4.4 berikut ini :

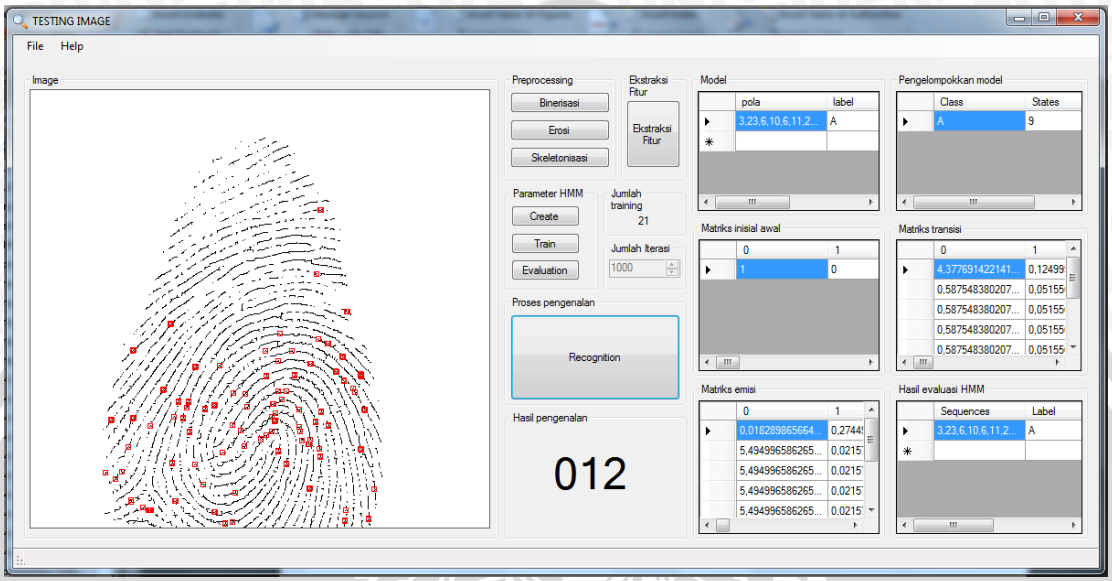


Gambar 4.4 Implementasi form HMM

d. Form *Testing*

Form ini memasukkan citra yang akan diuji dan hasil pengenalannya sama dengan id dikelasnya atau tidak. *Menu* yang terdapat pada form ini adalah *menu*

file yang berisi *open*, *load data*, dan *exit*, *menu* binerisasi, *menu* erosi, *menu* skeletonisasi, *menu* ekstraksi fitur, *menu* create, *menu* train, *menu* evaluate, dan *menu* recognition. Adapun tampilan form ditunjukkan pada Gambar 4.5 berikut ini :



Gambar 4.5 Implementasi form *testing*

BAB V
PENGUJIAN DAN ANALISIS

Dalam bab pengujian dan analisis akan dibahas mengenai pengujian terhadap sistem dan analisis yang didapatkan setelah pengujian. Pengujian yang akan dijelaskan pada subbab 5.1, untuk pengolahan hasil uji coba akan dijelaskan pada subbab 5.2 dan untuk analisis akan dijelaskan pada subbab 5.3.

5.1 Pengujian

Dalam pengujian sistem pengenalan sidik jari akan diambil citra sebanyak 8 citra dari 5 kelas. Dari 40 citra citra sidik jari tersebut kemudian akan dicoba pada program pengenalan sidik jari dengan metode *Hidden Markov Model* yang sudah dibuat untuk kemudian dibandingkan nilai *log likelihood training* dan *testing* untuk dikenali.

Pengujian dilakukan dengan membuat variasi jumlah data *training* dari sidik jari dan pengujian dilakukan dengan data / orang yang terdapat dalam data *training*. Untuk 20 data *training*, maka citra sidik jari dari kelas A, B, C, D dan E masing-masing diambil citranya sebanyak 4 citra. Untuk 40 data *training*, maka model citra sidik jari kelas A, B, C, D dan E masing-masing diambil citranya sebanyak 8 citra dan seterusnya. Pengujian yang dilakukan pada sistem ditunjukkan adalah pengujian terhadap 20 data *training*, pengujian terhadap 40 data *training*, pengujian terhadap 60 data *training* pengujian terhadap 80 data *training* dan pengujian terhadap 100 data *training*.

5.1.1 Pengujian terhadap 20 data *training*

Pada Tabel 5.1 adalah hasil pengujian terhadap 20 data *training* dengan masing-masing label diambil 4 data untuk dijadikan data *training*. Adapun hasil pengujiannya sebagai berikut :

Tabel 5.1 Tabel pengujian terhadap 20 data *training*

Pengujian terhadap 20 data *training*				
no	sidik jari	label	hasil pengenalan	dikenali/tidak
1	012_3_2	A	027	Tidak

2	012_3_4	A	012	Dikenali
3	012_3_5	A	012	Dikenali
4	012_4_2	A	012	Dikenali
5	012_5_2	A	012	Dikenali
6	012_5_4	A	012	Dikenali
7	012_6_8	A	076	Tidak
8	012_8_1	A	012	Dikenali
9	013_3_2	B	013	Dikenali
10	013_3_8	B	013	Dikenali
11	013_4_3	B	013	Dikenali
12	013_5_4	B	013	Dikenali
13	013_6_1	B	013	Dikenali
14	013_6_4	B	013	Dikenali
15	013_7_8	B	013	Dikenali
16	013_8_1	B	012	Tidak
17	022_3_6	C	022	Dikenali
18	022_5_3	C	022	Dikenali
19	022_5_6	C	022	Dikenali
20	022_6_8	C	022	Dikenali
21	022_7_1	C	022	Dikenali
22	022_7_2	C	022	Dikenali
23	022_7_3	C	076	Tidak
24	022_7_5	C	027	Tidak
25	027_4_2	D	027	Dikenali
26	027_4_8	D	013	Tidak
27	027_5_8	D	013	Tidak
28	027_6_3	D	027	Dikenali
29	027_7_2	D	027	Dikenali
30	027_7_5	D	027	Dikenali
31	027_8_2	D	027	Dikenali
32	027_8_5	D	027	Dikenali
33	076_3_4	E	076	Dikenali
34	076_3_8	E	012	Tidak
35	076_4_6	E	076	Dikenali
36	076_5_8	E	076	Dikenali
37	076_6_3	E	022	Tidak
38	076_7_1	E	027	Tidak
39	076_7_2	E	076	Dikenali
40	076_8_8	E	076	Dikenali

5.1.2 Pengujian terhadap 40 data training

Pada Tabel 5.2 adalah hasil pengujian terhadap 40 data *training* dengan masing-masing label diambil 8 data untuk dijadikan data *training*. Adapun hasil pengujiannya sebagai berikut :

Tabel 5.2 Tabel pengujian terhadap 40 data *training*

Pengujian terhadap 40 data *training*				
no	sidik jari	label	hasil pengenalan	dikenali/tidak
1	012_3_2	A	012	Dikenali
2	012_3_4	A	012	Dikenali
3	012_3_5	A	012	Dikenali
4	012_4_2	A	012	Dikenali
5	012_5_2	A	012	Dikenali
6	012_5_4	A	012	Dikenali
7	012_6_8	A	076	Tidak
8	012_8_1	A	013	Dikenali
9	013_3_2	B	013	Dikenali
10	013_3_8	B	013	Dikenali
11	013_4_3	B	013	Dikenali
12	013_5_4	B	013	Dikenali
13	013_6_1	B	013	Dikenali
14	013_6_4	B	013	Dikenali
15	013_7_8	B	013	Dikenali
16	013_8_1	B	013	Dikenali
17	022_3_6	C	022	Dikenali
18	022_5_3	C	022	Dikenali
19	022_5_6	C	022	Dikenali
20	022_6_8	C	022	Dikenali
21	022_7_1	C	022	Dikenali
22	022_7_2	C	022	Dikenali
23	022_7_3	C	076	Tidak
24	022_7_5	C	027	Tidak
25	027_4_2	D	027	Dikenali
26	027_4_8	D	027	Dikenali
27	027_5_8	D	027	Dikenali
28	027_6_3	D	027	Dikenali
29	027_7_2	D	027	Dikenali
30	027_7_5	D	027	Dikenali
31	027_8_2	D	027	Dikenali
32	027_8_5	D	027	Dikenali
33	076_3_4	E	076	Dikenali

34	076_3_8	E	076	Dikenali
35	076_4_6	E	076	Dikenali
36	076_5_8	E	022	Tidak
37	076_6_3	E	022	Tidak
38	076_7_1	E	012	Tidak
39	076_7_2	E	076	Dikenali
40	076_8_8	E	076	Dikenali

5.1.3 Pengujian terhadap 60 data *training*

Pada Tabel 5.3 adalah hasil pengujian terhadap 60 data *training* dengan masing – masing label diambil 12 data untuk dijadikan data *training*. Adapun hasil pengujiannya sebagai berikut :

Tabel 5.3 Tabel pengujian terhadap 60 data *training*

Pengujian terhadap 60 data *training*				
no	sidik jari	label	hasil pengenalan	dikenali/tidak
1	012_3_2	A	012	Dikenali
2	012_3_4	A	012	Dikenali
3	012_3_5	A	012	Dikenali
4	012_4_2	A	012	Dikenali
5	012_5_2	A	012	Dikenali
6	012_5_4	A	012	Dikenali
7	012_6_8	A	076	Tidak
8	012_8_1	A	012	Dikenali
9	013_3_2	B	013	Dikenali
10	013_3_8	B	013	Dikenali
11	013_4_3	B	013	Dikenali
12	013_5_4	B	013	Dikenali
13	013_6_1	B	013	Dikenali
14	013_6_4	B	013	Dikenali
15	013_7_8	B	013	Dikenali
16	013_8_1	B	013	Dikenali
17	022_3_6	C	022	Dikenali
18	022_5_3	C	022	Dikenali
19	022_5_6	C	022	Dikenali
20	022_6_8	C	022	Dikenali
21	022_7_1	C	022	Dikenali
22	022_7_2	C	022	Dikenali
23	022_7_3	C	013	Tidak
24	022_7_5	C	012	Tidak
25	027_4_2	D	027	Dikenali

26	027_4_8	D	027	Dikenali
27	027_5_8	D	027	Dikenali
28	027_6_3	D	027	Dikenali
29	027_7_2	D	027	Dikenali
30	027_7_5	D	027	Dikenali
31	027_8_2	D	027	Dikenali
32	027_8_5	D	027	Dikenali
33	076_3_4	E	076	Dikenali
34	076_3_8	E	076	Dikenali
35	076_4_6	E	076	Dikenali
36	076_5_8	E	022	Tidak
37	076_6_3	E	076	Dikenali
38	076_7_1	E	013	Tidak
39	076_7_2	E	076	Dikenali
40	076_8_8	E	076	Dikenali

5.1.4 Pengujian terhadap 80 data *training*

Pada Tabel 5.4 adalah hasil pengujian terhadap 80 data *training* berarti masing-masing label diambil 16 data untuk dijadikan data *training*. Berikut ini hasil pengujiannya :

Tabel 5.4 Tabel pengujian terhadap 80 data *training*

Pengujian terhadap 80 data *training*				
no	sidik jari	label	hasil pengenalan	dikenali/tidak
1	012_3_2	A	012	Dikenali
2	012_3_4	A	012	Dikenali
3	012_3_5	A	012	Dikenali
4	012_4_2	A	012	Dikenali
5	012_5_2	A	012	Dikenali
6	012_5_4	A	012	Dikenali
7	012_6_8	A	012	Dikenali
8	012_8_1	A	012	Dikenali
9	013_3_2	B	013	Dikenali
10	013_3_8	B	013	Dikenali
11	013_4_3	B	013	Dikenali
12	013_5_4	B	013	Dikenali
13	013_6_1	B	013	Dikenali
14	013_6_4	B	013	Dikenali
15	013_7_8	B	013	Dikenali
16	013_8_1	B	013	Dikenali
17	022_3_6	C	022	Dikenali

18	022_5_3	C	022	Dikenali
19	022_5_6	C	022	Dikenali
20	022_6_8	C	022	Dikenali
21	022_7_1	C	022	Dikenali
22	022_7_2	C	022	Dikenali
23	022_7_3	C	013	Tidak
24	022_7_5	C	076	Tidak
25	027_4_2	D	027	Dikenali
26	027_4_8	D	027	Dikenali
27	027_5_8	D	027	Dikenali
28	027_6_3	D	027	Dikenali
29	027_7_2	D	027	Dikenali
30	027_7_5	D	027	Dikenali
31	027_8_2	D	027	Dikenali
32	027_8_5	D	027	Dikenali
33	076_3_4	E	076	Dikenali
34	076_3_8	E	076	Dikenali
35	076_4_6	E	076	Dikenali
36	076_5_8	E	022	Tidak
37	076_6_3	E	076	Dikenali
38	076_7_1	E	013	Tidak
39	076_7_2	E	076	Dikenali
40	076_8_8	E	076	Dikenali

5.1.5 Pengujian terhadap 100 data *training*

Pada Tabel 5.5 adalah hasil uji coba terhadap 100 data *training* berarti masing-masing label diambil 20 data untuk dijadikan data *training*. Adapun hasil pengujiannya sebagai berikut :

Tabel 5.5 Tabel pengujian terhadap 100 data *training*

Pengujian terhadap 100 data *training*				
no	sidik jari	label	hasil pengenalan	dikenali/tidak
1	012_3_2	A	012	Dikenali
2	012_3_4	A	012	Dikenali
3	012_3_5	A	012	Dikenali
4	012_4_2	A	012	Dikenali
5	012_5_2	A	012	Dikenali
6	012_5_4	A	012	Dikenali
7	012_6_8	A	012	Dikenali
8	012_8_1	A	012	Dikenali
9	013_3_2	B	013	Dikenali

10	013_3_8	B	013	Dikenali
11	013_4_3	B	013	Dikenali
12	013_5_4	B	013	Dikenali
13	013_6_1	B	013	Dikenali
14	013_6_4	B	013	Dikenali
15	013_7_8	B	013	Dikenali
16	013_8_1	B	013	Dikenali
17	022_3_6	C	022	Dikenali
18	022_5_3	C	022	Dikenali
19	022_5_6	C	022	Dikenali
20	022_6_8	C	022	Dikenali
21	022_7_1	C	022	Dikenali
22	022_7_2	C	022	Dikenali
23	022_7_3	C	013	Tidak
24	022_7_5	C	022	Dikenali
25	027_4_2	D	027	Dikenali
26	027_4_8	D	027	Dikenali
27	027_5_8	D	027	Dikenali
28	027_6_3	D	027	Dikenali
29	027_7_2	D	027	Dikenali
30	027_7_5	D	027	Dikenali
31	027_8_2	D	027	Dikenali
32	027_8_5	D	027	Dikenali
33	076_3_4	E	076	Dikenali
34	076_3_8	E	076	Dikenali
35	076_4_6	E	076	Dikenali
36	076_5_8	E	012	Tidak
37	076_6_3	E	076	Dikenali
38	076_7_1	E	013	Tidak
39	076_7_2	E	076	Dikenali
40	076_8_8	E	076	Dikenali

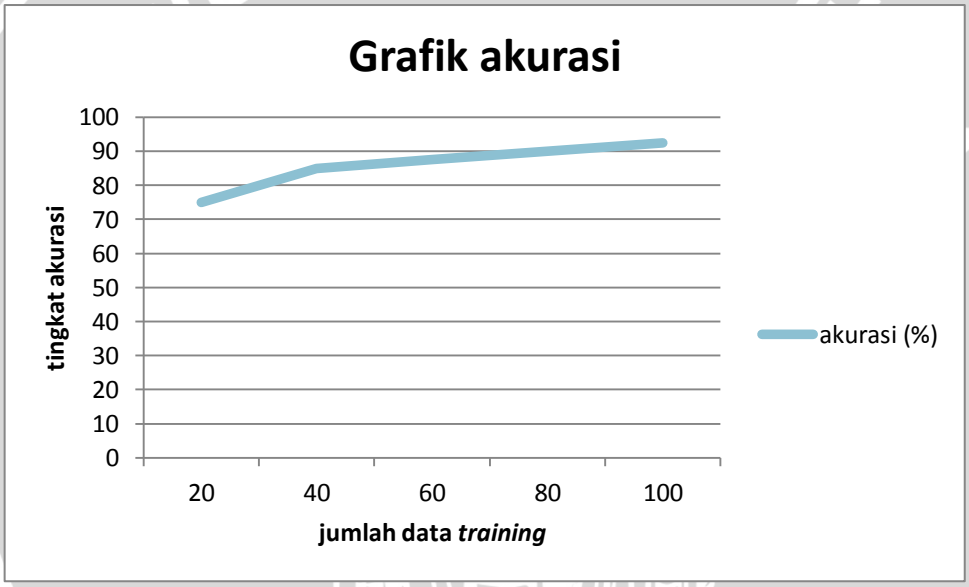
5.2 Pengolahan Hasil Uji Coba

Dari hasil yang didapatkan dari pengujian dapat dihitung persen akurasi keseluruhan sistem. Persen akurasi didapat dengan cara membagi jumlah sidik jari yang dikenali dengan jumlah seluruh citra sidik jari yang digunakan sebagai *testing*. Tabel 5.6 menunjukkan akurasi yang diperoleh dari masing-masing pengujian.

Tabel 5.6 Tabel akurasi hasil uji coba

Jumlah *training*	Akurasi (%)
20	75
40	85
60	87,5
80	90
100	92,5

Hubungan jumlah data *training* dengan tingkat akurasi ditunjukkan pada Gambar 5.1 berikut ini :



Gambar 5.1 Grafik akurasi terhadap jumlah data *training*

5.3 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian yang telah dilakukan. Analisis mengacu pada pengujian yang telah dilakukan dan berikut ini hasil analisis yang diperoleh dari pengujian yang dilakukan.

5.3.1 Pengaruh Jumlah Training Terhadap Tingkat Akurasi Proses Pengenalan Sidik Jari

Berdasarkan Tabel 5.6 peningkatan jumlah training mengakibatkan peningkatan besarnya persen akurasi pada keseluruhan sistem. Pada proses database dimasukkan citra atau gambar dari masing-masing sidik jari sebagai pembanding. Semakin banyak citra sidik jari yang dimasukkan ke dalam database, maka proses pengenalan akan lebih akurat karena dengan semakin banyaknya database untuk masing-masing sidik jari, masalah kemiripan antar sidik jari akan semakin berkurang.

Tabel 5.7 Tabel peningkatan akurasi terhadap jumlah data *training*

Jumlah data *training*	Peningkatan akurasi
20 data *training* dan 40 data *training*	1,133 kali lipat
40 data *training* dan 60 data *training*	1,029 kali lipat
60 data *training* dan 80 data *training*	1,029 kali lipat
80 data *training* dan 100 data *training*	1,028 kali lipat

5.3.2 Pengaruh Deretan / *Sequence* dan *State* Terhadap Proses Pengenalan Sidik Jari

Deretan yang dihasilkan dari proses ekstraksi fitur mempengaruhi nilai *likelihood* yang dihasilkan karena semakin banyak deretan yang dihasilkan semakin banyak *exponen* yang dihasilkan. Nilai *likelihood* bernilai sama untuk semua jumlah data *training*. Untuk 012_4_2 yang memiliki deretan sebanyak 153 buah yang menghasilkan *likelihood* 1.60126391514205e-72 dan untuk contoh dari citra 013_8_2 yang memiliki deretan sebanyak 161 buah yang menghasilkan *likelihood* 7.09686204869005e-76 dari sini dapat dianalisis bahwa semakin banyak suatu citra memiliki deretan semakin banyak *exponen* yang dihasilkan. Sedangkan untuk *state* berpengaruh terhadap iterasi yang dibutuhkan untuk mencapai konvergen untuk 1 *state* dibutuhkan hanya 5 kali iterasi dan untuk 2 *state* dibutuhkan lebih dari 100 kali iterasi dengan begitu semakin banyak *state* maka waktu yang dibutuhkan untuk mencapai konvergen juga akan semakin lama.

5.3.3 Pengaruh Parameter HMM Terhadap Nilai *Likelihood*

Metode HMM terdiri dari tiga parameter yang berbentuk matriks yang harus dihitung nilainya sampai dengan konvergen. Tiga parameternya adalah matriks inisial, matriks transisi, dan matriks emisi atau observasi. Dalam program yang telah diimplementasikan digunakan 1000 iterasi untuk menghitung parameter HMM agar mencapai kondisi konvergen. Parameter-parameter HMM memberikan nilai likelihood yang sudah di iterasi sebanyak 1000 kali.

BAB VI
PENUTUP

6.1 Kesimpulan

Dari hasil penelitian didapatkan kesimpulan sebagai berikut :

1. HMM mampu digunakan untuk pengenalan sidik jari dan bekerja pada *sequence* dari *bifurcation*.
2. HMM dirancang untuk menghitung nilai dari masing-masing *bifurcation* yang ditemukan dalam citra sidik jari dan untuk menghitung nilai *likelihood* dari citra sidik jari.
3. Implementasi HMM dalam pengenalan sidik jari menggunakan 3 algoritma yaitu algoritma forward, algoritma backward, dan algoritma Baum – Welch.
4. Pengenalan sidik jari dengan metode HMM memperoleh rata-rata akurasi hingga 86% dan penambahan jumlah data *training* meningkatkan akurasi sebesar 1,028 – 1,133 kali lipat.
5. Faktor yang paling penting untuk meningkatkan akurasi adalah dengan menambah jumlah data *training*. Selain itu, Metode HMM sangat berpengaruh pada banyaknya *sequence* dan *state* yang digunakan.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan lebih lanjut antara lain :

1. Untuk mencapai konvergen tidak hanya menggunakan iterasi tetapi bisa juga dengan menggunakan jarak antara *likelihood* yang lama dengan *likelihood* yang baru atau sampai konvergen.
2. Ekstraksi fitur yang digunakan tidak hanya percabangan tetapi bisa juga menggunakan *core point* dan *minutiae*.
3. Jumlah *bifurcation* tidak hanya 24 buah tetapi bisa ditambahkan lagi.
4. Banyak metode lain untuk dijadikan pengenalan sidik jari.
5. Obyek sidik jari sangat banyak dan satu dengan yang lainnya berbeda baik itu jenis, model dan lain-lain.

DAFTAR PUSTAKA

[ARD-11] Ardhianto, Eka, Siti Munawaroh, and Agung Prihandono. (2011),"Pengolah Citra Digital untuk Identifikasi Ciri Sidik Jari Berbasis Minutiae." Dinamik-Jurnal Teknologi Informasi 16, no. 1.

[BIC-03] Bicego, M., U. Castellani, and V. Murino, (2003) "Using Hidden Markov Models and wavelets for face recognition." In Image Analysis and Processing, 2003. Proceedings. 12th International Conference on, pp. 52-56. IEEE.

[ELI-08] Elizabeth, (2008), "Pengembangan Sistem Identifikasi Biometrik Wajah Menggunakan Metode *Neural Network* dan *Pattern Matching*", Jurusan Teknik Elektro Universitas Indonesia, Depok.

[GON-02] Gonzales, R.C., Woods, R. E., dan Eddins, S. L., (2002)."Digital Image Processing Second Edition", dalam *Image Sampling and Quantization*, Prentice Hall, hal. 55.

[HUA-12] Huabiao Qin; Jun Liu; Tianyi Hong, (2012), "An eye state identification method based on the Embedded Hidden Markov Model," Vehicular Electronics and Safety (ICVES), IEEE International Conference on , vol., no., pp.255,260, 24-27 July

[JAI-07] Jain, Anil K., Patrick Flynn, Arun A. Ross, (2007),"Handbook of Biometrics", New York : Springer.

[JAM-08] James, (2008), "Identifikasi Plat Nomor Mobil Dengan Skeletonisasi Menggunakan Jaringan Saraf Tiruan", Jurusan Teknik Elektro Universitas Indonesia, Depok.

[KAU-08] Kaur, Manvjeet, Mukhwinder Singh, Akshay Girdhar, and Parvinder S. Sandhu, (2008),"Fingerprint verification system using minutiae extraction technique", World Academy of Science, Engineering and Technology 46, 497-502.

[KRI-11] Krishnasamy Prasanna, Serge Belongie, David Kriegman, (2011),"Wet Fingerprint Recognition : Challenges and

Opportunities", IEEE, International Conference on In Biometrics International Joint Conference, 1-7.

[LEO-08] Leon, J., Sanchez, G., Aguilar, G., Toscano, K., Perez,H., Nakano, M., (2008)," Fingerprint Recongnition Using Espatial Minutae Information", The International Conference on Electronics, Robotics and Automotive Mechanics, 381-386.

[LEK-11] Leksono, Bowo, Achmad Hidayatno, dan R. Rizal Isnanto. (2011), "Aplikasi Metode Template Matching untuk Klasifikasi Sidik Jari." TRANSMISI 13, no. 1: 1-6.

[LEO-09] Leon,J., Sanchez,G ., Aguilar,G ., Toscano,L., Perez,H., Ramirez,J.M., (2009),"Fingerprint verification applying invariant moments", The IEEE International Midwest Symposium on Circuits and Systems, 751-757.

[MAL-03] Maltoni, dkk, (2003),"Handbook of Fingerprint Recognition", New York:Springer.

[MUL-08] Mulyani, dkk, (2008), "Penerapan Hidden Markov Model Dalam Clustering Sequence Protein Globin", Program Studi Ilmu Komputer, FMIPA, Universitas Gadjah mada, Yogyakarta.

[NEU-12] http://www.neurotechnology.com/download.html diakses pada Oktober 2012.

[PAT-05] Patil, P.M.; Suralkar, S.R.; Sheikh, F.B.,(2005),"Rotation invariant thinning algorithm to detect ridge bifurcations for fingerprint identification," Tools with Artificial Intelligence, ICTAI 05. 17th IEEE International Conference on , vol., no., pp.8 pp.,641, 16-16 Nov.

[POR-10] Pornpanomchai Chomtip, Apiradee Phaisitkulwiwat, (2010),"Fingerprint Recognition By Euclidian Distance", IEEE, Second International Conference on Computer and Network Technology, 437-441.

[PRA-11] Prabhakar, Salil, Alexander Ivanisov, and A. K. Jain, (2011),"Biometric recognition: Sensor characteristics and image

quality." Instrumentation & Measurement Magazine, IEEE 14, no. 3, 10-16.

[SEP-12] Sepritahara, (2012), "Sistem Pengenalan Wajah (Face Recognition) Menggunakan Hidden Markov Model (HMM)", Jurusan Teknik Elektro Universitas Indonesia, Depok.

[SER-82] Serra, J, (1982), "Image Analysis and Mathematical Morphology", Academic Press, Inc., London.

[SRI-10] Sri Huning Anwariningsih, Agus Zainal Arifin, Anny Yuniarti, (2010), "Metode Shape Descriptor Berbasis Shape Matrix Untuk Estimasi Bentuk Structuring Element", Tesis, Program Magister Teknik Informatika, Jurusan Teknik Informatika, FakultasTeknologiInformasi, Institut Teknologi Sepuluh Nopember, Surabaya.

[SUN-10] Sun Zhipeng, JinGuang Sun, (2010),"Face Recognition Based on Fractal and Hidden Markov Model", IEEE, Third International Symposium on Electronic Commerce and Security, 293-297.

[WEI-12] Wei Chen; Lichun Sui; Zhengchao Xu; Yu Lang, (2012),"Improved Zhang-Suen thinning algorithm in binary line drawing applications," Systems and Informatics (ICSAI), International Conference on , vol., no., pp.1947,1950, 19-20 May

[WIB-12] Wibowo, A., & Fathurrahman, F, (2012), "Implementasi Algoritma Breadth First Search Dan Obstacle Detection Dalam Penelusuran Labirin Dinamis Menggunakan Robot Lego". *Jurnal Ilmu Komputer dan Informasi*, *4*(1), 15.

[YUL-08] Yulianti S., Maria, (2008),"Pengenalan Retina Dengan Hidden Markov Model", Jurusan Teknik Elektro Universitas Indonesia, Depok.

[ZHA-12] Zhao Weizhou, Hui Zhang, (2012), "Secure Fingerprint Recognition Based on Frobenius Norm", IEEE, International Conference on Computer Science and Electronics Engineering, Vol. 1, 388-391.

LAMPIRAN

Lampiran 1. Hasil perhitungan Binerisasi Metode otsu

nilai piksel citra	jumlah piksel	jumlah piksel total	Pi	i.Pi	Mean	Zerothcm	Firstcm	variance akhir	cek
53	1		0,006493506	0,344155844		0,006493506	0,344155844	8,982292917	8
55	1		0,006493506	0,357142857		0,006493506	0,357142857	8,039249033	8
59	1		0,006493506	0,383116883		0,006493506	0,383116883	6,31002401	8
62	2		0,012987013	0,805194805		0,012987013	0,805194805	10,36848818	10
63	2		0,012987013	0,818181818		0,012987013	0,818181818	9,642924275	10
69	1		0,006493506	0,448051948		0,006493506	0,448051948	2,901994131	10
70	1		0,006493506	0,454545455		0,006493506	0,454545455	2,633086568	10
71	2		0,012987013	0,922077922		0,012987013	0,922077922	4,785781418	10
72	2		0,012987013	0,935064935		0,012987013	0,935064935	4,297059613	10
73	1		0,006493506	0,474025974		0,006493506	0,474025974	1,904795251	10
74	1	154	0,006493506	0,480519481	90,07142857	0,006493506	0,480519481	1,68817527	10
75	2		0,012987013	0,974025974		0,012987013	0,974025974	2,988788937	10
76	1		0,006493506	0,493506494		0,006493506	0,493506494	1,294150994	10
77	1		0,006493506	0,5		0,006493506	0,5	1,116746699	10
.	.		.	.		.	.	.	.
.	.		.	.		.	.	.	.
.	.		.	.		.	.	.	.
248	1		0,006493506	1,61038961		0,006493506	1,61038961	163,0159064	163
249	1		0,006493506	1,616883117		0,006493506	1,616883117	165,0868681	165
255	39		0,253246753	64,57792208		0,253246753	64,57792208	9224,834028	165

Lampiran 2. Nilai parameter HMM awal

Tabel 1. Matriks inisial

	0
0	1

Tabel 2. Matriks emisi / observasi

	0	1	2	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	23
0	0,041667	0,041667	0,041667	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	0,041667

Tabel 3. Matriks transisi

	0
0	1

Lampiran 3. Hasil algoritma *forward*

Tabel 1. Hasil Inisialisasi algoritma *forward*

I	i < states	fwd(0,i)		c[0]!=0	i < states	c[0]	fwd[0,i]
0	TRUE	0,041667		TRUE	TRUE	0,041667	1
1	FALSE	-		-	-		-

Tabel 2. Hasil induksi algoritma *forward*

t	i	J	t < T	i < states	j < states	t-1	J	i	fwd[t-1,j]	A[i,j]	fwd[t-1,j] * A[i,j]	sum	fwd[t,i]	c[t]	c[t] != 0	i < states	fwd[t,i]
1	0	0	TRUE	TRUE	TRUE	0	0	0	1	1	1	1	0,041667	0,041667	TRUE	TRUE	1
2	1	1	TRUE	FALSE	FALSE	1	0	0	1	1	1	1	0,041667	0,041667	TRUE	TRUE	1
3	2	2	TRUE	FALSE	FALSE	2	0	0	1	1	1	1	0,041667	0,04166667	TRUE	TRUE	1
4	3	3	TRUE	FALSE	FALSE	3	0	0	1	1	1	1	0,041667	0,04166667	TRUE	TRUE	1
5	4	4	TRUE	FALSE	FALSE	4	0	0	1	1	1	1	0,041667	0,04166667	TRUE	TRUE	1
6	5	5	FALSE	FALSE	FALSE												

Lampiran 4. Hasil algoritma *backward*

Tabel 1. Hasil inisialisasi algoritma *backward*

I	i < states	bwd[T-1,i]
0	TRUE	24
1	FALSE	

Tabel 2. Hasil induksi algoritma *backward*

t	I	J	t >= 0	i< states	j < states	T	i	j	A[i,j] * B[j,observations[t+1]] * bwd[t+1,j]	sum	sum / c[t]	bwd[t,i]
4	0	0	TRUE	TRUE	TRUE	4	0	0	1	1	24	24
3	1	1	TRUE	FALSE	FALSE	3	0	0	1	1	24	24
2			TRUE			2	0	0	1	1	24	24
1			TRUE			1	0	0	1	1	24	24
0			TRUE			0	0	0	1	1	24	24

Lampiran 5. Hasil perhitungan matriks *gamma*

t	k	t < T	k < states	T	K	gamma[i][t,k]	s	gamma[i][t,k]
0	0	TRUE	TRUE	0	0	24	24	1
1	1	TRUE	FALSE	1	0	24	24	1
2	2	TRUE	FALSE	2	0	24	24	1
3	3	TRUE	FALSE	3	0	24	24	1
4	4	TRUE	FALSE	4	0	24	24	1
5	5	TRUE	FALSE	5	0	24	24	1
6	6	FALSE	FALSE					

Lampiran 6. Hasil perhitungan matriks *epsilon*

t	k	l	t < T-1	k < states	i < states	t	k	l	epsilon[i][t,k,l]	s+	epsilon[i][t,k,l]
0	0	0	TRUE	TRUE	TRUE	0	0	0	1	1	1
1	1	1	TRUE	FALSE	FALSE	1	0	0	1	1	1
2	2	2	TRUE	FALSE	FALSE	2	0	0	1	1	1
3	3	3	TRUE	FALSE	FALSE	3	0	0	1	1	1
4	4	4	TRUE	FALSE	FALSE	4	0	0	1	1	1
5	5	5	FALSE	FALSE	FALSE						
6	6	6	FALSE	FALSE	FALSE						

Lampiran 7. Hasil perhitungan *likelihood*

t	t<scaling.length	log10	newlikelihood +
0	TRUE	-1,38021	-8,28127
1	TRUE	-1,38021	
2	TRUE	-1,38021	
3	TRUE	-1,38021	
4	TRUE	-1,38021	
5	TRUE	-1,38021	
6	FALSE		

Lampiran 8. Re-estimasi parameter HMM dan perhitungan algoritma *forward, backward,* matriks *gamma, epsilon, likelihood* baru dan evaluasi hasil setelah parameter HMM terestimasi kembali

Tabel 1. Re-estimasi matriks inisial

k	i	k < states	i < N	K	i	gamma[i][0,k]	sum +=	pi[k]
0	0	TRUE	TRUE	0	0	1	1	1
1	1	FALSE	FALSE					

Tabel 2. Matriks inisial yang terestimasi kembali

pi	0
0	1

Tabel 3. Re-estimasi matriks transisi

i	j	K	l	i < states	j < states	k < N	l < T-1	k	l	i	j	epsilon[k][l,i,j]	num	gamma[k][l,i]	den	A[i,j]
0	0	0	0	TRUE	TRUE	TRUE	TRUE	0	0	0	0	1	5	1	5	1
1	1	1	1	FALSE	FALSE	FALSE	TRUE	0	1	0	0	1		1		
2	2	2	2	FALSE	FALSE	FALSE	TRUE	0	2	0	0	1		1		
3	3	3	3	FALSE	FALSE	FALSE	TRUE	0	3	0	0	1		1		
4	4	4	4	FALSE	FALSE	FALSE	TRUE	0	4	0	0	1		1		
5	5	5	5	FALSE	FALSE	FALSE	FALSE									
6	6	6	6	FALSE	FALSE	FALSE	FALSE									

Tabel 4. Matriks transisi yang terestimasi kembali

A	0
0	1

Tabel 5. Re-estimasi matriks emisi

k	l	i	k < N	l < T	i < states	k	L	i	den+
0	0	0	TRUE	TRUE	TRUE	0	0	0	5
1	1	1	FALSE	TRUE	FALSE	0	1	0	
2	2	2	FALSE	TRUE	FALSE	0	2	0	
3	3	3	FALSE	TRUE	FALSE	0	3	0	
4	4	4	FALSE	TRUE	FALSE	0	4	0	
5	5	5	FALSE	TRUE	FALSE	0	5	0	
6	6	6	FALSE	FALSE	FALSE				

Tabel 6. Matriks emisi yang terestimasi kembali

B	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0,2	1,00E-10	1,00E-10	0,2	…	…	…	…	…	…	0,2	…	…	…	…	0,2	…	0,2	…	…	0,2	…	…	1,00E-10

Tabel 7. Hasil inisialisasi algoritma *forward* setelah parameter HMM terestimasi kembali

I	i < states	fwd(0,i)		c[0]!=0	i < states	c[0]	fwd[0,i]
0	TRUE	0,2		TRUE	TRUE	0,2	1
1	FALSE	-		-	-		-

Tabel 8. Hasil induksi algoritma *forward* setelah parameter HMM terestimasi kembali

t	i	J	t < T	i < states	j < states	t-1	j	I	fwd[t-1,j]	A[i,j]	fwd[t-1,j] * A[i,j]	sum	fwd[t,i]	c[t]	c[t] != 0	i < states	fwd[t,i]
1	0	0	TRUE	TRUE	TRUE	0	0	0	1	1	1	1	0,2	0,2	TRUE	TRUE	1
2	1	1	TRUE	FALSE	FALSE	1	0	0	1	1	1	1	0,2	0,2	TRUE	TRUE	1
3	2	2	TRUE	FALSE	FALSE	2	0	0	1	1	1	1	0,2	0,2	TRUE	TRUE	1
4	3	3	TRUE	FALSE	FALSE	3	0	0	1	1	1	1	0,2	0,2	TRUE	TRUE	1
5	4	4	TRUE	FALSE	FALSE	4	0	0	1	1	1	1	0,2	0,2	TRUE	TRUE	1
6	5	5	FALSE	FALSE	FALSE												

Tabel 9. Hasil inisialisasi algoritma *backward* setelah parameter HMM terestimasi kembali

i	i < states	bwd[T-1,i]
0	TRUE	5
1	FALSE	

Tabel 10. Hasil induksi algoritma *backward* setelah parameter HMM terestimasi kembali

t	i	J	t >= 0	i< states	j < states	t	i	j	A[i,j] * B[j,observations[t+1]] * bwd[t+1,j]	sum	sum / c[t]	bwd[t,i]
4	0	0	TRUE	TRUE	TRUE	4	0	0	1	1	5	5
3	1	1	TRUE	FALSE	FALSE	3	0	0	1	1	5	5
2			TRUE			2	0	0	1	1	5	5
1			TRUE			1	0	0	1	1	5	5
0			TRUE			0	0	0	1	1	5	5

Tabel 11. Hasil perhitungan matriks *gamma* setelah parameter HMM terestimasi kembali

t	k	t < T	k < states	t	k	gamma[i][t,k]	s	gamma[i][t,k]
0	0	TRUE	TRUE	0	0	5	5	1
1	1	TRUE	FALSE	1	0	5	5	1
2	2	TRUE	FALSE	2	0	5	5	1
3	3	TRUE	FALSE	3	0	5	5	1
4	4	TRUE	FALSE	4	0	5	5	1
5	5	TRUE	FALSE	5	0	5	5	1
6	6	FALSE	FALSE					

Tabel 12. Hasil perhitungan matriks *epsilon* setelah parameter HMM terestimasi kembali

t	k	l	t < T-1	k < states	i < states	t	k	l	epsilon[i][t,k,l]	s+	epsilon[i][t,k,l]
0	0	0	TRUE	TRUE	TRUE	0	0	0	1	1	1
1	1	1	TRUE	FALSE	FALSE	1	0	0	1	1	1
2	2	2	TRUE	FALSE	FALSE	2	0	0	1	1	1
3	3	3	TRUE	FALSE	FALSE	3	0	0	1	1	1
4	4	4	TRUE	FALSE	FALSE	4	0	0	1	1	1
5	5	5	FALSE	FALSE	FALSE						
6	6	6	FALSE	FALSE	FALSE						

Tabel 13. Perhitungan *likelihood* baru setelah parameter HMM terestimasi kembali

t	T<scaling.length	log10	newlikelihood +
0	TRUE	-0,69897	-4,19382
1	TRUE	-0,69897	
2	TRUE	-0,69897	
3	TRUE	-0,69897	
4	TRUE	-0,69897	
5	TRUE	-0,69897	
6	FALSE		

Tabel 14. Hasil evaluasi

i	coefficient.length	i < coef..	log10(coeffi..	likelihood +	hasil akhir
0	6	TRUE	-0,69897	-4,19382	0,015089
1		TRUE	-0,69897		
2		TRUE	-0,69897		
3		TRUE	-0,69897		
4		TRUE	-0,69897		
5		TRUE	-0,69897		
6		FALSE			