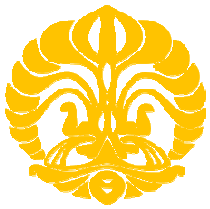


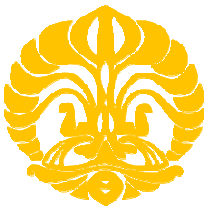
UNIVERSITAS INDONESIA

SISTEM IDENTIFIKASI BIOMETRIK PEMBULUH DARAH TELAPAK TANGAN MENGGUNAKAN METODE HIDDEN MARKOV MODEL

SKRIPSI

DONA ANDIKA SUKMA
0906602585

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JANUARI 2012



UNIVERSITAS INDONESIA

SISTEM IDENTIFIKASI BIOMETRIK PEMBULUH DARAH TELAPAK TANGAN MENGGUNAKAN METODE HIDDEN MARKOV MODEL

SKRIPSI
Diajukan sebagai salah satu syarat untuk memperoleh gelar menjadi Sarjana Teknik

DONA ANDIKA SUKMA
0906602585

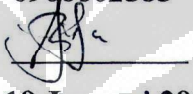
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JANUARI 2012

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul :

SISTEM IDENTIFIKASI BIOMETRIK PEMBULUH DARAH TELAPAK TANGAN MENGGUNAKAN METODE HIDDEN MARKOV MODEL

Adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama	**: Dona Andika Sukma**
NPM	**: 0906602585**
Tanda Tangan	**:** 
Tanggal	**: 19 Januari 2012**

PENGESAHAN

Skripsi diajukan oleh	:
Nama	: **Dona Andika Sukma**
NPM	: **0906602585**
Program Studi	: **Teknik Elektro**
Judul Skripsi	: **Sistem Identifikasi Biometrik Pembuluh Darah Telapak Tangan Menggunakan Metode Hidden Markov Model**

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing	: **Dr. Ir. Dodi Sudiana M.Eng**	( )
Penguji I	: **Dr. Ir. Arman D. Diponegoro**	( )
Penguji II	: **Ir. Purnomo Sidi Priambodo M.Sc., Ph.D.**	( )

Ditetapkan di : **Depok**

Tanggal : **19 Januari 2012**

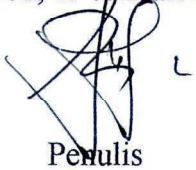
KATA PENGANTAR

Puji Syukur saya panjatkan kepada ALLAH SWT, karena atas Berkat dan Rahmat-Nya, saya dapat menyelesaikan skripsi ini. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelsaikan skripsi ini. Oleh karena itu, saya mengucapkan terima terima kasih kepada :

1. Dr. Ir. Dodi Sudiana M.Eng, selaku dosen pembimbing telah menyediakan waktu, tenaga dan pikiran untuk mengarahkan dalam penyusunan skripsi ini;
2. Dr. Ir. Arman D. Diponegoro M.Eng, yang juga telah memberikan bimbingan dan ilmunya dalam penyusunan skripsi ini;
3. PT. Fujitsu Indonesia, yang telah bersedia memberikan bantuan dalam penelitian ini;
4. Orang tua dan keluarga besar saya yang telah memberikan bantuan dukungan materil dan moril; dan
5. Teman dan sahabat yang telah banyak membantu saya dalam menyelesaikan skripsi ini.

Akhir kata, saya berharap ALLAH SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmupengetahuan.

Depok, 19 Januari 2012



Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan dibawah ini :

Nama	**: Dona Andika Sukma**
NPM	**: 0906602585**
Program Studi	**: Teknik Elektro**
Departemen	**: TEKNIK ELEKTRO**
Fakultas	**: TEKNIK**
Jenis Karya	**: SKRIPSI**

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Sistem Identifikasi Biometrik Pembuluh Darah Telapak Tangan Menggunakan Metode Hidden Markov Model

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 19 Januari 2012

Yang menyatakan

(Dona Andika Sukma)

ABSTRAK

Nama : Dona Andika Sukma
Program Studi : Teknik Elektro
Judul : Sistem Identifikasi Biometrik Pembuluh Darah Telapak Tangan Menggunakan Metode Hidden Markov Model

Skripsi ini berisi tentang pengidentifikasian biometrik melalui pola pembuluh darah telapak tangan dengan menggunakan metode *Hidden Markov Model* (HMM), dengan membandingkan keseluruhan sistem terhadap perubahan ukuran *codebook* dan jumlah iterasi. Metode HMM secara garis besar terdiri dari dua tahapan proses, yakni proses *training* database, dan proses identifikasi.

Pada sistem pengidentifikasian ini, gambar pembuluh darah telapak tangan yang digunakan adalah gambar dari *database CASIA-MS-PalmprintV1* yang dikumpulkan oleh *Chinese Academy of Sciences Institute of Automation* (CASIA). Gambar tersebut terlebih dahulu diolah dengan menentukan ROI. ROI yang sudah didapatkan kemudian diekstraksi dengan melakukan penambahan kontras, pengubahan gambar ke biner dan melakukan *thinning* terhadap garis-garis yang ada pada gambar sehingga pola pembuluh darah terlihat jelas.

Kata kunci : *Hidden Markov Model*, *codebook*, identifikasi biometrik, pembuluh darah telapak tangan

ABSTRACT

Name : Dona Andika Sukma
Study Program : Electrical Engineering
Title : Palm Vein Biometric Identification System Using Hidden Markov Model Method

This thesis contains a biometric identification through palm vein patterns using Hidden Markov Models (HMM), by comparing the overall system to changes in the size of the codebook and the number of iterations. HMM method mainly consists of two stages of the process, first one is database training process, and the identification process.

This identification system is using palm vein images from Casia-MS-PalmprintV1 database that collected by the Chinese Academy of Sciences Institute of Automation (Casia). First, images are processed by determining the ROI. ROI then extracted by adding contrast, convert to binary image and do the thinning of the lines in the image so that the pattern of vein clearly visible.

Key words : Hidden Markov Model, codebook, biometric identification, palm vein

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Sistem biometrik pada dasarnya adalah sebuah sistem pengenalan pola yang mengenali seseorang berdasarkan fitur vektor yang berasal dari karakteristik fisiologis atau perilaku tertentu yang dimiliki seseorang [1].

Dewasa ini sudah banyak sekali sistem identifikasi biometrik yang berkembang, diantaranya identifikasi biometrik melalui sidik jari, wajah, iris mata, suara, tanda tangan, dan lain sebagainya. Dan belakangan sedang dikembangkan adalah identifikasi biometrik melalui pembuluh darah telapak tangan.

Setiap individu memiliki pola pembuluh darah telapak tangan yang berbeda-beda, walaupun bagi orang yang kembar identik sekalipun. Pola pembuluh darahnya pasti tidak akan sama, dan pola itu tidak berubah sepanjang hidup manusia. Hal itu menjadi latar belakang para peneliti untuk menggunakan pola pembuluh darah telapak tangan sebagai cara untuk mengidentifikasi seseorang. Ditambah lagi pembuluh darah sangat tidak memungkinkan untuk dipalsukan. Tidak seorangpun yang bisa memanipulasi pembuluh darahnya.

Pembuluh darah tidak dapat dilihat secara kasat mata, tetapi dapat dideteksi dengan menggunakan spektrum *near-infrared* yang mana darah dan aliran darah menimbulkan perbedaan temperatur dengan jaringan disekitarnya. Perbedaaan tersebut dapat diperjelas dengan menggunakan pencahayaan *near-infrared* [2]. Telapak tangan adalah bagian tubuh ideal dari teknologi ini, yang mana tidak memiliki rambut yang bisa menghalangi pemotretan pola pembuluh darah dan merupakan bagian yang tidak rentan terhadap perubahan warna kulit, tidak seperti halnya jari atau bagian punggung tangan [1].

Kelebihan identifikasi biometrik pola pembuluh darah telapak tangan dibandingkan biometrik lainya adalah [3]:

1. Dapat diaplikasikan ke setiap orang.
2. Dapat digunakan tanpa adanya kontak langsung (*contactless*) dan bersifat higienis, tidak seperti sidik jari yang mengharuskan adanya kontak langsung dan pembersihan sensor secara berkala.

3. Dapat bekerja pada tangan yang kotor atau tangan terluka (selama hanya luka pada kulit).
4. Pendaftaran sekali seumur hidup, karena pola pembuluh darah tidak akan ada perubahan sepanjang hidup.
5. Fitur biometrik berada didalam tubuh.
6. Kriteria keamanan yang terjamin.

Metode yang akan digunakan untuk mengidentifikasi pola pembuluh darah telapak tangan pada skripsi ini adalah metode *Hidden Markov Model* (HMM). Pemilihan HMM didasarkan atas kemampuannya yang dapat memodelkan data 2 dimensi seperti citra dengan baik, serta mendapatkan hasil yang lebih teliti.

1.2 Tujuan Penulisan

Tujuan penulisan skripsi ini adalah untuk membangun perangkat lunak biometrik yang mampu mengidentifikasi pola pembuluh darah telapak tangan dengan menggunakan metode *Hidden Markov Model* dan merancang sistem identifikasi menggunakan perangkat lunak pengolah data matematis. Serta dapat mengekstraksi pola pembuluh darah telapak tangan (*Feature Extraction*).

1.3 Batasan Masalah

Perangkat lunak tidak bersifat *real-time*. Gambar pembuluh darah telapak tangan pada penilitian ini akan menggunakan gambar dari *database CASIA-MS-PalmprintV1* yang dikumpulkan oleh *Chinese Academy of Sciences Institute of Automation* (CASIA), yang kemudian dipanggil pada perangkat lunak. Gambar yang dipakai dari *database* adalah hasil pengambilan gambar dengan menggunakan panjang gelombang *near-infrared* sebesar 940nm.

1.4 Metode Penulisan

1. Studi Literatur

Studi literatur sangat berguna untuk memperoleh informasi yang berkaitan dengan penelitian yang akan dibuat, mengacu pada buku-buku pegangan, informasi yang didapat dari internet, jurnal-jurnal dan makalah-makalah yang membahas tentang penelitian tersebut.

2. Perancangan dan Pembuatan Sistem

Proses perancangan merupakan suatu proses perencanaan bagaimana sistem akan bekerja. Berisi tentang proses perencanaan sistem yang berupa perangkat lunak.

3. Pengujian Sistem

Dari sistem yang dibuat, maka dilakukan pengujian terhadap masing-masing bagian dengan tujuan untuk mengetahui kinerja dari sistem tersebut apakah sudah sesuai dengan yang diharapkan atau belum.

4. Pengumpulan Data

Setelah diuji secara keseluruhan sebagai suatu kesatuan sistem, dapat dilihat apakah rancangan software sudah dapat bekerja dengan benar atau masih dibutuhkan beberapa perbaikan. Jika sistem sudah dapat bekerja dengan benar, maka dapat dilakukan pengumpulan data yang dianggap penting dan diperlukan.

5. Penulisan Hasil penelitian

Hasil dari pengujian dan pengumpulan data kemudian dianalisa. Dari sini dapat ditarik kesimpulan dari penelitian yang telah dilakukan.

1.5 Sistematika Penulisan

Pada penulisan skripsi ini terdiri dari 5 (lima) bab, dimana masing-masing bab mempunyai kaitan satu sama lain yaitu :

BAB 1. PENDAHULUAN

Pada bab ini berisi tentang latar belakang, tujuan, pembatasan masalah, dan sistematika penulisan.

BAB 2. TINJAUAN PUSTAKA

Tinjauan pustaka berisi landasan-landasan teori sebagai hasil dari studi literatur yang berhubungan dalam perancangan dan pembuatan.

BAB 3. PERANCANGAN SISTEM IDENTIFIKASI PEMBULUH DARAH TELAPAK TANGAN

Proses perancangan merupakan suatu proses perencanaan bagaimana sistem ini akan bekerja. Berisi tentang proses perencanaan software. Pada bagian ini akan membahas perancangan dan pembuatan sistem.

BAB 4. UJI COBA DAN ANALISA

Bab ini berisi tentang implementasi sistem sebagai hasil dari perancangan sistem. Pengujian akhir nantinya akan dilakukan untuk memastikan bahwa sistem dapat berfungsi sesuai dengan tujuan awal. Setelah sistem berfungsi dengan baik maka akan dilanjutkan dengan pengambilan data untuk memastikan kapabilitas dari sistem yang dibuat.

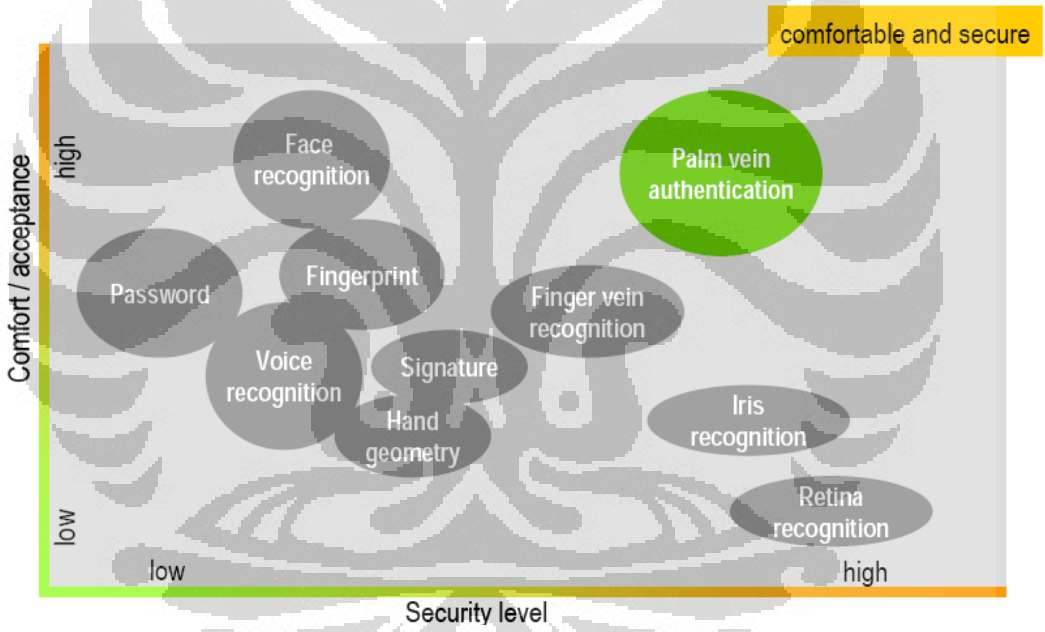
BAB 5. KESIMPULAN

Bab ini berisi kesimpulan yang diperoleh dari perancangan, pembuatan, dan implementasi sistem yang telah dilakukan.

BAB 2
TINJAUAN PUSTAKA

2.1 Identifikasi Biometrik Pembuluh Darah Telapak Tangan (*Palm Vein*)

Biometrik adalah suatu metode pengenalan seseorang berdasarkan karakteristik fisiologis atau perilaku. Fitur-fitur yang biasanya diukur adalah ; wajah, sidik jari, geometri tangan, tulisan tangan, iris, retina, pembuluh darah dan suara. Poin kunci dalam metode identifikasi adalah '*nontransferable*' yaitu tidak dapat diberikan atau dipinjamkan kepada orang lain [4]. Hal lain yang perlu diperhatikan dari berbagai metode identifikasi biometrik adalah tingkat keamanan dan kenyamanan dari metode identifikasi biometrik itu sendiri.

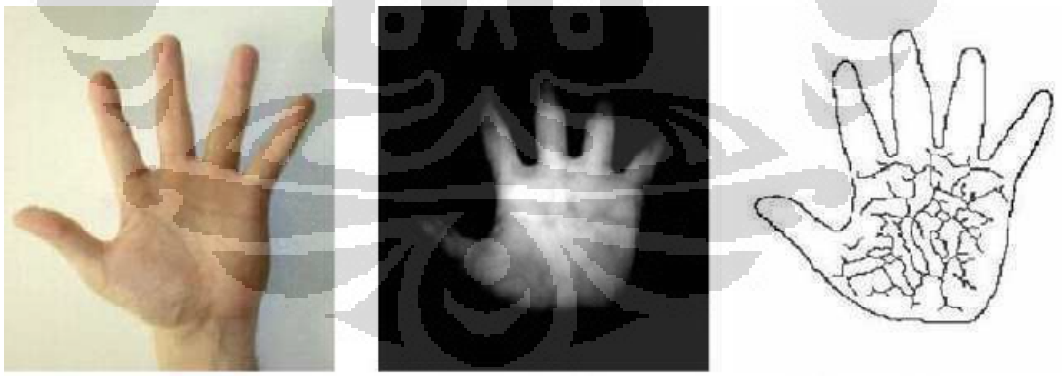


Gambar 2.1 Tingkat keamanan dan kenyamanan dari berbagai metode identifikasi biometrik [5]

Identifikasi pembuluh darah telapak tangan bekerja dengan membandingkan pola pembuluh darah dari seseorang yang akan diidentifikasi dengan pola yang sudah tersimpan didalam *database.* Pola pembuluh darah memiliki keunikan bagi setiap individu, berdasarkan *research* yang telah dilakukan oleh Fujitsu - bahkan kembar identik sekalipun memiliki pola pembuluh darah yang berbeda. Dan karena pola pembuluh darah berada pada bagian dalam tubuh, maka informasinya tidak dapat dicuri atau diduplikasikan

dengan cara mengambilan foto biasa, perekaman suara atau sidik jari, sehingga membuat metode identifikasi ini lebih aman dari pada metode lainnya.

Hemoglobin dalam darah teroksigenasi di paru-paru dan membawa oksigen ke jaringan-jaringan tubuh melalui arteri. Setelah ia melepaskan oksigen ke jaringan, hemoglobin terdeoksidasi kembali ke jantung melalui pembuluh darah. Kedua jenis hemoglobin memiliki tingkat penyerapan yang berbeda. Hemoglobin terdioksidasi menyerap cahaya pada panjang gelombang sekitar 760nm di daerah *near-infrared*. Ketika telapak tangan disinari oleh cahaya *near-infrared*, gambar yang terlihat tidak menyerupai gambar yang dapat dilihat oleh mata manusia (Gambar 2.2 (a)), hemoglobin terdeoksidasi pada pembuluh darah telapak tangan menyerap cahaya tersebut, sehingga mengurangi tingkat refleksi dan menyebabkan pembuluh darah muncul sebagai pola hitam (Gambar 2.2 (b)). Dalam identifikasi pembuluh darah pada prinsip ini, daerah yang digunakan untuk identifikasi adalah hasil foto dengan cahaya *near-infrared,* kemudian pola pembuluh darah diekstrak menggunakan pengolahan citra / *image processing* (Gambar 2.2 (c)) dan diregistrasikan ke *database*. Pola pembuluh darah orang yang akan diidentifikasi kemudian dibandingkan dengan pola yang sudah diregistrasikan sebelumnya.



(a) (b) (c)

Gambar 2.2 Ektraksi pola pembuluh darah telapak tangan, (a) Gambar terlihat oleh mata, (b) Gambar dari sinar *near-infrared*, dan (c) Pola pembuluh darah yang sudah diekstrak [5]

Selain pada telapak tangan, identifikasi pembuluh darah dapat dilakukan dengan menggunakan pola pembuluh darah di punggung tangan atau jari. Namun, pola pembuluh darah telapak tangan adalah paling kompleks dan mencangkup

wilayah terluas. Karena telapak tangan tidak memiliki rambut, akan lebih memudahkan untuk mengambil gambar pola pembuluh darahnya. Telapak tangan juga tidak memiliki variasi warna kulit yang signifikan dibandingkan jari atau punggung tangan, yang mana warna bisa lebih gelap pada daerah tertentu.

Fitur *contactless* membuat perangkat ini cocok digunakan untuk tempat dimana sangat memperhatikan faktor higienis, seperti tempat-tempat umum atau fasilitas medis. Hal ini juga menghilangkan keraguan yang mungkin dimiliki seseorang untuk berkontak dengan sesuatu yang sebelumnya sudah disentuh banyak orang [5].

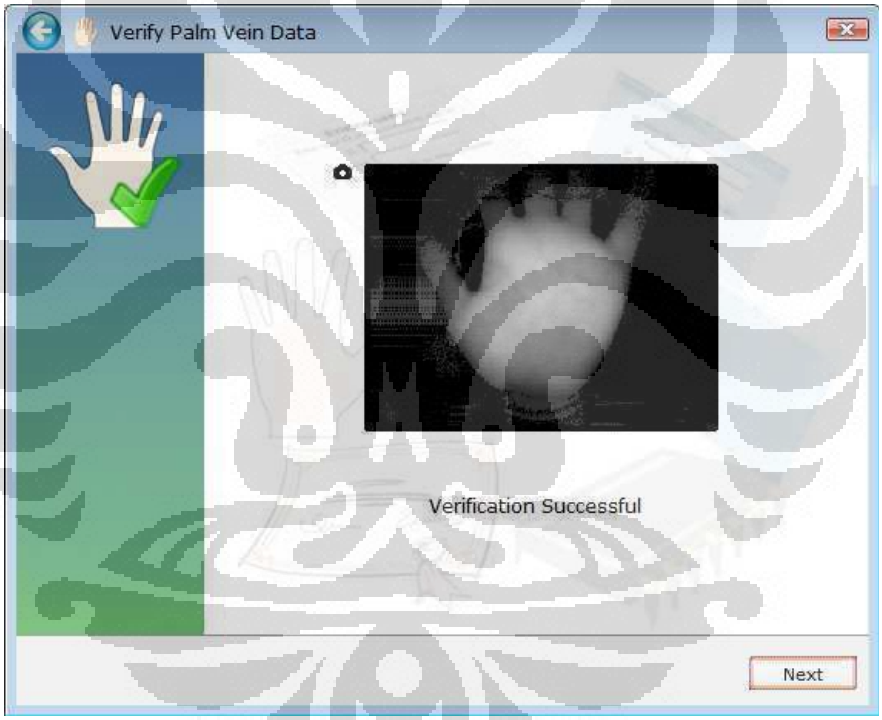
Dalam upaya pengambilan gambar pembuluh darah pada penelitian ini, telah dilakukan berbagai cara, diantaranya dengan menggunakan sensor dalam bentuk *mouse* yang diproduksi oleh PT. Fujitsu dan dikenal dengan *PalmSecure Mouse*. Dan cara lain yaitu pengambilan gambar dengan menggunakan kamera DSLR yang telah terpasang *filter infrared*.

2.1.1 *PalmSecure Mouse* Fujitsu

PalmSecure Mouse yang diproduksi oleh PT. Fujitsu ini merupakan contoh produk dengan pemanfaatan pembuluh darah telapak tangan sebagai metode identifikasi untuk *aplikasi* login pada komputer. *Mouse* ini bekerja dengan menggunakan *software Softex Omnipass Version 7.0*. Tetapi *software* ini hanya dapat bekerja pada komputer dengan *processor* 64bit.

Gambar 2.3 *PalmSecure Mouse* Fujitsu

Cara kerja aplikasi *palmsecure mouse* ini adalah terlebih dahulu memverifikasi data telapak tangan *user*. Telapak tangan yang akan diverifikasi didekatkan ke mouse. Sistem akan mengambil gambar dari pembuluh darah *user* dan melakukan verifikasi. Gambar dari telapak tangan tersebut akan terlihat pada layar aplikasi. Untuk proses *login*, sistem akan meminta *user* untuk mendekatkan telapak tangannya ke *mouse* dan membandingkannya dengan data pembuluh darah yang telah diverifikasi sebelumnya. Apabila data pembuluh darah tersebut sama dengan data yang telah diverifikasi, maka komputer akan *login* dengan sendirinya tanpa harus mengetik *password* lagi. Ini sangat memudahkan user untuk *login* tanpa harus mengingat *password* dan mengetiknya pada saat *login*.



Gambar 2.4 Proses verifikasi *user* pada *software Omnipass* dengan menggunakan *PalmSecure Mouse*

Pada software aplikasi ini, file gambar hasil verifikasi tidak tersimpan dalam sistem. Sehingga pengambilan gambar pada penelitian skripsi ini yang tadinya direncanakan dilakukan dengan menggunakan sensor *mouse* tersebut tidak jadi dilakukan karena tidak tersedianya *file* gambar hasil verifikasi tersebut, dan apabila pengambilan gambar dilakukan dengan *printscreen* pada layar hasil verifikasi, maka gambar pembuluh darah tidak akan terlihat jelas.

2.1.2 Kamera Dengan *Filter Infrared*

Untuk mendapatkan gambar pembuluh darah telapak tangan, cara lain yang dilakukan adalah dengan menggunakan kamera DSLR Canon 1000D dan *filter infrared* HOYA R72 dengan panjang gelombang 720nm. Lensa kamera dipasangkan *filter infrared* dan pencahayaan diatur sedemikian rupa.

Gambar 2.5 *Filter infrared* HOYA R72

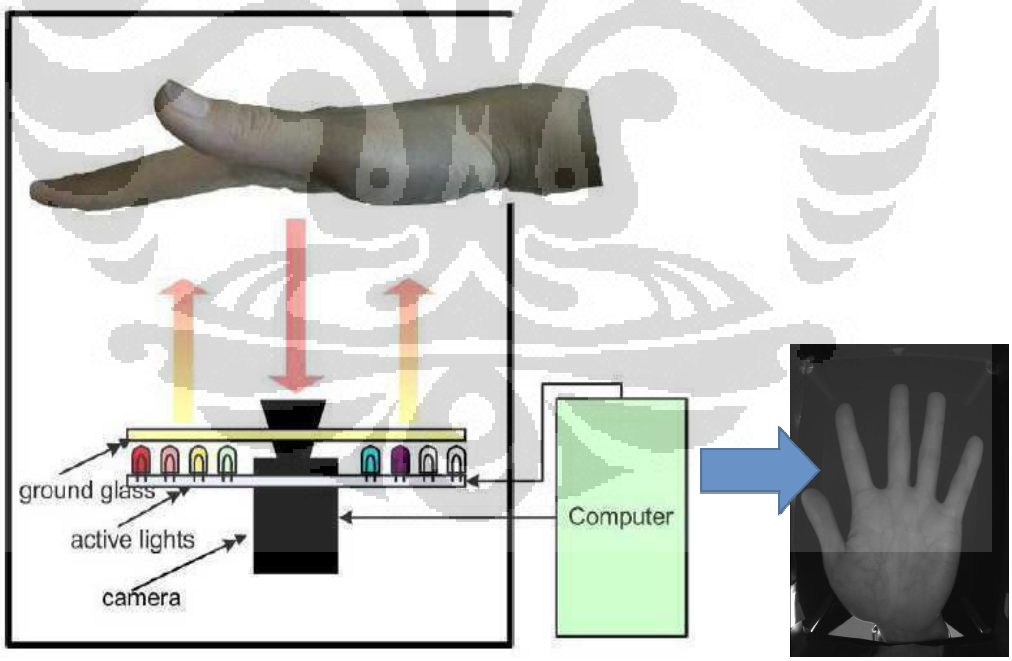
Banyak faktor yang mempengaruhi hasil pengambilan gambar, diantaranya pengaturan cahaya dan *shutter speed* atau waktu *capture* gambar yang lambat dalam upaya pengumpulan cahaya yang dibutuhkan. Sehingga apabila objek yang akan difoto bergerak sedikit saja, maka gambar yang dihasilkan akan *blur.* Dan pola pembuluh darah tidak terlihat dengan jelas.

Gambar 2.6 Hasil pengambilan gambar menggunakan *filter infrared* 720nm

Karena hasil gambar yang *blur* dan pembuluh darah tidak terlalu terlihat, maka pengambilan gambar dengan menggunakan kamera dan *filter infrared* pada penelitian ini juga tidak jadi dilakukan.

2.1.3 *Multi-Spectral Casia palmprint Image Database* V1.0

Multi-Spectral Casia palmprint Image Database V1.0 (atau disingkat dengan Casia-MS-*Palmprint*V1) dirilis dalam rangka untuk mempromosikan penelitian dan kemajuan pada pencitraan *multiple spectral* dari modalitas biometrik. CASIA *Multi-Spectral Palmprint Image Database* terdiri atas 7.200 gambar telapak tangan dari 100 orang yang berbeda dengan menggunakan perangkat pencitraan *multiple spectral* yang dirancang sendiri. Perangkat dilengkapi dengan penerangan secara merata dan pengambilan gambar telapak tangan menggunakan kamera CCD yang terletak pada bagian bawah perangkat. Semua gambar telapak tangan merupakan *file* JPEG dengan tingkat keabuan 8bit [6].



Gambar 2.7 Perangkat pencitraan *multiple spectral* [6]

2.2 Pengolahan Citra

Citra adalah gambar dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinu menjadi gambar diskrit melalui proses digitalisasi. Citra

yang terlihat merupakan cahaya uang direfleksikan dari sebuah objek. Sumber cahaya menerangi objek lalu objek memantulkan kembali sebagian dari berkas cahaya tersebut dan pantulan cayaha ditangkap oleh alat-alat optik, misalnya mata manusia, kamera, *scanner*, sensor satelit dan sebagainya, kemudian direkam. Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpanan magnetik. Citra juga dapat dikelompokkan menjadi dua yaitu : citra tampak seperti foto/gambar, lukisan, dan apa saja yang tampak di layar monitor/televisi, hologram, dan sebagainya, serta citra yang tidak nampak seperti data foto/gambar dalam file, dan citra yang dipresentasikan dalam fungsi matematis. Ada beberapa macam tipe suatu citra didasarkan dari format penyimpanan warnanya, yaitu [7]:

1. Citra biner adalah citra yang setiap pikselnya hanya bernilai 0 (warna hitam) dan 1 (putih).
2. Citra skala keabuan adalah citra yang setiap pikselnya mempunyai kemungkinan warna antara hitam (minimal) dan putih (maksimal),
3. Citra warna (true color) adalah citra yang setiap pikselnya memiliki warna yang merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau dan biru (RGB).
4. Citra warna berindeks adalah citra yang setiap pikselnya mewakili indeks dari suatu tabel warna yang tersedia (biasanya disebit palet warna).

Meskipun sebuah citra kaya informasi, namun seringkali citra yang dimiliki mengalami penurunan mutu (degrasi), misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, kabur (*blurring*), dan sebagainya. Tentu saja citra semacam ini menjadi lebih sulit diinterpretasi karena informasi yang disampaikan oleh citra tersebut menjadi berkurang. Agar citra yang mengalami gangguan mudah diinterpretasi (baik oleh manusia maupun mesin), maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik. Bidang yang menyangkut hal ini adalah pengolahan citra (*image processing)*.

Pada dasarnya ada tiga bidang yang menangani pengolahan data berbentuk citra, yaitu : grafika komputer, pengolahan citra, dan visi komputer. Pada bidang grafika komputer banyak dilakukan proses yang bersifat sintesis yang mempunyai ciri data masukan berbentuk deskriptif dengan keluaran hasil proses yang berbentuk citra. Sedangkan proses di dalam bidang visi komputer merupakan kebalikan dari proses grafika. Terakhir, bidang pengolahan citra merupakan proses pengolahan dan analisis citra dengan data masukan maupun data keluarannya berbentuk citra. Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Pengolahan citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia ataupun mesin (dalam hal ini komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih baik daripada citra masukan.

Operasi-operasi yang dilakukan dalam pengolahan citra banyak ragamnya. Namun, secara umum, pada pengolahan citra terdapat enam jeins operasi pengolahan, yaitu [7]:

1. Peningkatan kualitas citra (*image enhancement*)

   Jenis operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Dengan operasi ini, ciri-ciri khusus yang terdapat di dalam citra lebih ditonjolkan. Contoh-contoh operasi peningkatan kualitas citra :

   a. Perbaikan kontras gelap/terang
   b. Perbaikan tepian objek (*edge enhancement*)
   c. Penajaman (*sharpening*)
   d. Pemberian warna semu (*pseudocoloring*)
   e. Penapisan derau (*noise filtering*)

2. Restorasi citra (*image restoration*)

   Operasi ini bertujuan menghilangkan/meminimumkan cacat pada citra. Tujuan restorasi citra hampir sama dengan operasi peningkatan kualitas citra. Bedanya, pada restorasi citra menyebabkan degradasi gambar diketahui. Contoh-contoh operasi restorasi citra :

   a. Penghilangan kesamaran (*deblurring*)

b. Penghilangan derau (*noise*)

3. Kompresi citra (*image compression*)

   Jenis operasi ini dilakukan agar citra dapat dipresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam kompresi citra adalah citra yang telah dikompresikan harus tetap mempunyai kualitas gambar yang bagus. Contoh metode kompresi citra adalah metode JPEG.

4. Segmentasi citra (*image segmentation*)

   Operasi ini adalah suatu tahap proses analisis citra yang bertujuan untuk memperoleh informasi yang ada dalam citra tersebut dengan membagi citra ke dalam daerah-daerah terpisah dimana setiap daerah adalah homogen dan mengacu pada sebuah kriteria keseragaman yang jelas. Segmentasi yang dilakukan pada citra harus tepat agar informasi yang terkandung di dalamnya dapat diterjemahkan dengan baik.

5. Analisis citra (*image analysis*)

   Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Teknik analisis citra mengekstraksi ciri-ciri tertentu yang membantu dalan identifikasi objek. Proses segmentasi kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya. Contoh-contoh operasi analisis citra :

   a. Pendeteksian tepi objek (*edge detection*)
   b. Ekstraksi batas (*boundary*)
   c. Representasi daerah (*region*)

6. Rekonstruksi citra (*image recontruction*)

   Jenis operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekonstruksi citra banyak digunakan dalam bidang medis. Misalnya beberapa foto *rontgen* dengan sinar *X* digunakan untuk membentuk ulang gambar organ tubuh,

2.2.1 Peningkatan Kualitas Citra

Peningkatan kualitas citra atau *image enhancement* merupakan salah satu proses awal dalam pengolahan citra sebelum aplikasi pengenalan objek di dalam citra. Tujuan dari teknik peningkatan mutu citra adalah untuk melakukan pemrosesan terhadap citra agar hasilnya mempunyai kwalitas relatif lebih baik dari citra awal untuk aplikasi tertentu. Perbaikan ini diperlukan karena citra yang dijadikan objek pembahasan mempunyai kualitas yang buruk, misal [8]:

- Citra mengalami derau pd saat transmisi
- Citra terlalu gelap atau terang
- Citra kurang tajam, kabur dan sebagainya

Operasi citra digital pada dasarnya adalah memanipulasi elemen-elemen matriks. Elemen matriks yang dimanipulasi dapat berupa:

- Elemen tunggal (sebuah *pixel*)
- Sekumpulan elemen yang berdekatan
- Keseluruhan elemen matriks

Cara paling mudah untuk melakukan peningkatan mutu pada domain spasial adalah dengan melakukan pemrosesan yang hanya melibatkan satu piksel saja (tidak menggunakan jendela ketetanggaan). Pengolahan menggunakan *histogram* juga termasuk dalam bagian *point processing*.

Operasi titik disebut juga operasi *pointwise*, terdiri dari:

- Pengaksesan *pixel* pada lokasi yang diberikan
- Memodifikasinya dengan operasi linier atau non linier
- Menempatkan nilai pixel baru pada lokasi yang bersesuaian di dalam citra yang baru

Secara matematis dapat dituliskan sebagai berikut:

$$f_B(x, y) = O_{titik}\{f_A(x, y)\} \tag{2.1}$$

Operasi ini dapat dibagi menjadi 3 macam [8]:

1. Berdasarkan intensitas
   - *Contrast stretching*
   - *Image negative*
   - *Histogram equalization*

- *Image Substration*
- *Image Averaging*

2. Berdasarkan geometri
   - Posisi pixel diubah ke posisi yang baru, sedangkan intensitasnya tidak berubah.
   - Contoh : rotasi, translasi, penskalaan(dilatasi), distorsi geometri
3. Gabungan keduanya
   - Operasi ini tidak hanya mengubah nilai intensitas *pixel*, tapi juga mengubah posisinya.
   - Misal: *image morphing* yaitu perubahan bentuk objek beserta intensitasnya.

2.2.2 *Region Of Interest*

Region of Interest (ROI) merupakan salah satu fitur yang tersedia dalam JPEG2000. ROI memungkinkan dilakukannya pengkodean secara berbeda pada area tertentu dari citra dijital, sehingga mempunyai kualitas yang lebih baik dari area sekitarnya (*background*). Fitur ini menjadi sangat penting, bila terdapat bagian tertentu dari citra dijital yang dirasakan lebih penting dari bagian yang lainnya.

Untuk melakukan ROI, perlu diidentifikasi koefisien-koefisien yang termasuk dalam ROI. Hal tersebut dimaksudkan untuk menghasilkan sebuah ROI *mask*, yang merupakan indikator dari koefisien yang termasuk ROI atau bukan. Umumnya yang digunakan sebagai penanda merupakan sebuah angka biner, yang dimiliki oleh piksel-piksel yang termasuk area ROI. Penanda tersebut berperan sebagai pemetaan area dari citra dijital yang termasuk ROI. Pada proses DWT pun, *mask* tersebut akan berubah, mengikuti transformasi yang dilakukan pada citra dijital. Sehingga pada akhirnya akan diperoleh informasi mengenai koefisien yang termasuk ROI.

2.3 *Discrete Fourier Transform* (DFT) dan *Fast Fourier Transform* (FFT)

Discrete Fourier Transform (DFT) digunakan untuk mengubah *frame-frame* dari domain spasial ke domain frekuensi. Transformasi diskrit merupakan

transformasi dimana input dan output bernilai diskrit yang digunakan untuk manipulasi di komputer. Rumus DFT untuk mengubah *N* data dari domain spasial ke domain frekuensi adalah sebagai berikut [9]:

$$F(u)=\frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N} \tag{2.2}$$

dimana :

F(u) : transformasi fourier

f(x) : fungsi image dalam domain spasial

N : jumlah data

Fast Fourier Transform (FFT) merupakan algoritma yang lebih cepat dari Discrete Fourier transform (DFT). FFT dapat mereduksi jumlah perhitungan untuk setiap *N* data yang sama pada perhitungan DFT sehingga perhitungan yang ada menjadi lebih cepat khususnya ketika nilai *N* yang digunakan cukup besar menggunakan persamaan:

$$F(u)=\frac{1}{2}\left[\frac{1}{M}\sum_{x=0}^{M-1} f(2x)W_M^{ux}+\frac{1}{M}\sum_{x=0}^{M-1} f(2x+1)W_M^{ux}W_{2M}^{u}\right] \tag{2.3}$$

W_M^{ux} dapat dituliskan sebagai

$$W_M^{ux} = e^{-j2ux\pi/M} \tag{2.4}$$

W_{2M}^{u} dapat dituliskan sebagai

$$W_{2M}^{u} = e^{-j\pi u/M} \tag{2.5}$$

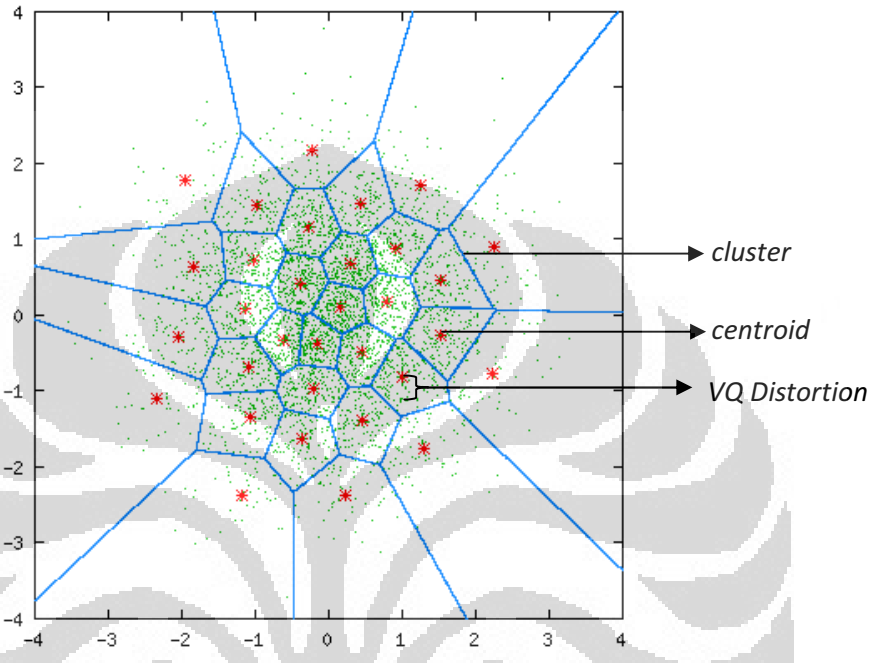
Dimana : $M = \frac{1}{2}N$

Baik DFT maupun FFT akan menghasilkan spektrum frekuensi berupa kumpulan titik-titik dimana masing-masing titik terdiri dari komponen *real* (fungsi *Cosinus*) dan komponen imajiner (fungsi *sinusoidal*) disebabkan adanya bilangan eksponensial [9].

2.4 Vektor Kuantisasi

Vector quantization (VQ) adalah proses pemetaan vektor data yang merupakan titik-titik hasil dari proses FFT ke dalam sebuah wilayah yang

terbatas dalam grafik dua dimensi (X-Y) dimana sumbu X merupakan komponen *real* dari masing-masing titik dan sumbu Y merupakan komponen imajiner dari masing-masing titik. Pemetaan titik-titik tersebut ditunjukkan oleh Gambar 2.8 [9].



Gambar 2.8 Pemetaan pada proses vektor kuantisasi [9]

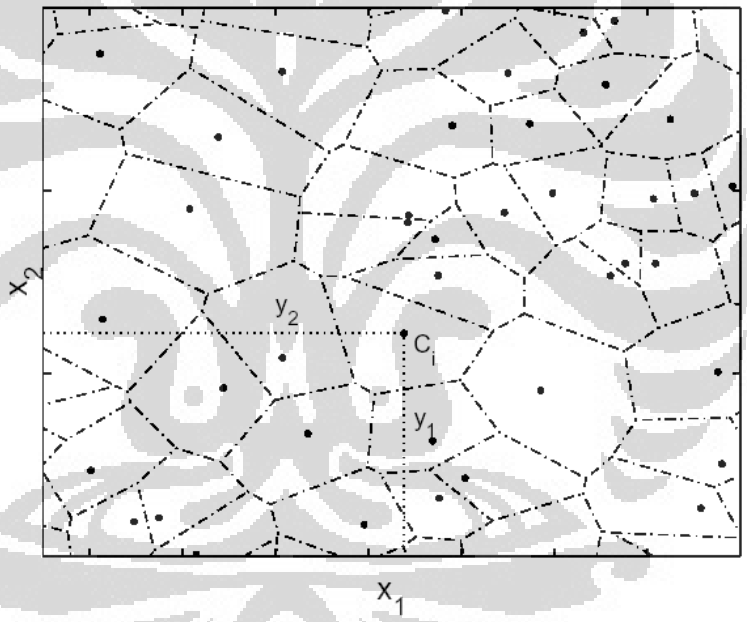
Tujuan dari proses vektor kuantisasi adalah untuk menyederhanakan panjang data masukan agar proses selanjutnya menjadi lebih mudah. Tiap komponen dari spketrum frekuensi yang merupakan hasil FFT memiliki beberapa titik yang masing-masing memiliki komponen real dan imajiner. Kumpulan dari titik-titik yang memiliki jarak berdekatan membentuk suatu *cluster* dan setiap *cluster* yang terbentuk dapat direpresentasikan dengan *centroid* yang disebut *codeword*. Koleksi dari semua *codeword* disebut *codebook*. Jarak antara satu titik dengan titik lain dalam sebuah *cluster* disebut VQ *Distortion*. Semakin kecil VQ *Distortion* nya, maka *cluster* yang terbentuk menjadi lebih akurat.

Luas daerah *cluster* ditentukan oleh ukuran *codebook* dimana semakin besar ukuran *codebook* nya, maka luas daerah masing-masing *cluster* menjadi lebih kecil dan jumlah *cluster* yang terbentuk menjadi lebih banyak disertai nilai

VQ distortion yang semakin kecil sehingga *codeword* yang terbentuk akan semakin mewakili informasi dari masukannya [9].

VQ diinterpretasikan dengan skalar kuantisasi. Sinyal input akan dikuantisasi menjadi *codebook* $C = \{y_k \mid k = 1,...,N\}$. Hampir keseluruhan sinyal input merupakan sebuah vektor yang harus dikodekan kedalam ruang multidimensi. Gambar 2.9 merupakan contoh ruang dua dimensi dari *codebook*. Gambar 2.9 menunjukan partisi dari ruang multidimensi sebuah input vektor yang dibagi menjadi L wilayah yang dapat dinotasikan sebagai $P = \{C_1, C_{2,}, \dots, C_{L,}\}$ dimana :

$$C_i = \{x \mid d(x, y_i) \leq d(x, y_j), j \neq i\} \tag{2.6}$$



Gambar 2.9 *Codebook* dari suatu input vektor [10]

Dalam pembentukan *codebook* untuk iterasi guna memperbaiki VQ digunakan *General Lloyd Algorithm* (GLA) atau yang sering disebut dengan LBG *Algoritm*. LBG VQ *algorithm* tersebut dapat diimplementasikan dengan prosedur rekursif sebagai berikut:

1. Mendesign suatu vektor *codebook* yang merupakan *centroid* dari keseluruhan vektor *training*.
2. Menjadikan ukuran *codebook* dua kali lipat dengan membagi masing-masing current *codebook* C_n menurut aturan

$$C_n^+ = C_n(1+\varepsilon) \tag{2.7}$$

$$C_n^- = C_n(1+\varepsilon) \tag{2.8}$$

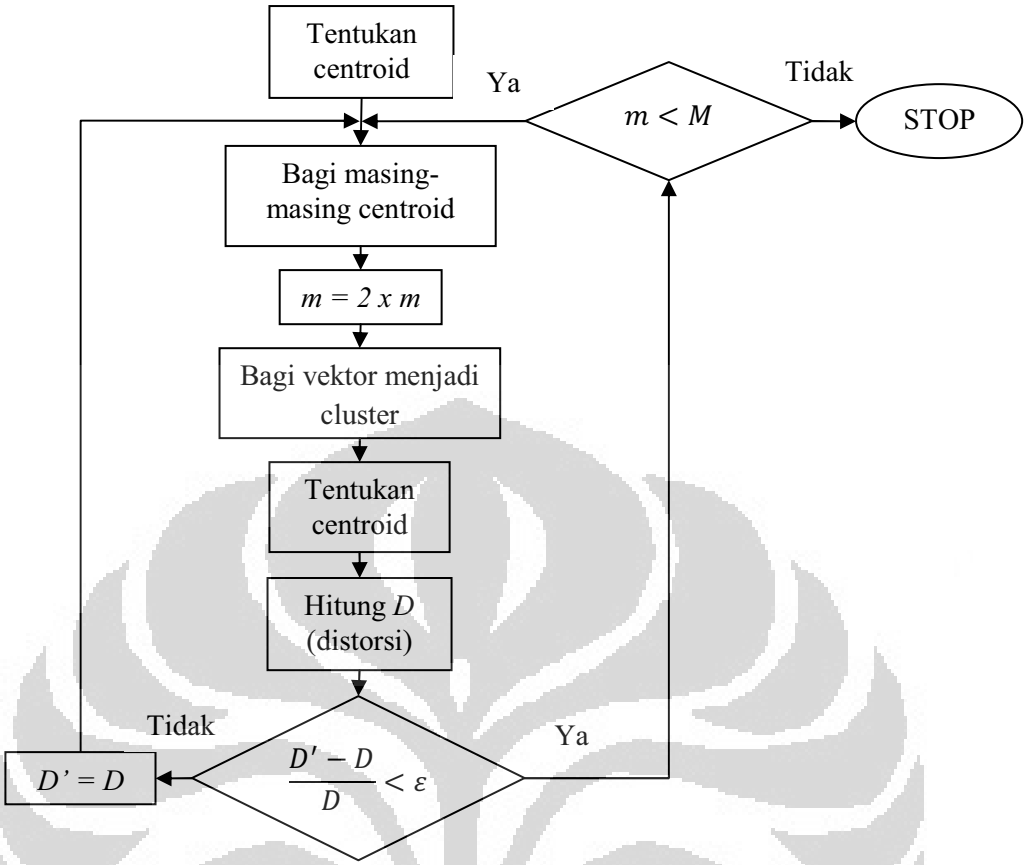
dimana n bervariasi dari 1 sampai dengan *current size codebook* dan ε adalah parameter *splitting* ($\varepsilon = 0.01$).

3. *Nearest Neighbour Search*, yaitu mengelompokan *training vector* yang mengumpul pada blok tertentu. Selanjutnya menentukan *codeword* dalam current *codebook* yang terdekat dan memberikan tanda vektor yaitu *cell* yang diasosiasikan dengan *codeword* yang terdekat.
4. *Centroid update*, yaitu menentukan *centroid* baru yang merupakan *codeword* yang baru pada masing-masing *cell* dengan menggunakan *training vector* pada *cell* tersebut.
5. Iterasi 1

   mengulang step 3 dan 4 sampai jarak rata-rata dibawah present *threshold*.
6. Iterasi 2

   mengulang step 2, 3, 4 sampai *codebook* berukuran *M*.

Gambar 2.10 menunjukan diagram alur, langkah detail dari LBG *algorithm*. *Cluster* vektor menerapkan prosedur *nearest neighbour search* yang mana menandai masing-masing training vektor ke sebuah *cluster* yang diasosiasikan dengan *codeword* terdekat. '*Find centroid*' merupakan prosedur meng-*update centroid* untuk menentukan *codeword* yang baru. '*Compute D (distortion)*' berarti menjumlah jarak semua *training* vektor dalam *nearest neighbour search* terhadap *centroid* untuk menentukan besarnya *distortion* [10].



Gambar 2.10 Diagram alur dari LBG *algorithm* [10]

2.5 *Hidden Markov Model*

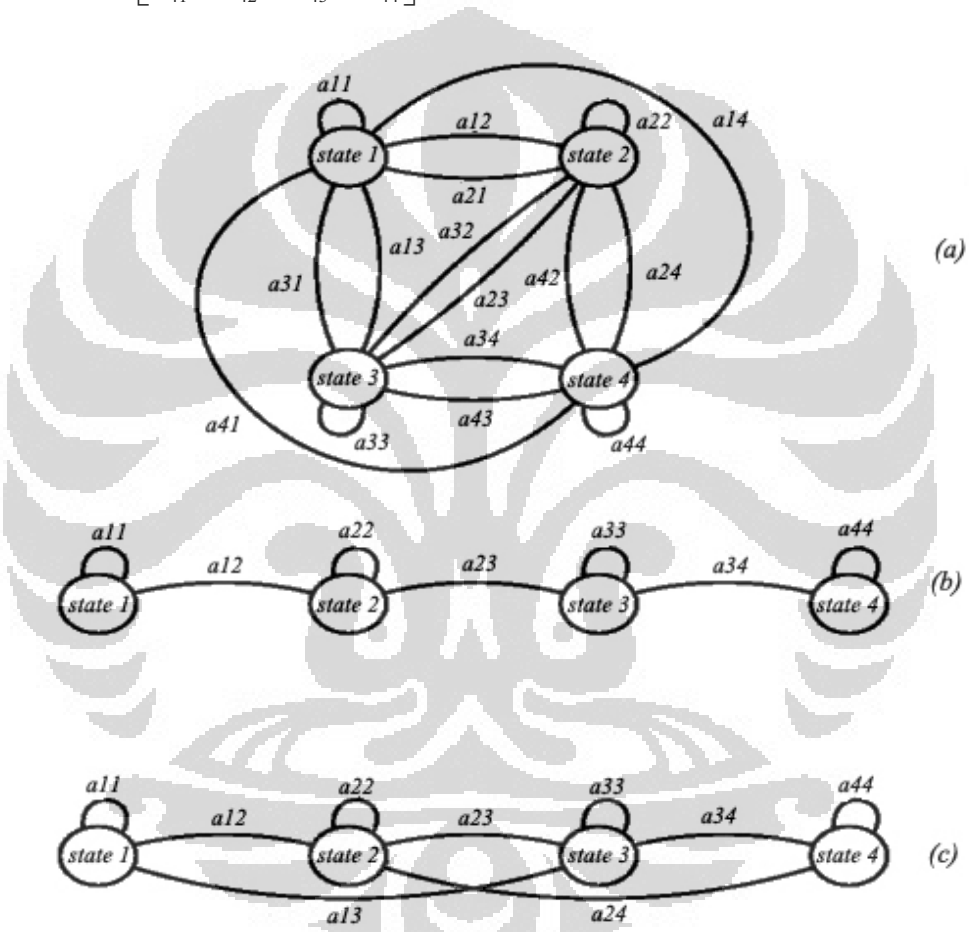
Hidden Markov models (HMM) merupakan model dengan pendekatan statistik yang digunakan dalam berbagai implementasi pengenal gambar dan suara. *Time variance* dalam suatu bahasa dimodelkan sebagai proses *Markov* dengan *discrete state*. Masing-masing *state* menghasilkan observasi menurut karakteristik distribusi probabilitas dari *state* tersebut. Observasi dapat mengambil pada harga diskrit atau kontinyu. Observasi merepresentasikan durasi waktu yang tetap yang disebut *frame*. Pada model ini *state* tidak secara langsung dapat diamati, hal ini yang menjadikan model ini disebut sebagai *hidden Markov model* [10].

2.5.1 Tipe-tipe *Hidden Markov Model*

Salah satu cara untuk mengklasifikasikan HMM adalah dengan melihat bentuk matrix transisinya (A) dari rantai markov. Bentuk yang umum adalah bentuk *ergodic* atau bentuk yang setiap *state* saling terhubung (*fully connected*

HMM). Seperti terlihat pada Gambar 2.11,a untuk N = 4 state model, model ini mempunyai nilai aij antara 0 dan 1. Nilai 0 dan 1 tidak termasuk, jika tidak maka bentuk model ergodic tidak akan terwujud. Matriks transisi untuk ergodic model dapat dimisalkan seperti dibawah ini.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{2.9}$$



Gambar 2.11 HMM dengan 4 state[10]

Untuk pengidentifikasian citra yang tepat digunakan adalah model *left-right* HMM atau biasa disebut Bakis Model. Seperti terlihat pada Gambar 2.11b,c [10].

2.5.2 Elemen- elemen Hidden Markov Model

Elemen-elemen *Hidden Markov Model* meliputi [10] :

1. *N*, jumlah *state* dalam model. Umumnya *state* dapat diinterkoneksi, sehingga setiap *state* dapat dicapai dari *state* yang lain. *State* individual dinotasikan sebagai $S = \{S_1, S_2, \ldots S_N\}$ dan *state* pada waktu *t* adalah q_t.
2. *M*, jumlah observasi simbol yang berbeda tiap *state*. Simbol-simbol tersebut dapat dinotasikan dalam $V = \{v_1, v_2, \ldots v_M\}$.
3. $A = \{a_{ij}\}$, distribusi probabilitas transisi *state*, dimana
$$a_{ij} = P\left[q_{t+1} = S_j \middle| q_t = S_i\right], 1 \le i, j \le N \quad (2.10)$$
4. $B = \{b_j(k)\}$, distribusi probabilitas simbol observasi pada *state* j, dimana
$$b_j(k) = P\left[v_k \_pada\_t \middle| q_t = S_j\right], 1 \le j \le N, 1 \le k \le M \quad (2.11)$$
5. $\pi = \{\pi_i\}$, distribusi *state initial*, dimana
$$\pi_i = P[q_1 = S_i], 1 \le i \le N \quad (2.12)$$

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 $O = O_1 O_2 \ldots O_T$ dimana masing-masing observasi O_t adalah simbol dari *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 $O = O_1 O_2 \ldots O_T$ dan sebuah model $\lambda = (A,B,\pi)$.

Misalkan diberikan urutan *state*

$$Q = q_1 q_2 \ldots q_T \quad (2.13)$$

dimana q_1 adalah inisial *state*. Dengan demikian probabilitas urutan observasi *O* untuk urutan *state* pada persamaan (2.13) adalah

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(O_t \mid q_t, \lambda) \quad (2.14)$$

sehingga didapatkan

$$P(O \mid Q, \lambda) = b_{q_1}(O_1).b_{q_2}(O_2)\ldots b_{q_T}(O_T) \quad (2.15)$$

Probabilitas dari urutan *state Q* dapat dituliskan

$$P(Q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T} \quad (2.16)$$

Probabilitas gabungan dari O dan Q yaitu probabilitas dari O dan Q yang terjadi secara bersamaan. Probabilitas gabungan ini dapat dituliskan

$$P(O,Q \mid \lambda) = P(O \mid Q,\lambda)P(Q \mid \lambda) \tag{2.17}$$

Probabilitas observasi O yang diberikan, diperoleh dengan menjumlahkan seluruh probabilitas gabungan terhadap semua kemungkinan urutan *state* q, yaitu

$$P(O \mid \lambda) = \sum_{allQ} P(O \mid Q,\lambda)P(Q \mid \lambda) \tag{2.18}$$

$$= \sum_{q_1 q_2 \cdots q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} . b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(O_T) \tag{2.19}$$

Untuk menghitung persamaan (2.19) dengan menggunakan prosedur *forward*. Variabel *forward* $\alpha_1(i)$ didefinisikan sebagai probabilitas sebagian urutan observasi $O_1 O_2 \ldots O_t$ (hingga waktu t) dan *state* S_i pada waktu t, dari model λ yang diberikan.

$$\alpha_t(i) = P(O_1 O_2 \ldots O_t, q_t = S_i \mid \lambda) \tag{2.20}$$

untuk menyelesaikan $\alpha_1(i)$ adalah sebagai berikut :

a. Inisialisasi

$$\alpha_1(i) = \pi_i b_i(O_1) \quad 1 \le i \le N \tag{2.21}$$

b. Induksi

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \qquad 1 \le t \le T-1, \quad 1 \le j \le N \tag{2.22}$$

c. Terminasi

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.23}$$

Dari nilai probabilitas yang telah didapatkan, dapat dihitung nilai *Log of Probability*-nya (LoP) :

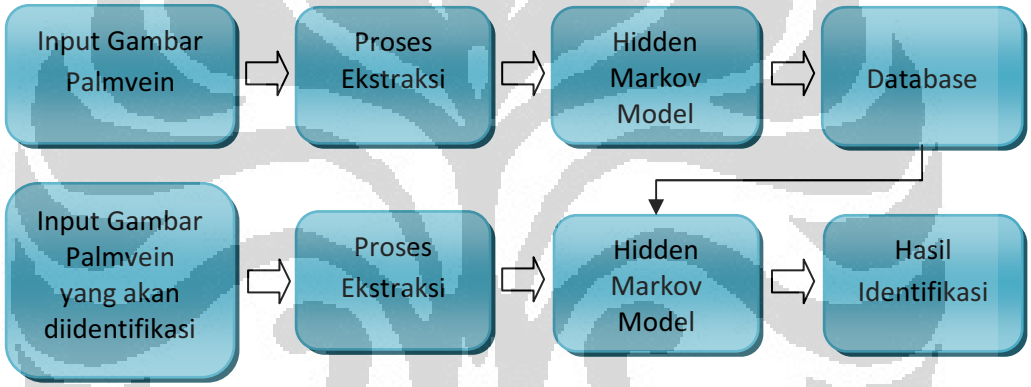
$$LoP = Log\ P(O \mid \lambda) \tag{2.24}$$

BAB 3
PERANCANGAN SISTEM IDENTIFIKASI PEMBULUH DARAH TELAPAK TANGAN

3.1 Prinsip Kerja Sistem

Dalam perancangan perancak lunak sistem identifikasi biometrik pembuluh darah telapak tangan ini terdapat dua proses utama, pertama yaitu proses *training* atau pembuatan *database*, kedua yaitu proses identifikasi. Dan metode yang digunakan adalah *Hidden Markov Model*.

Secara garis besar, prinsip kerja sistem dapat dilihat pada blok diagram berikut ini.



Gambar 3.1 Blok Diagram Sistem

Pada bagian atas dari blok diagram merupakan proses *training* untuk mendapatkan parameter HMM dari gambar yang sebelumnya sudah diekstraksi guna mendapatkan pola pembuluh darah dengan jelas. Dan kemudian parameter HMM tersebut disimpan kedalam *database*. Pada proses identifikasi, gambar yang sudah diekstraksi kemudian dicari *Log of Probability* (LoP) HMM-nya dengan menggunakan parameter HMM pada setiap data dalam *database*. Dari hasil LoP akan didapatkan hasil identifikasi yang sesuai dengan gambar yang di*input*kan.

Perangkat lunak ini dirancang dengan menggunakan komputer dengan spesifikasi sebagai berikut :

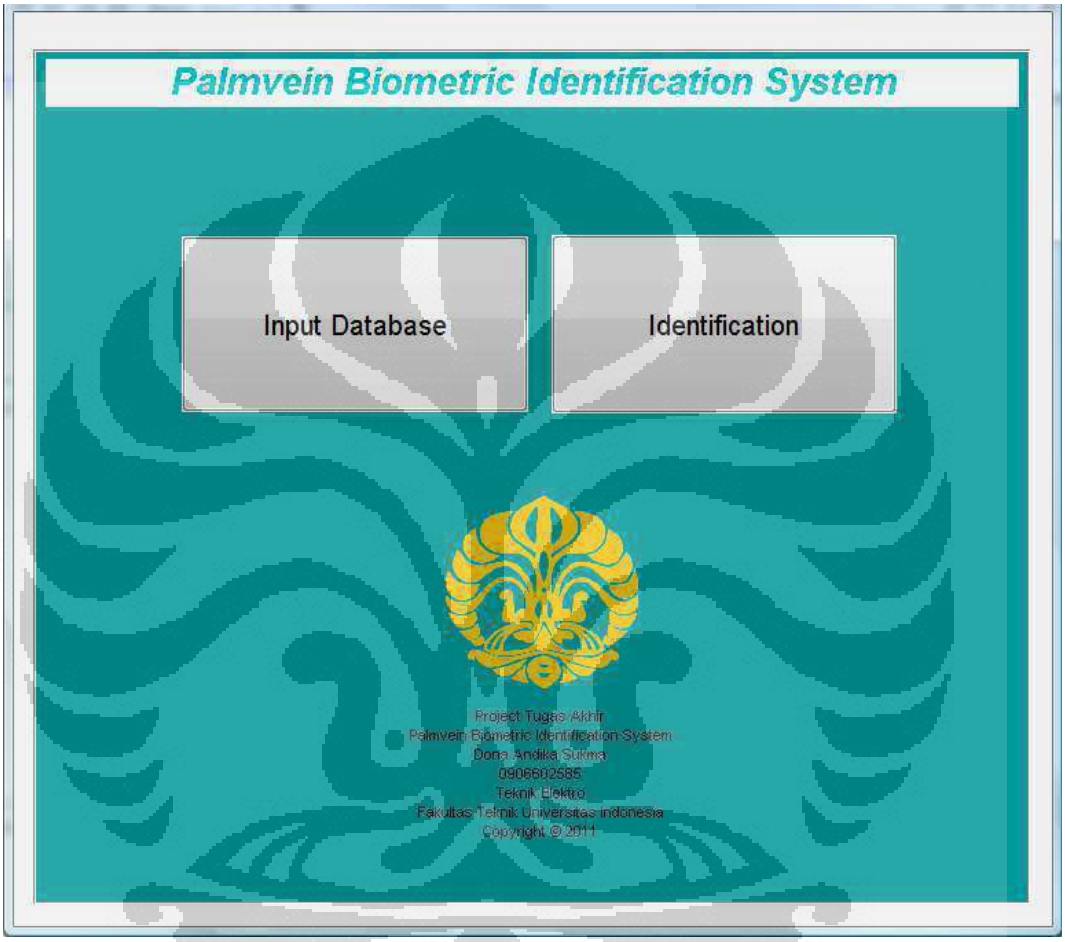
Sistem Operasi : *Windows Vista Home Premium* (32-bit)

Prosesor : Intel Core 2 Duo T5550 1,83GHz

VGA *Card* : Mobile Intel(R) 965 Express Chipset Family

Memory : 3GB

Perangkat lunak dilengkapi dengan GUI untuk mempermudah *user* dalam penggunaannya. Rancangan GUI untuk halaman utama sistem dapat dilihat pada Gambar 3.2.

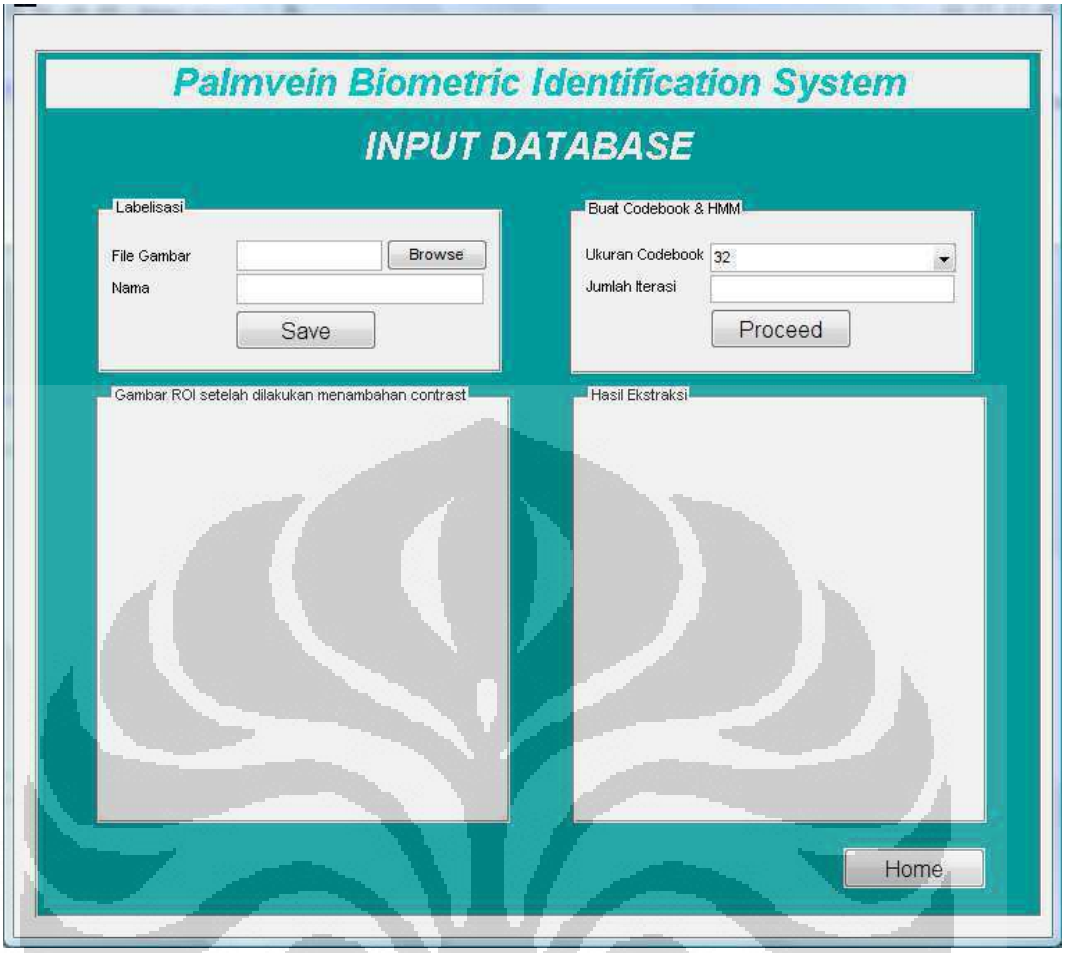


Gambar 3.2 Rancangan GUI untuk halaman utama

3.2 Pembentukan *Database*

Pada proses pembentukan *database* ini terdapat empat tahapan, yakni tahap ekstraksi, pelabelan, pembuatan *codebook*, dan pembentukan *Hidden Markov Model* (HMM). Secara garis besar digambarkan pada diagram alir berikut ini.

Gambar 3.3 Diagram alir pembentukan *database*



Gambar 3.4 Tampilan GUI pembentukan *database*

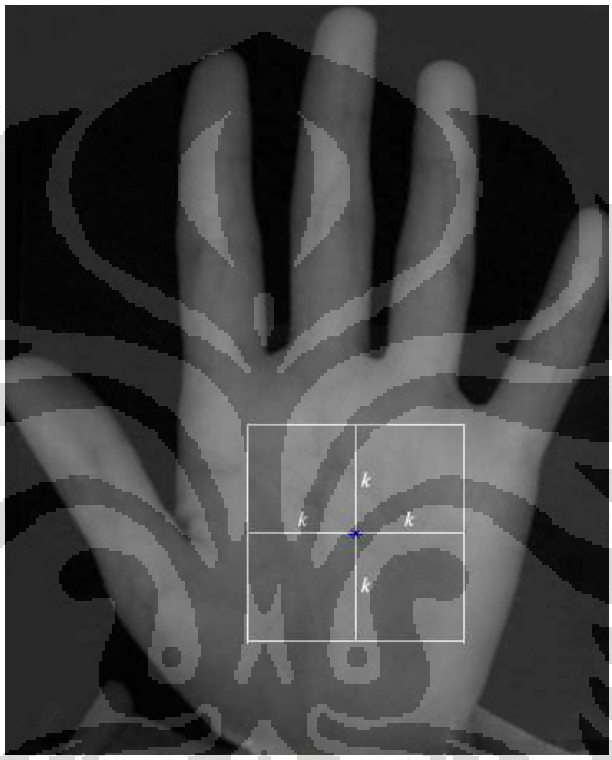
3.2.1 Tahap Ekstraksi Fitur Gambar

Algoritma ekstraksi fitur dijelaskan dalam langkah-langkah sebagai berikut :

```
Mulai
    Input gambar;
    Tentukan daerah ROI;
    Tingkatkan kontras pada gambar;
    Ubah gambar ke biner;
    Filter gambar;
    Thinning gambar;
Selesai
```

Langkah pertama dalam ekstraksi adalah menentukan *Region Of Interest* (ROI) dari telapak tangan yang akan ekstraksi. Pada penelitian ini, ROI ditentukan

berdasarkan titik tengah dari telapak tangan, yaitu dengan menentukan titik tengah dari suatu objek pada gambar. Dari titik tengah tersebut dibuat dua garis sejajar yang berjarak sebesar *k* kearah atas dan bawah dari titik tengah tersebut. Kedua garis itu merupakan batas atas dan bawah dari ROI. Dan dua garis sejajar yang berjarak sebesar *k* kesamping kiri dan kanan. Garis tersebut akan menjadi batas kiri dan kanan dari ROI.

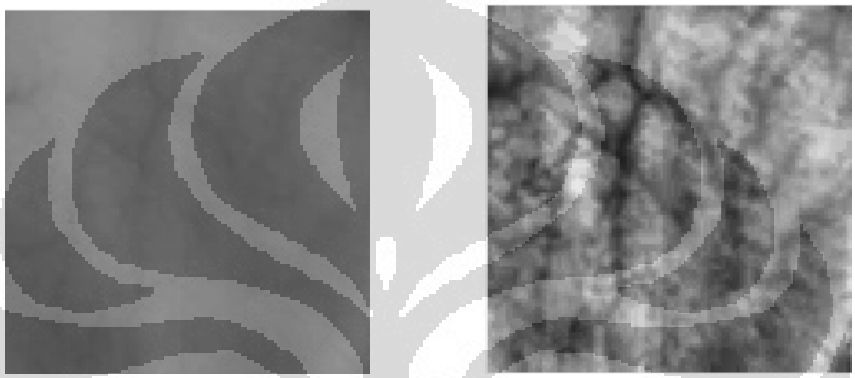


(a)

(b)

Gambar 3.5 Menentukan ROI (a) Penentuan ROI dari telapak tangan, (b) Hasil *cropping* ROI

Setelah menentukan ROI, langkah selanjutnya adalah melakukan perbaikan kualitas citra guna memperjelas pola pembuluh darah pada gambar. Kontras pada gambar ditingkatkan dengan mengubah nilainya menggunakan metode pemerataan histogram adaptif kontras terbatas. Metode ini bekerja pada daerah kecil pada gambar yang disebut *tiles*, bukan pada seluruh bagian gambar. Setiap kontras *tiles* ditingkatkan, sehingga histogram dari output kira-kira cocok dengan histogram yang ditentukan oleh parameter distribusi.



(a) (b)

Gambar 3.6 Peningkatan kualitas citra, (a) Gambar ROI *original*, (b) Gambar yang sudah mengalami peningkatan kontras

Gambar yang sudah ditingkatkan kontrasnya kemudian diubah menjadi gambar biner yang bernilai 1 dan 0.

$$B(x,y) = \begin{cases} 255 & if\ I(x,y) > T \\ 0 & otherwise \end{cases} \quad (3.1)$$

I(x,y) adalah gambar *grayscale* dan *B(x,y)* adalah gambar yang sudah diubah ke dalam bentuk biner. Sedangkan T adalah nilai *threshold*.

Gambar biner tersebut kemudian di*filter* untuk mengurangi *noise* dan menghilangkan daerah yang besarnya kurang dari *range* yang telah ditentukan. Metode *filter* ini bekerja pada gambar biner dengan menghapus atau menghilangkan objek-objek dari gambar yang luasnya lebih kecil dari luas pixel yang telah ditentukan sebelumnya. Sehingga yang tersisa hanyalah objek yang besar saja.

Langkah selanjutnya yaitu melakukan *thinning* atau penipisan garis. Algoritma yang digunakan adalah [11]:

a. Membagi gambar menjadi dua sub bidang berbeda dalam pola kotak-kotak.
b. Pada sub iterasi pertama, hapus *p* pixel dari sub bidang pertama jika dan hanya jika kondisi *G1*, *G2* dan *G3* terpenuhi.
c. Pada sub iterasi kedua, hapus *p* pixel dari sub bidang kedua jika dan hanya jika kondisi *G1*, *G2* dan *G3'* terpenuhi.

Kondisi *G1* :

$$X_H(P) = 1 \tag{3.2}$$

dimana

$$X_H(P) = \sum_{i=1}^{4} b_i \tag{3.3}$$

$$b_i = \begin{cases} 1 \; if \; x_{2i-1} = 0 \; and \; (x_{2i} = 1 \; or \; x_{2i+1} = 1) \\ 0 \; otherwise \end{cases} \tag{3.4}$$

$x_1, x_2, \ldots, x_8$ adalah nilai dari delapan *neighbor p*, dimulai dari *neighbor* timur dan diurutkan searah jarum jam.

Kondisi *G2* :

$$2 \leq min\,\{n_1(P), n_2(P)\} \leq 3 \tag{3.5}$$

dimana

$$n_1(P) = \sum_{k=1}^{4} x_{2k-1} \vee x_{2k} \tag{3.6}$$

$$n_2(P) = \sum_{k=1}^{4} x_{2k} \vee x_{2k+1} \tag{3.7}$$

Kondisi *G3* :

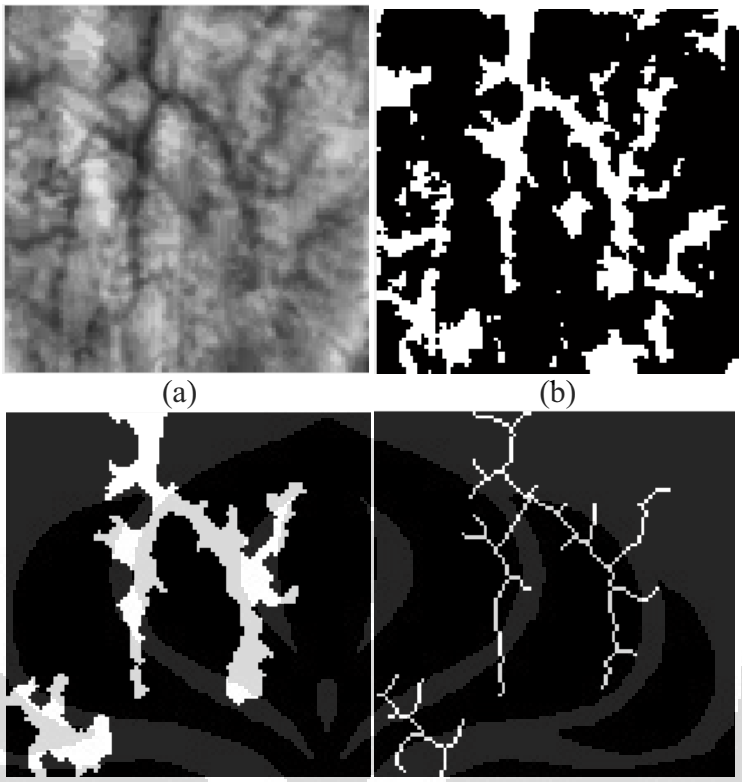
$$(x_2 \vee x_3 \vee \bar{x}_8) \wedge x_1 = 0 \tag{3.8}$$

Kondisi *G3'* :

$$(x_6 \vee x_7 \vee \bar{x}_4) \wedge x_5 = 0 \tag{3.9}$$

Kedua sub iterasi bersama-sama membentuk satu iterasi dari algoritma *thinning*. Ketika ditentukan jumlah iterasi yang tidak terbatas (*n=inf*), iterasi diulang sampai gambar berhenti berubah.

Hasil *thinning* inilah yang merupakan hasil ekstraksi gambar, dimana sudah sangat jelas mempresentasikan informasi dari pola pembuluh darah telapak tangan. Proses dari ekstraksi tersebut dapat dilihat pada Gambar 3.7.



Gambar 3.7 Proses ekstraksi, (a) Target gambar (b) Konversi gambar ke biner (c) Gambar biner yang sudah di*filter* (d) Gambar hasil *thinning*

3.2.2 Pelabelan

Algoritma program untuk proses pelabelan adalah sebagai berikut :

```
Mulai
     Gambar hasil ekstraksi;
     Input nama;
     Simpan matriks gambar hasil ekstraksi dan
     nama pada database;
Selesai
```

Tahap pelabelan merupakan tahap dimana dilakukan pemberian nama bagi setiap *input* gambar yang akan dimasukan ke dalam *database*, dengan kata lain pemberian identitas bagi pemilik pola pembuluh darah yang diinputkan. Nama tersebut di*input*kan pada GUI, dan nama inilah yang nantinya akan menjadi keluaran pada saat proses identifikasi. Pada tahap pelabelan ini, gambar yang sebelumnya sudah diekstraksi akan diubah menjadi matriks *M* x *N* (ukuran matrik

M x *N* akan sama dengan ukuran gambar yang dimasukkan). Nama dan matriks ini kemudian disimpan kedalam *database*.

3.2.3 Pembuatan *Codebook*

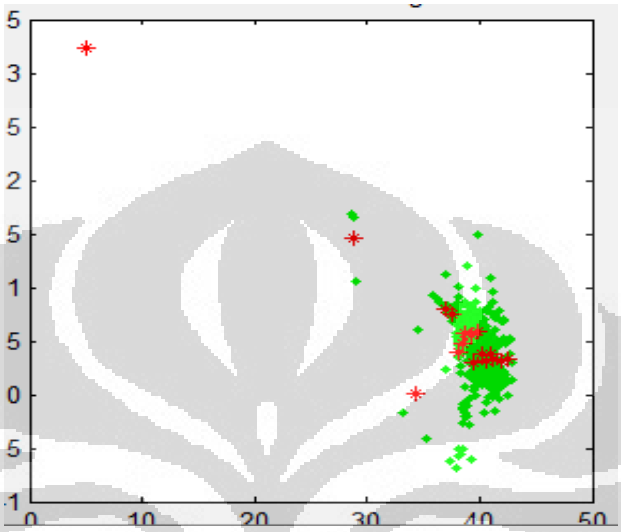
Algoritma program untuk proses pembuatan *codebook* adalah :

```
Mulai
    Input besar codebook dan iterasi;
    Panggil matriks gambar yang sebelumnya sudah
    disimpan pada database;
    Ubah matriks gambar mejadi matriks kolom
    Bagi matriks menjadi frame-frame;
    Hitung FFT untuk setiap frame sehingga
    didapatkan sample point;
    Pengelompokkan sample point menjadi Cluster-
    cluster sebanyak besar codebook yang
    diinputkan;
    Hitung centroid tiap cluster;
    Codebook = matriks dari gabungan centroid;
    Simpan codebook dalam file database;
Selesai
```

Setelah pelabelan dilakukan, tahap selanjutnya yaitu proses pembuatan *codebook*. Data matriks dari hasil pelabelan akan menjadi input pada tahap ini. Matrik diubah menjadi matrik kolom sehingga membentuk sinyal diskrit dan kemudian dibentuk *frame-frame* untuk memecah sinyal menjadi bagian-bagian kecil. Masing-masing *frame* ini akan diubah kedalam bentuk domain frekuensi yang terdiri dari nilai *real* dan imajiner dengan menggunakan *fast fourier transform* (FFT). Data hasil FFT merupakan *codeword* yang kemudian akan dilakukan proses vektor kuantisasi guna mencari *cluster* dan *centroid* pada masing-masing *cluster*. Kumpulan *centroid* inilah yang disebut dengan *codebook*.

Codebook yang sudah didapatkan akan disimpan ke dalam *database*. Penyimpanan *codebook* pada *database* akan dipisahkan antara besar *codebook* dan jumlah iterasi yang berbeda. Hal ini bertujuan untuk memdahkan dalam

perbandingan antara ukuran *codebook* dan jumlah iterasi yang berbeda. Jumlah iterasi merupakan banyaknya proses pengulangan yang dilakukan dalam menentukan *centroid* guna mendapatkan *centroid* yang cukup presisi. Semakin besar iterasinya, maka akan semakin presisi letak *centroid* yang didapat, namun proses pembuatan *codebook* juga akan semakin lama.



Gambar 3.8 Hasil tampilan *codebook*

Jumlah *codebook* yang tersedia dalam program ini adalah 32, 64, 128, 256, 512, dan 1024. Dimana ke enam ukuran *codebook* ini akan dijadikan bahan perbandingan pada skripsi ini untuk dilihat berapa nilai *codebook* yang sesuai pada proses identifikasi pola pembuluh darah tangan.

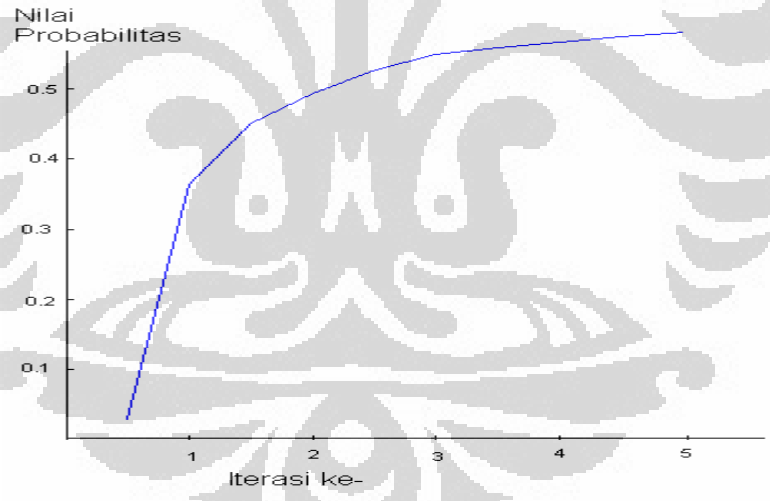
3.2.4 Pembentukan parameter HMM

Algoritma program untuk pembentukan parameter HMM adalah sebagai berikut :

```
Mulai
    Panggil codebook yang sudah dibuat;
    Panggil matriks gambar pada label;
    Pelatihan dengan memasukan data gambar sampai
    nilai matriks tidak berubah;
    Hitung probabilitas A, B untuk setiap gambar;
    Simpan hasil dalam file database;
Selesai
```

Pada proses pembentukan paremeter HMM terdapat beberapa inputan, antara lain : ukuran *codebook*, jumlah iterasi, label dan *codebook* yang sesuai dengan ukuran *codebook* dan jumlah iterasi dimana sudah dibuat pada tahap sebelumnya. Hasil keluaran dari tahap ini dalah parameter HMM : A, B, dan π, dari setiap label yang telah dibuat. Paramater tersebut kemudian disimpan ke dalam *file database* dengan format penulisan nama *file* berdasarkan ukuran *codebook* dan jumlah iterasi yang telah ditentukan, sama halnya dengan tahap pembuatan *database codebook* yang sudah dijelaskan sebelumnya.

Pada proses pembentukan parameter HMM, jumlah iterasi merupakan penentu banyaknya pengulangan yang dilakukan dalam mengolah parameter HMM guna mendapatkan probabilitas yang paling baik. Semakin besar jumlah iterasinya, maka akan semakin baik probabilitas yang didapat. Gambar 3.9 merupakan tampilan grafik probabilitas dalam pembuatan HMM.



Gambar 3.9 Probabilitas satu label *palmvein* dengan 5 iterasi

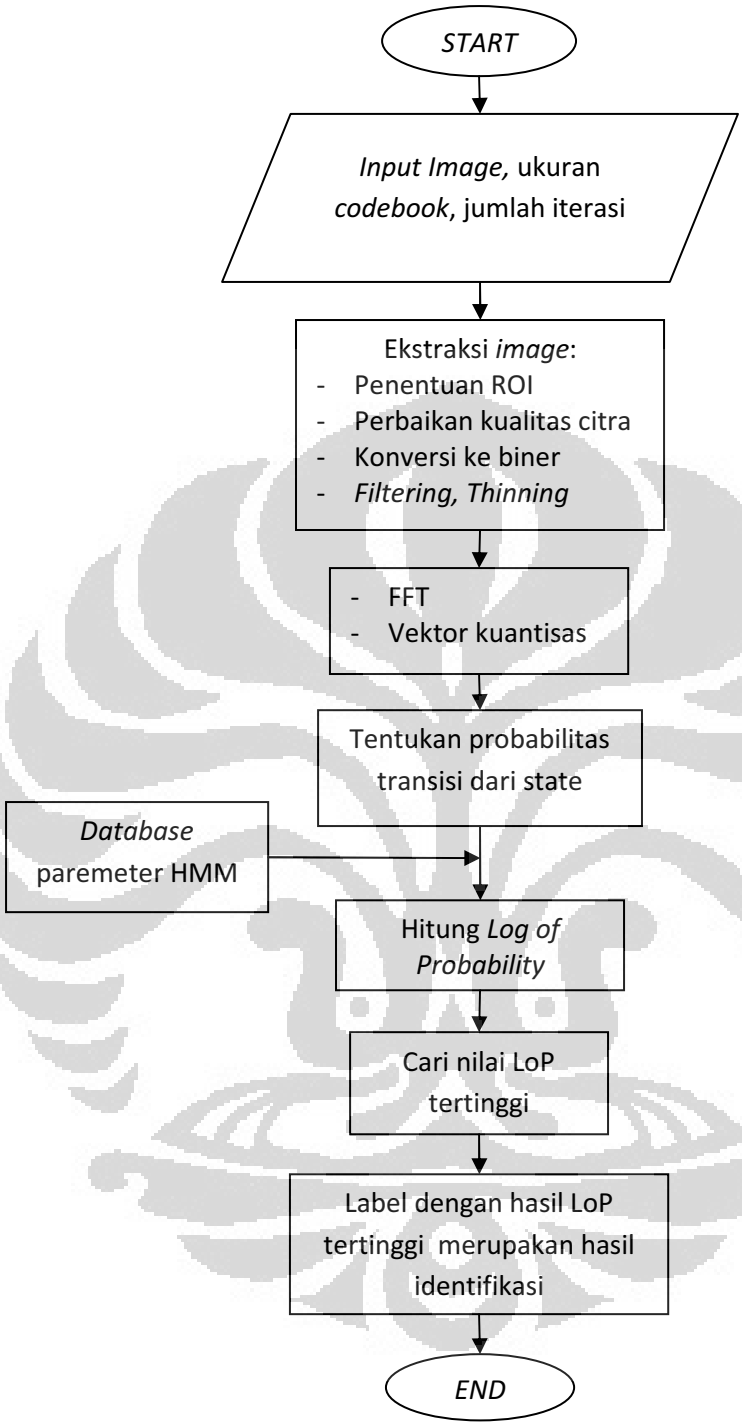
3.3 Proses Identifikasi

Algoritma program dari proses identifikasi adalah sebagai berikut :

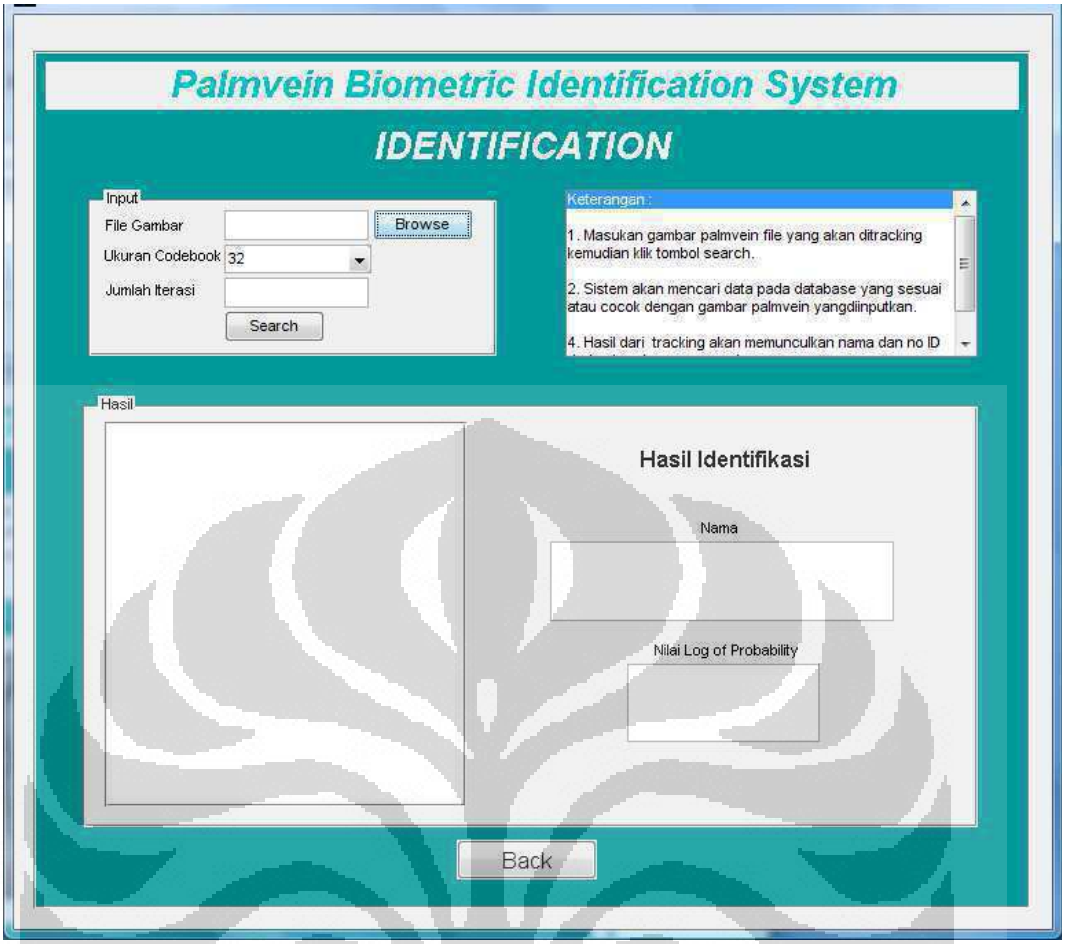
```
Mulai
    Input gambar, codebook dan iterasi;
    Tentukan ROI dari gambar;
    Tingkatkan kontras pad gambar;
```

```
        Ubah gambar ke biner;
        Filter gambar;
        Thinning gambar;
        Bagi matriks gambar kedalam bentuk frame;
        Hitung FFT masing-masing frame;
        Tentukan state;
        Panggil paramater HMM database;
        Hitung LoP dari masing data pada database;
        Hasil identifikasi = nilai maksimum dari LoP;
        Tentukan nama label dari hasil identifikasi;
Selesai
```

Setelah proses pembuatan *database* dilakukan, langkah selanjutnya adalah proses identifikasi dari gambar *input* pembuluh darah telapak tangan dengan menggunakan parameter HMM yang telah disimpan kedalam *database*. Diagram alir proses identifikasi dapat dilihat pada Gambar 3.10.



Gambar 3.10 Diagram alir proses identifikasi



Gambar 3.11 Tampilan GUI identifikasi

Inputkan gambar yang akan diidentifikasi terlebih dahulu, beserta ukuran *codebook* dan jumlah iterasinya. Sama halnya pada proses pembentukan *database*, tentukan ROI gambar *input* dan kemudian diekstraksi terlebih dahulu untuk mendapatkan pola pembuluh darahnya. Matriks hasil ekstraksi diubah kedalam bentuk domain frekuensi guna mendapatkan nilai *real* dan imajinernya menggunakan *fast fourier transform* (FFT). Hasil FFT tersebut dipetakan dalam bentuk *sample point*. Selanjutnya dari *sample point* dikuantisasikan ke satu titik *centroid* pada *codebook* untuk dicari statenya. State yang didapat akan dicocokkan dengan matriks-matriks dari parameter HMM dalam *database*. Dari parameter-parameter HMM tersebut dihitung besar *Log of Probability* (LoP) untuk semua data pada *database*. Gambar pada *database* yang memiliki kemiripan dengan gambar yang diinputkan akan memiliki nilai LoP yang lebih besar dibandingkan dengan data tidak ada kemiripan. *Database* dengan nilai LoP terbesar merupakan hasil identifikasi dari gambar yang diinputkan.

BAB IV
UJI COBA DAN ANALISA

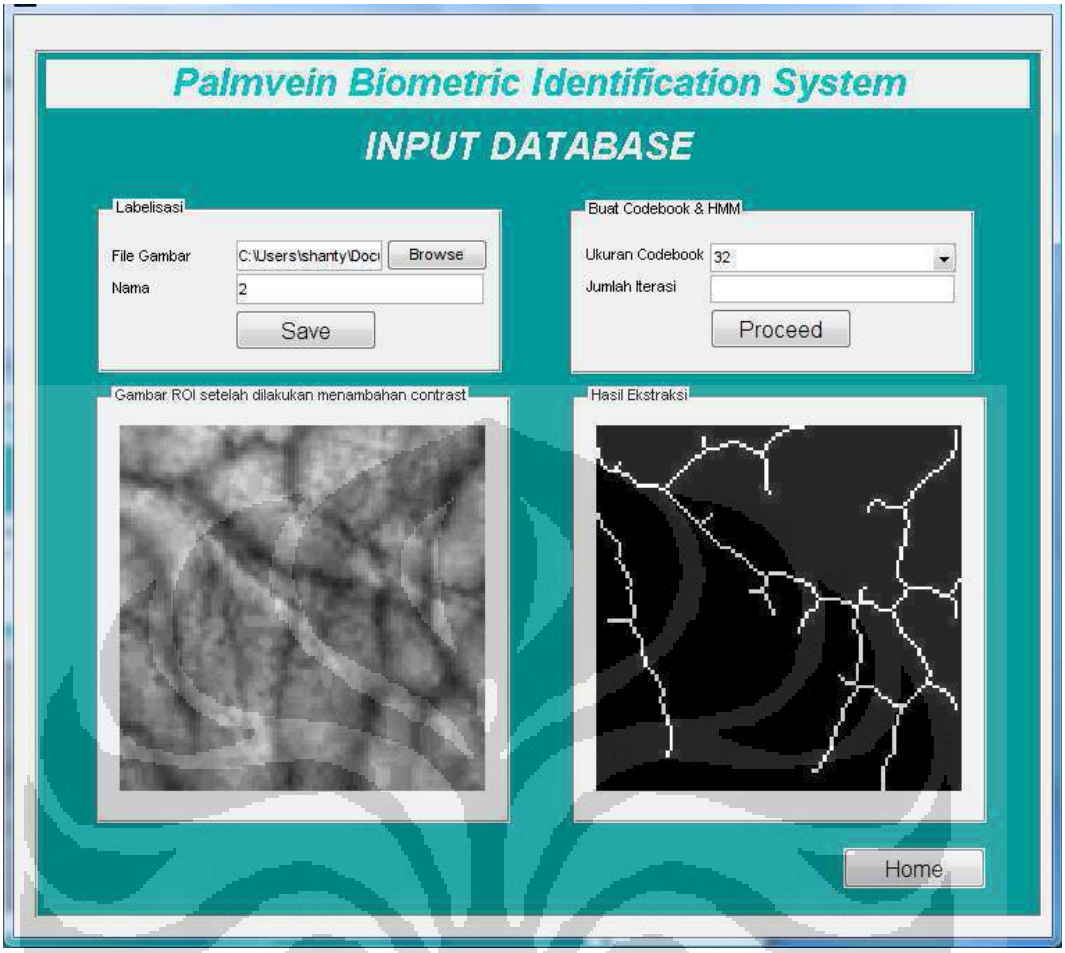
4.1 Uji Coba Sistem

Pada uji coba sistem dalam skripsi ini akan menggunakan data dari 40 (sepuluh) orang yang masing-masingnya memiliki 1 gambar untuk *training* dan 1 gambar untuk proses identifikasi, yang mana setiap gambar diambil pada waktu yang berbeda. Pengujian dilakukan pada 6 (enam) variasi ukuran *codebook*, diantaranya : 32, 64, 128, 256, 512 dan 1024. Dan jumlah iterasi yang diuji adalah 5, 10 dan 15. Setiap variasi ukuran *codebook* akan diterapkan pada setiap variasi jumlah iterasi, diantaranya :

1. Uji coba untuk ukuran *codebook* 32.
2. Uji coba untuk ukuran *codebook* 64.
3. Uji coba untuk ukuran *codebook* 128.
4. Uji coba untuk ukuran *codebook* 256.
5. Uji coba untuk ukuran *codebook* 512.
6. Uji coba untuk ukuran *codebook* 1024.

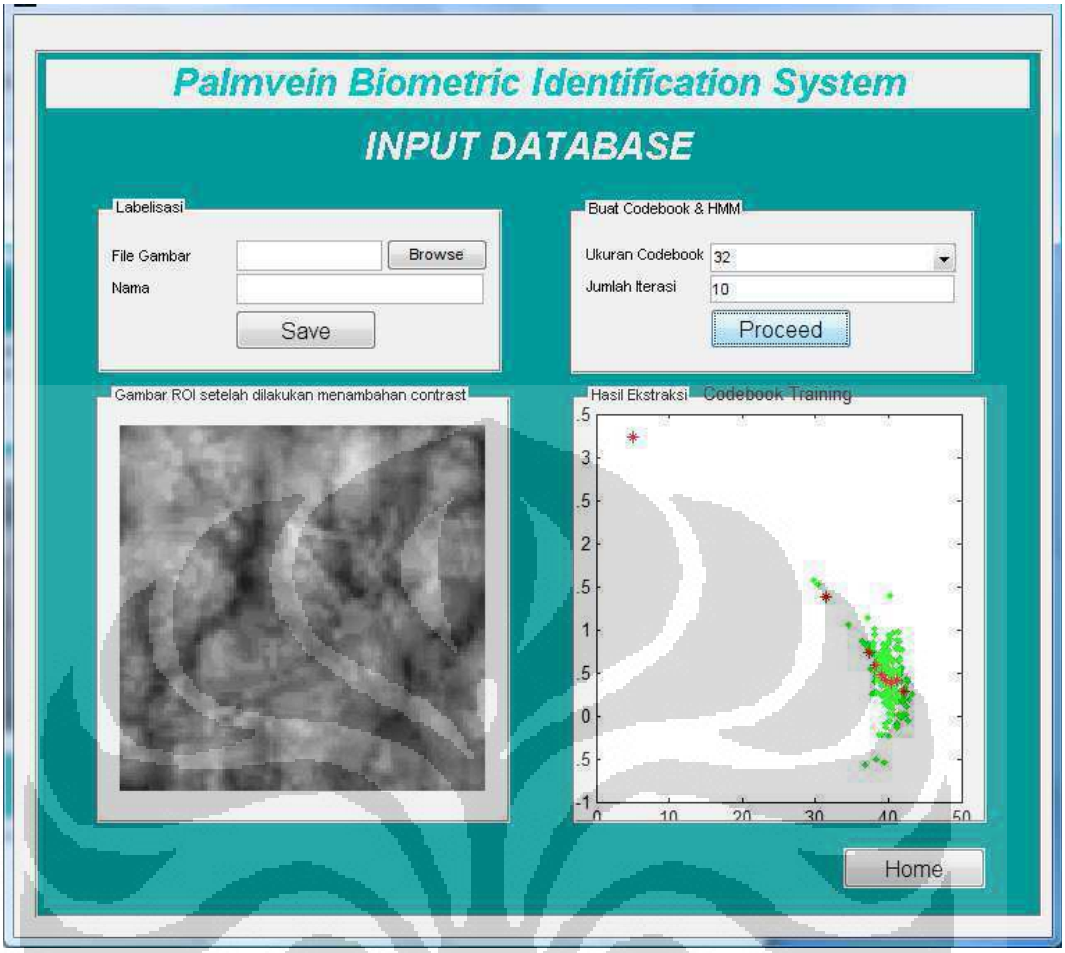
Langkah pengujian untuk proses *training database* dan identifikasi adalah sebagai berikut :

1. Untuk *training* dan *input database*, klik tombol '*Input Database*' pada menu utama GUI. Masukan gambar yang akan di*training* dengan mengklik tombol '*browse*' dan nama pemilik gambar telapak tangan tersebut pada *field* 'Nama'. Klik tombol '*save*' pada bagian bawah, dan kemudian *input*kan lagi gambar dan nama yang akan di*training* selanjutnya. Tahap inilah yang disebut sebagai labelisasi. Tampilan GUI akan terlihat seperti pada Gambar 4.1.

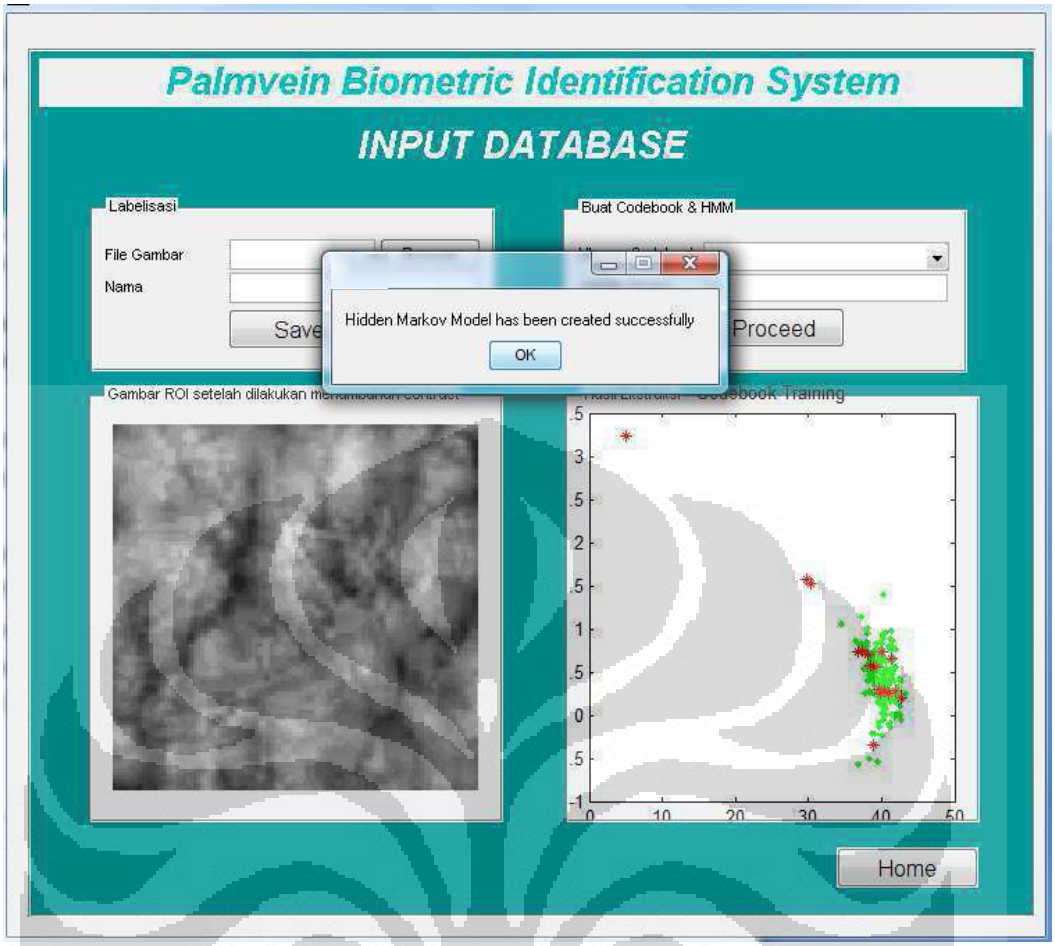


Gambar 4.1 Tampilan GUI setelah menginput gambar untuk *training database*

2. Setelah menginputkan gambar-gambar yang akan di*training*, langkah selanjutnya adalah menghitung *codebook* dan parameter HMM dari gambar-gambar yang telah di*input*kan. Pilih ukuran *codebook* dan jumlah iterasi kemudian klik tombol '*Proceed*'. Apabila sistem sudah selesai menghitung nilai *codebook* dan parameter HMM dari masing-masing data, akan muncul pemberitahuan '*Hidden Markov Model have been created successfully*'. Pada Gambar 4.2 dapat dilihat hasil pembuatan *codebook*.

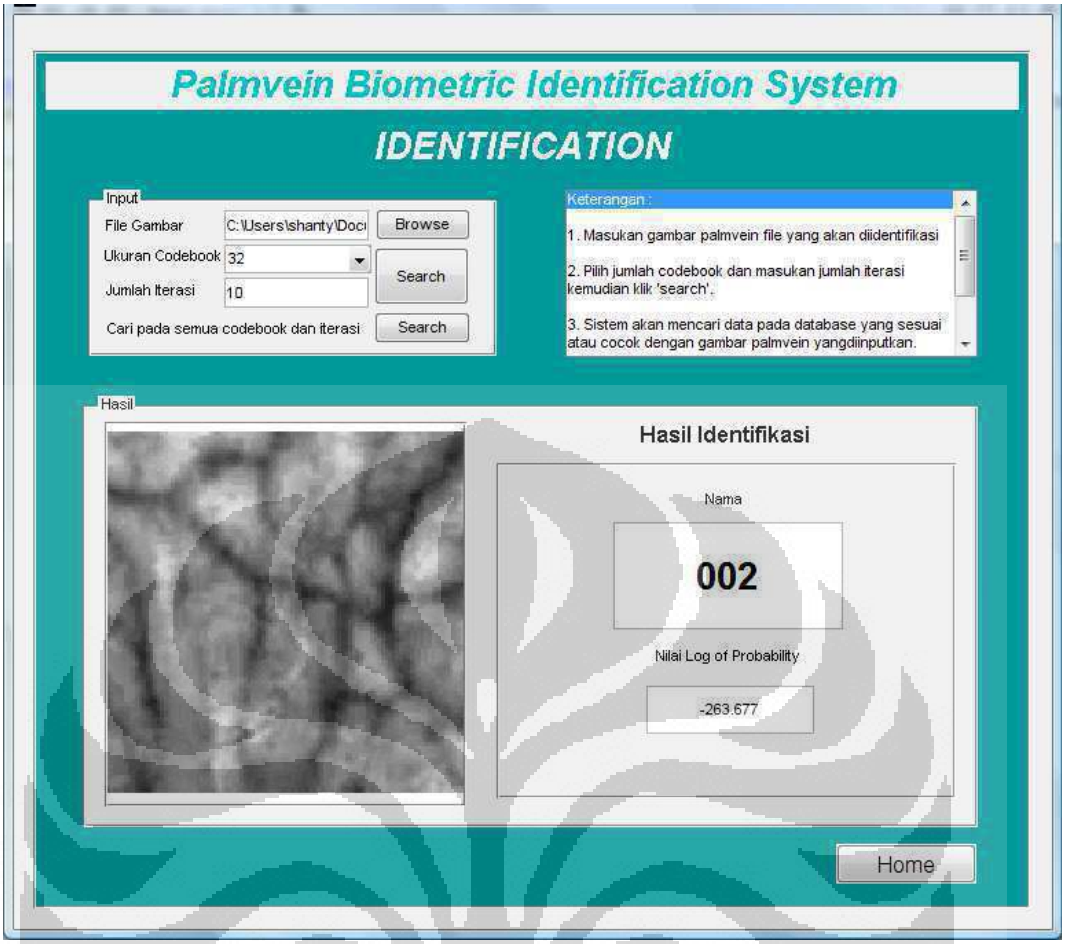


Gambar 4.2 Tampilan GUI saat *training codebook*



Gambar 4.3 Tampilan GUI saat sistem sudah berhasil membuat HMM

3. Tahap selanjutnya yaitu uji coba identifikasi. Gambar yang akan diidentifikasi di*input*kan pada sistem untuk mengetahui apakah sistem bisa mengenali identitas dari pemilik pola pembuluh darah telapak tangan pada gambar tersebut. Klik '*browse*' untuk memilih gambar yang diidentifikasi, disertai dengan meng*input*kan ukuran *codebook* dan jumlah iterasinya. Klik '*search*' untuk memulai proses pengidentifikasian. Sistem akan menghitung nilai *Log of Probability* (LoP) dari masing-masing data pada *database* yang memiliki ukuran *codebook* dan jumlah iterasi yang sama dengan yang diinputkan. Dan kemudian hasil LoP terbesar akan ditampilkan pada *field* hasil pada GUI beserta dengan nama label dari data tersebut.



Gambar 4.4 Tampilan GUI pada saat identifikasi

4.1.1 Hasil Uji Coba Untuk Ukuran *Codebook* 32

Pada Tabel 4.1 dapat dilihat hasil uji coba untuk ukuran *codebook* 32 pada jumlah iterasi 5, 10 dan 15.

Tabel 4.1 Hasil uji coba untuk ukuran *codebook* 32

Nama File	32		
	5	10	15
001 _ 2	001	001	001
002 _ 2	040	020	002
003 _ 2	004	003	003
004 _ 2	011	011	004
005 _ 2	004	005	005
006 _ 2	029	021	027
007 _ 2	037	037	007
008 _ 2	012	008	008

009 _ 2	029	023	007
010 _ 2	010	010	010
011 _ 2	020	011	011
012 _ 2	015	013	013
013 _ 2	020	020	020
014 _ 2	014	012	014
015 _ 2	024	015	015
016 _ 2	029	016	016
017 _ 2	020	031	017
018 _ 2	029	012	018
019 _ 2	029	019	001
020 _ 2	031	031	031
021 _ 2	028	031	028
022 _ 2	002	007	022
023 _ 2	040	034	040
024 _ 2	024	024	015
025 _ 2	025	025	025
026 _ 2	022	008	026
027 _ 2	028	011	028
028 _ 2	028	031	028
029 _ 2	029	029	029
030 _ 2	030	020	030
031 _ 2	025	025	031
032 _ 2	029	012	032
033 _ 2	033	033	035
034 _ 2	004	004	047
035 _ 2	035	035	035
036 _ 2	012	036	036
037 _ 2	012	037	037
038 _ 2	038	038	038
039 _ 2	040	039	039
040 _ 2	040	011	040

4.1.2 Hasil Uji Coba Untuk Ukuran *Codebook* 64

Pada Tabel 4.2 dapat dilihat hasil uji coba untuk ukuran *codebook* 64 pada jumlah iterasi 5, 10 dan 15.

Tabel 4.2 Hasil uji coba untuk ukuran *codebook* 64

Nama File	64		
	5	10	15
001 _ 2	001	019	001

002	2	029	019	002
003	2	003	019	003
004	2	029	004	004
005	2	011	005	005
006	2	027	006	027
007	2	001	001	007
008	2	008	008	008
009	2	008	008	009
010	2	010	019	010
011	2	011	019	011
012	2	018	040	012
013	2	011	019	011
014	2	029	019	014
015	2	008	008	027
016	2	008	021	016
017	2	029	017	035
018	2	008	018	018
019	2	020	019	029
020	2	029	019	020
021	2	028	021	024
022	2	008	008	022
023	2	011	019	027
024	2	024	024	029
025	2	025	025	008
026	2	008	027	026
027	2	027	008	027
028	2	028	019	028
029	2	004	029	029
030	2	030	019	030
031	2	029	031	031
032	2	029	032	032
033	2	033	033	033
034	2	011	011	034
035	2	038	035	035
036	2	036	036	008
037	2	016	037	037
038	2	028	038	038
039	2	022	039	039
040	2	011	011	029

4.1.3 Hasil Uji Coba Untuk Ukuran *Codebook* 128

Pada Tabel 4.3 dapat dilihat hasil uji coba untuk ukuran *codebook* 128 pada jumlah iterasi 5, 10 dan 15.

Tabel 4.3 Hasil uji coba untuk ukuran *codebook* 128

Nama File	128		
	5	10	15
001_2	016	029	001
002_2	002	029	002
003_2	003	019	003
004_2	004	029	004
005_2	030	005	005
006_2	030	006	008
007_2	001	012	007
008_2	008	008	008
009_2	008	009	008
010_2	010	040	010
011_2	027	011	011
012_2	018	012	040
013_2	017	029	027
014_2	027	029	014
015_2	008	015	008
016_2	018	023	016
017_2	027	027	017
018_2	017	024	018
019_2	019	029	019
020_2	030	020	030
021_2	017	021	040
022_2	002	005	022
023_2	023	027	022
024_2	024	001	024
025_2	025	025	008
026_2	027	034	026
027_2	027	027	022
028_2	017	028	028
029_2	020	029	029
030_2	017	019	030
031_2	020	031	031
032_2	019	032	032
033_2	033	033	033
034_2	030	034	034
035_2	035	035	035

036 _ 2	036	036	008
037 _ 2	016	037	037
038 _ 2	028	035	038
039 _ 2	016	023	039
040 _ 2	008	040	001

4.1.4 Hasil Uji Coba Untuk Ukuran *Codebook* 256

Pada Tabel 4.4 dapat dilihat hasil uji coba untuk ukuran *codebook* 256 pada jumlah iterasi 5, 10 dan 15.

Tabel 4.4 Hasil uji coba untuk ukuran *codebook* 256

Nama File	256		
	5	10	15
001 _ 2	029	029	001
002 _ 2	029	002	002
003 _ 2	003	029	003
004 _ 2	004	004	004
005 _ 2	030	005	017
006 _ 2	027	027	006
007 _ 2	007	029	007
008 _ 2	023	008	008
009 _ 2	036	009	039
010 _ 2	005	010	010
011 _ 2	030	048	040
012 _ 2	012	018	034
013 _ 2	011	038	011
014 _ 2	029	029	014
015 _ 2	012	015	015
016 _ 2	029	037	016
017 _ 2	027	010	017
018 _ 2	018	018	017
019 _ 2	019	019	040
020 _ 2	011	011	020
021 _ 2	026	021	017
022 _ 2	011	022	022
023 _ 2	027	023	014
024 _ 2	001	024	024
025 _ 2	025	039	017
026 _ 2	027	027	026
027 _ 2	027	029	011
028 _ 2	029	028	028

029 _ 2	029	029	029
030 _ 2	030	030	030
031 _ 2	029	031	031
032 _ 2	001	032	032
033 _ 2	033	030	033
034 _ 2	027	034	034
035 _ 2	035	035	035
036 _ 2	036	036	036
037 _ 2	037	037	036
038 _ 2	028	025	038
039 _ 2	039	039	039
040 _ 2	027	040	040

4.1.5 Hasil Uji Coba Untuk Ukuran *Codebook* 512

Pada Tabel 4.5 dapat dilihat hasil uji coba untuk ukuran *codebook* 512 pada jumlah iterasi 5, 10 dan 15.

Tabel 4.5 Hasil uji coba untuk ukuran *codebook* 512

Nama File	512		
	5	10	15
001 _ 2	001	029	001
002 _ 2	002	002	002
003 _ 2	003	029	003
004 _ 2	002	004	002
005 _ 2	020	010	005
006 _ 2	006	034	026
007 _ 2	021	007	007
008 _ 2	008	021	008
009 _ 2	040	040	009
010 _ 2	020	005	010
011 _ 2	040	011	011
012 _ 2	012	018	012
013 _ 2	027	038	040
014 _ 2	014	029	014
015 _ 2	039	015	015
016 _ 2	023	006	016
017 _ 2	027	017	005
018 _ 2	002	002	018
019 _ 2	019	005	019
020 _ 2	020	005	020
021 _ 2	040	021	021

022_2	002	026	022
023_2	020	023	023
024_2	004	024	024
025_2	025	025	025
026_2	026	026	027
027_2	027	027	027
028_2	003	028	028
029_2	029	029	002
030_2	040	030	030
031_2	021	031	031
032_2	010	032	032
033_2	033	030	033
034_2	027	034	034
035_2	021	035	035
036_2	036	036	036
037_2	036	037	037
038_2	010	038	025
039_2	039	039	039
040_2	027	040	040

4.1.6 Hasil Uji Coba Untuk Ukuran *Codebook* 1024

Pada Tabel 4.6 dapat dilihat hasil uji coba untuk ukuran *codebook* 1024 pada jumlah iterasi 5, 10 dan 15.

Tabel 4.6 Hasil uji coba untuk ukuran *codebook* 1024

Nama File	1024		
	5	10	15
001_2	003	001	001
002_2	002	002	002
003_2	003	003	003
004_2	002	004	004
005_2	010	005	005
006_2	006	034	026
007_2	019	019	002
008_2	008	023	008
009_2	039	009	009
010_2	040	010	020
011_2	005	011	011
012_2	018	012	012
013_2	040	038	013
014_2	014	014	020

015	2	036	015	015
016	2	021	016	016
017	2	020	017	017
018	2	026	018	018
019	2	019	019	001
020	2	034	020	020
021	2	028	021	021
022	2	022	010	022
023	2	015	023	023
024	2	012	012	024
025	2	025	025	025
026	2	026	026	026
027	2	027	027	027
028	2	028	034	028
029	2	029	002	029
030	2	030	030	030
031	2	020	025	031
032	2	040	032	032
033	2	030	033	033
034	2	034	034	034
035	2	035	035	035
036	2	036	036	036
037	2	037	037	037
038	2	028	028	038
039	2	039	039	039
040	2	040	040	040

4.2 Analisa Sistem

Dari hasil uji coba yang didapat sebelumnya, analisa dapat dilakukan dengan melihat berbedaan hasil antara ukuran *codebook* yang berbeda dan jumlah iterasi yang berbeda pula. Akurasi sistem dapat dihitung dengan membandingkan antara jumlah data yang dapat teridentifikasi dengan tepat dengan 50 jumlah data yang diidentifikasi secara keseluruhan. Berikut adalah kesimpulan akurasi setiap jumlah iterasi dan ukuran *codebook*.

Tabel 4.7 Persentase akurasi dari hasil percobaan

Jumlah Iterasi	Ukuran *Codebook*					
	32	64	128	256	512	1024
5	30,0%	30,0%	32,5%	37,5%	40,0%	47,5%
10	45,0%	47,5%	52,5%	60,0%	60,0%	75,0%
15	70,0%	72,5%	70,0%	70,0%	82,5%	87,5%

4.2.1 Pengaruh Variasi Ukuran *Codebook* Terhadap Tingkat Akurasi

Untuk jumlah iterasi yang sama, peningkatan ukuran *codebook* akan mempengaruhi persentase akurasi sistem. Secara garis besar, semakin besar ukuran *codebook* maka akan semakin besar tingkat akurasi sistem tersebut. Hal ini disebabkan karena ukuran *codebook* yang lebih besar akan membuat jumlah *centroid* semakin banyak. Banyaknya *centroid* ini membuat proses kuantisasi pemilihan nilai vektor data semakin teliti, sehingga pemetaan terahadap vektor data dapat dilakukan dengan jarak yang lebih kecil. Namun dari hasil percobaan terlihat bahwa pengaruh *codebook* tidak terlalu mempengaruhi persentase akurasi sistem. Dimana dengan peningkatan ukuran *codebook* hanya akan menambah satu sampai lima jumlah gambar yang teridentifikasi, dan ada juga yang memiliki persentase akurasi yang sama antara ukuran *codebook* yang satu dengan ukuran *codebook* lainnya, atau bahkan terjadi penurunan persentase akurasi jika menaikkan ukuran *codebook*. Hal ini terjadi karena adanya kemiripan karakteristik pola-pola pembuluh darah yang ada pada *database*, sehingga *centroid-centroid* dari keseluruhan sistem menjadi sangat berdekatan, dan menyebabkan kesulitan dalam proses identifikasi.

Semakin besar ukuran *codebook*, waktu pemrosesan juga akan semakin lama karena akan semakin banyak jumlah *centroid* yang akan dicari. Jadi menerapan ukuran *codebook* yang lebih tinggi lagi akan menyebabkan sistem menjadi tidak efisien, sedangkan hasil yang didapatkan tidak terlalu berbeda. Karena untuk mendapatkan hasil uji coba yang efektif, perlu diperhatikan juga waktu pemrosesan dengan persentase akurasi yang tinggi.

4.2.2 Pengaruh Variasi Jumlah Iterasi Terhadap Tingkat Akurasi

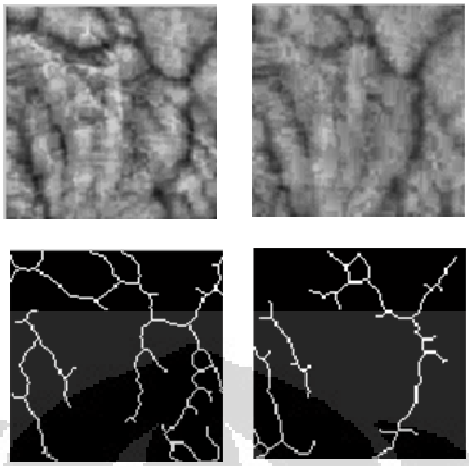
Dari data hasil percobaan, sangat jelas terlihat bahwa variasi jumlah iterasi sangat mempengaruhi tingkat akurasi sistem. Dimana semakin banyak jumlah

iterasi maka persentase tingkat akurasi akan semakin meningkat. Peningkatan akurasi ini jauh melebihi akurasi pada saat peningkatan ukuran *codebook*. Hal ini disebabkan semakin besar jumlah iterasi, maka akan semakin presisi letak *centroid* yang didapat pada saat pembuatan *codebook*, karena jumlah iterasi merupakan banyaknya proses pengulangan yang dilakukan dalam menentukan *centroid* guna medapatkan *centroid* yang cukup presisi.

Sedangkan pada saat pembentukan HMM, iterasi merupakan banyaknya proses pengulangan yang dilakukan dalam mengolah parameter HMM guna mendapatkan probabilitas yang paling baik. Semakin besar jumlah iterasinya, maka akan semakin baik probabilitas yang didapat, namun terdapat suatu nilai iterasi dimana probabilitas yang didapat akan konstan atau relatif sangat kecil perubahannya. Selain itu, sama halnya dengan peningkatan ukuran *codebook*, peningkatan jumlah iterasi juga akan menyebabkan waktu pemrosesan menjadi lebih lama. Jadi, penerapan jumlah iterasi yang terlalu tinggi juga akan membuat sistem menjadi tidak efektif. Oleh karena itu, pada skripsi ini hanya membatasi jumlah iterasi hanya sampai 15 iterasi saja dengan akurasi yang sudah cukup tinggi.

4.3 Faktor Lain Yang Mempengaruhi Tingkat Akurasi

Selain variasi ukuran *codebook* dan jumlah iterasi yang akan mempengaruhi tingkat akurasi, faktor lain yang mempengaruhi tingkat akurasi adalah perbedaan tingkat kualitas dan kontras dari gambar. Perbedaan kualitas dan kontras ini dapat disebabkan dari berbagai faktor, diantaranya : temperatur tubuh, suhu lingkungan, dan kelembaban [12]. Hal ini akan mempengaruhi pola yang dihasilkan dalam ekstraksi gambar. Pada Gambar 4.5 dapat dilihat bahwa dua gambar dari telapak orang tangan yang sama diambil pada waktu yang berbeda. Antara kedua gambar tersebut terdapat perbedaan kontras. Sehingga hasil ekstraksi dari kedua gambar juga akan menghasilkan pola pembuluh darah yang berbeda. Pada gambar dengan kontras yang lebih rendah, terdapat garis yang tidak terdeteksi oleh sistem sehingga garis tersebut akan hilang pada saat proses ekstraksi. Hal ini menyebabkan berkurangnya informasi yang dapat diambil dari gambar dan memungkinkan sistem keliru dalam pengidentifikasian.



Gambar 4.5 Ekstraksi dari dua gambar yang memiliki perbedaan kualitas gambar

BAB 5
KESIMPULAN

Dari hasil uji coba dan analisis yang dilakukan, maka dapat diambil beberapa kesimpulan, yaitu :

1. Nilai akurasi dari keseluruhan uji coba bekisar antara 30% sampai dengan 87,5%.
2. Peningkatan ukuran *codebook* akan meningkatkan persentase akurasi.
3. Penambahkan jumlah iterasi juga akan meningkatkan persentase akurasi.
4. Penambahan jumlah iterasi lebih mempengaruhi persentase akurasi dari pada peningkatan ukuran *codebook*. Dimana penambahan jumlah iterasi akan meningkatkan persentase akurasi mencapai 25%, sedangkan peningkatan jumlah *codebook* hanya akan meningkatkan persentase akurasi sekitar 7%.
5. Untuk mendapatkan perangkat lunak yang efisien, sebaiknya ukuran *codebook* dan jumlah iterasi tidak terlalu besar, karena akan mempengaruhi kinerja sistem menjadi semakin lama. Selain itu, terdapat suatu nilai dimana ukuran *codebook* dan jumlah iterasi tidak terlalu mempengaruhi persentase akurasi dan relatif konstan.
6. Temperatur tubuh, suhu lingkungan, dan kelembaban pada saat gambar diambil akan mempengaruhi kualitas kontras citra masukan sehingga mengurangi tingkat akurasi pengidentifikasian.

DAFTAR ACUAN

[1] Huan Zhang, Dewen Hu. *A Palm Vein Recognition System*. National University of Defense Technology, China, 2010

[2] Wolthusen, Stephen D., Busch, Christoph. *Development of Palm-vein Illumination and sensor system*.

[3] Tan, Ewin. *Fujitsu PalmSecure Business Model*. Fujitsu Indonesia, Jakarta, 2011

[4] Sarkar, I., Alisherov, F., Taihoon Kim. *Palm Vein Authentication System*. International Journal of Control and Automation, 2010

[5] *Palm Vein Pattern Authentication Technology*. Fujitsu Computer Product of America, Inc, 2006

[6] CASIA-MS-PalmprintV1, http://biometrics.idealtest.org/

[7] Siahaan, Meilinda. *Implementasi Segmentasi Citra Menggunakan Metode Graph yang Efisien*. Universitas Sumatera Utara, Medan, 2009

[8] Emy. *Peningkatan Mutu Citra (Image Enhancement) Pada Domain Spatial*. 2007

[9] Sugiarto, Ferry. *Pengenalan Plat Mobil Dengan Skeletonisasi dan Hidden Markov Model*. Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2007

[10] Pribadi, Allpins. *Penerapan Hidden Markov Model Dalam Pengidentifikasian Gelombang Perubahan Fase Dari Gerakan Ikan*. Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2004

[11] Lam, L., Seong-Whan Lee, and Ching Y. Suen, *Thinning Methodologies-A Comprehensive Survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 14, No. 9, September 1992

[12] Wong, K., Lai, T., Bosco Lee, Shum, F. *Analysis of Palm Vein Biometric System*

DAFTAR PUSTAKA

Sugiarto, Ferry. *Pengenalan Plat Mobil Dengan Skeletonisasi dan Hidden Markov Model*. Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2007

Pribadi, Allpins. *Penerapan Hidden Markov Model Dalam Pengidentifikasian Gelombang Perubahan Fase Dari Gerakan Ikan*. Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2004

Saaddatuddaroin. *Perancangan Perangkat Lunak Sendor Tsunami dengan Teknik Hidden Markov Model*. Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2009

Andriani, Evi. *Analisa Dan Identifikasi Berbagai Penyakit Paru-paru Dengan Metode Hidden Markov Model*. Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2009