UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM PENGENALAN PENYAKIT JANTUNG DENGAN METODE *HIDDEN MARKOV MODEL*

SKRIPSI

MUHAMMAD RIZKY HARTAMAN
04 05 03 7103

FAKULTAS TEKNIK
PROGRAM STUDI ELEKTRO
DEPOK
JULI 2009

UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM PENGENALAN PENYAKIT JANTUNG DENGAN METODE *HIDDEN MARKOV MODEL*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

MUHAMMAD RIZKY HARTAMAN
04 05 03 7103

FAKULTAS TEKNIK
PROGRAM STUDI ELEKTRO
DEPOK
JULI 2009

HALAMAN PERNYATAAN ORISINALITAS

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

Nama	**: Muhammad Rizky Hartaman**
NPM	**: 0405037103**
Tanda Tangan	**:**
Tanggal	**: 10 Juli 2009**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama	: Muhammad Rizky Hartaman
NPM	: 0405037103
Program Studi	: Elektro
Judul Skripsi	: Rancang Bangun Sistem Pengenalan Penyakit Jantung dengan 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 Elektro, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Ir. Djamhari Sirat, M.Sc, Ph.D ( )

Penguji 1 : Dr. Ir. Arman Djohan Diponegoro ( )

Penguji 2 : Dr. Abdul Muis ST, M.Eng ( )

Ditetapkan di : Depok

Tanggal : 10 Juli 2009

UCAPAN TERIMA KASIH

Alhamdulillah, segala puji dan syukur penulis ucapkan kehadirat Allah SWT karena dengan rahmat, ridho, dan kasih sayang-Nya, penulisan skripsi ini bisa selesai tepat pada waktunya. Salawat dan salam selalu penulis haturkan kepada baginda Rasulullah Muhammad SAW, karena berkat jasa beliaulah kita dapat hidup di zaman yang terang benderang ini. Penulisan skripsi ini dilakukan dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Elektro Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

1. Ayahanda tersayang, **Pamudjidjana S. Saputro** dan Ibunda tercinta, **Asmarany**, yang telah membesarkan, mendidik dan mendoakan saya sehingga saya menjadi laki-laki yang luar biasa seperti sekarang ini. Ayahanda dan Ibunda merupakan motivasi saya untuk menjadi seorang yang berguna bagi keluarga, nusa, bangsa dan agama serta motivasi saya untuk menyelesaikan skripsi ini. Love you Mom. Love You Dad;
2. Ir. Djamhari Sirat, M.Sc, Ph.D selaku dosen pembimbing resmi yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
3. Dr. Ir. Arman Djohan Diponegoro selaku dosen pembimbing harian yang telah memberi saya tema skripsi ini dan ilmu yang sangat berguna untuk menyusun dan menyelesaikan skripsi ini. Tanpa beliau, saya mungkin tidak akan pernah menyusun skripsi dengan tema ini. Anda memang pantas dijuluki Mr.Markov, Pak!
4. My lovely sista and My Lovely Bro, Oci, Dewi, dan Sigit, yang telah mendoakan dan memotivasi saya selama menyusun skripsi ini.
5. Mario, yang telah banyak membantu memecahkan masalah program bersama.
6. Amin A. Mujadid, atas ide pemrogramannya.

7. Dita yang telah banyak membantu dalam ide penyusunan dan edit-editannya. Makasih banyak ya.
8. Rena Winanti dan Fenny Tania yang telah memberikan sampel data penyakit jantung. Wish u two become a great doctor! ;
9. Bang Suryanegara yang telah memberikan saran-saran dalam penyusunan skripsi ini.
10. Tanoto Foundation, yang telah memberikan beasiswa selama saya kuliah.
11. Teman-teman yang telah banyak mendoakan saya. Kamsamhamnida! Domo Arigatou! Thank You! Jazakillah! Gracias! Sie Sie! Merci!

Depok, 10 Juli 2009

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI

TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Muhammad Rizky Hartaman
NPM : 0405037103
Program Studi : Elektro
Departemen : 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 :

RANCANG BANGUN SISTEM PENGENALAN PENYAKIT JANTUNG DENGAN 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 : 10 Juli 2009
Yang menyatakan

(Muhammad Rizky Hartaman)

ABSTRAK

Nama : Muhammad Rizky Hartaman
Program Studi : Elektro
Judul : Rancang Bangun Sistem Pengenalan Penyakit Jantung dengan Metode *Hidden Markov Model*

Sampai saat ini, serangan jantung masih menjadi penyebab utama kematian dibanyak tempat di dunia. Salah satunya adalah kelainan pada katup jantung yang dapat dideteksi melalui suara *murmur* pada detak jantung penderita. Skripsi ini membahas tentang perancangan sistem pengenalan penyakit jantung berdasarkan suara detak jantung dengan metode HMM. Sistem ini terbagi menjadi dua proses utama, yaitu pembentukan database dan pengenalan penyakit jantung. Kedua proses ini dilakukan dengan cara yang hampir sama, yaitu tiap sampel akan mengalami proses pelabelan, pembuatan codebook dan pembentukan parameter HMM. Hanya saja, pengolahan sinyal suara pada proses pengenalan mengacu database yang telah lebih dulu diproses. Dimulai dengan pembentukan vektor-vektor data dengan teknik kuantisasi vektor (VQ), yang kemudian dicari suatu nilai *centroid* yang presisi untuk dijadikan state HMM dalam menentukan nilai-nilai parameter yang dibutuhkan. Berdasarkan parameter-parameter inilah, dapat dihitung suatu nilai probabilitas (*Log of Probability*) maksimum yang akan menunjukkan hasil keluarannya. Dari hasil perancangan sistem ini, akan dibandingkan akurasi sistem terhadap variasi nilai durasi sampel, jumlah sampel, dan ukuran *codebook.*

Pada penelitian ini ukuran *codebook* yang optimal adalah 64, jumlah database yang optimal sebesar 10 (sepuluh) buah, dan rentang waktu sampel yang optimal adalah 0,7 detik. Sementara akurasi sistem secara keseluruhan bervariasi antara 60% hingga 85%.

Kata kunci:
Murmur, penyakit jantung, HMM, centroid, VQ, LoP, ukuran *codebook*, durasi sampel, jumlah sampel.

ABSTRACT

Name : Muhammad Rizky Hartaman
Study Program : Electrical Engineering
Title : Design of Heart Disease Recognition System Using Hidden Markov Model

Heart attack is still being the number one killer until now all over the world. A part of heart diseases which can be detected by *murmur* sound and will be explained here is valve anomaly. This thesis is talking about heart disease recognition based on its heart sound system design using HMM method. The system consists of two main processes: database construction and diseases recognition. Both of this processes is done with almost exact ways. Each samples will be processed through labelling, codebook construction, and HMM parameter making. The difference is that in recognizing process, sound signal will be compared to database which has been made before. The whole process is started with data vectors production by vector quantization (VQ) which can be used to analyze precise centroid positions. The centroid will define HMM states and parameters. A Log of Probability (LoP) will be calculated from the parameter values. The largest value of LoP will be declared as an output of the system. Output of each samples are compared to get system accuracy based on variation of sample duration, sample amount, and codebook size.

The optimum codebook size in this research is 64, optimum sample amount in database is 10, and 0.7s sample duration. Overall, accuracy of the system is variating from 60% up to 85%.

Key words:
Murmur, heart disease, HMM, centroid, VQ, LoP, codebook size, sample duration, sample amount.

DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS iii

HALAMAN PENGESAHAN iv

UCAPAN TERIMA KASIH v

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS vii

ABSTRAK viii

ABSTRACT ix

DAFTAR ISI x

DAFTAR TABEL xii

DAFTAR GAMBAR xiii

DAFTAR LAMPIRAN xiv

DAFTAR SINGKATAN xv

BAB I PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Tujuan Penelitian 2

1.3 Batasan Masalah 2

1.4 Metode Penulisan 2

1.5 Sistematika Penulisan 3

BAB II DASAR TEORI 4

2.1 Anatomi Jantung 4

2.1.1 Struktur Internal Jantung 4

2.1.2 Cara Kerja Jantung 5

2.2 Penyakit Jantung 6

2.2.1 Regurgitasi Katup Mitral 6

2.2.2 Regurgitasi Katup Aorta 8

2.2.3 Stenosis Katup Mitral 9

2.2.4 Stenosis Katup Aorta 11

2.3 Isyarat Suara Jantung 12

2.4 Proses Pengolahan Sinyal Suara 15

2.4.1 *Sampling* 16

2.4.2 Ekstraksi 16

2.4.3 Kuantisasi Vektor 23

2.5 Hidden Markov Model 25

BAB III RANCANG BANGUN PENELITIAN 29

3.1 Pembuatan *Database* 29
3.1.1 Pelabelan 29
3.1.2 Pembuatan *Codebook* 32
3.1.3 Pembentukan Parameter HMM 35

3.2 Proses Pengenalan 36

BAB IV UJI COBA DAN ANALISIS PERANGKAT LUNAK PENGENALAN PENYAKIT JANTUNG DENGAN METODE HMM 40

4.1 Daftar Penyakit dan Jenis Percobaan 40

4.2 Hasil Uji Coba 41

4.3 Persentase Akurasi 44

4.4 Analisis Hasil Percobaan 45
4.4.1 Analisis Pengaruh Variasi Durasi Sampel 46
4.4.2 Analisis Pengaruh Variasi Ukuran *Codebook* 48
4.4.3 Analisis Pengaruh Variasi Jumlah *Database* 51
4.4.4 Analisis Pengaruh Hubungan Ukuran *Codebook*, Jumlah *Database* dan Durasi Sampel 52

BAB V KESIMPULAN 53

DAFTAR ACUAN 54

DAFTAR PUSTAKA 56

LAMPIRAN 57

DAFTAR TABEL

Tabel 2.1 Jenis-jenis Jantung Abnormal 15
Tabel 4.1 Nama File dan Jenis Penyakit Jantung..... 40
Tabel 4.2 Hasil uji coba penyakit *Aortic Regurgitation* untuk durasi 1,5s..... 41
Tabel 4.3 Hasil uji coba penyakit *Aortic Stenosis* untuk durasi 1,5s..... 42
Tabel 4.4 Hasil uji coba penyakit *Mitral Regurgitation* untuk durasi 1,5s..... 42
Tabel 4.5 Hasil uji coba penyakit *Mitral Stenosis* untuk durasi 1,5s..... 42
Tabel 4.6 Hasil uji coba penyakit *Aortic Regurgitation* untuk durasi 0,7s..... 43
Tabel 4.7 Hasil uji coba penyakit *Aortic Stenosis* untuk durasi 0,7s..... 43
Tabel 4.8 Hasil uji coba penyakit *Mitral Regurgitation* untuk durasi 0,7s..... 43
Tabel 4.9 Hasil uji coba penyakit *Mitral Stenosis* untuk durasi 0,7s..... 44
Tabel 4.10 Persentase akurasi semua penyakit untuk durasi 1,5s..... 44
Tabel 4.11 Persentase akurasi semua parameter untuk durasi 1,5s..... 44
Tabel 4.12 Persentase akurasi semua penyakit untuk durasi 0,7s..... 45
Tabel 4.13 Persentase akurasi semua parameter untuk durasi 0,7s..... 45

DAFTAR GAMBAR

Gambar 2.1 Anatomi jantung manusia 4
Gambar 2.2 Regurgitasi Katup Aorta 8
Gambar 2.3 Stenosis Katup Mitral..... 9
Gambar 2.4 Stenosis katup aorta..... 11
Gambar 2.5 Contoh bentuk gelombang suara regurgitasi..... 13
Gambar 2.6 Contoh bentuk gelombang suara stenosis 14
Gambar 2.7 Blok Diagram MFCC [4] 17
Gambar 2.8 Proses *frame blocking* dan *windowing*..... 18
Gambar 2.9 Windowing pada spektrum frekuensi..... 21
Gambar 2.10 Grafik mel-frekuensi versus frekuensi 22
Gambar 2.11 Contoh *codeword* pada ruang dua dimensi [13] 23
Gambar 2.12 Diagram konsep pembentukan *codebook* dengan *vector quantization*..... 25
Gambar 2.13 Model Markov (a) *Ergodic* (b) *Left-right* [13]..... 26
Gambar 3.1 Diagram alir pembuatan database jantung 30
Gambar 3.2 Tampilan Program Pembuatan Label..... 31
Gambar 3.3 Keluaran proses pelabelan..... 32
Gambar 3.4 Tampilan program pembuatan *codebook*..... 33
Gambar 3.5 Tampilan *software* pembentukan parameter HMM 36
Gambar 3.6 Diagram alir proses pengenalan penyakit jantung 37
Gambar 3.7 Tampilan program pengenalan penyakit jantung..... 38
Gambar 4.1 Gelombang suara *Aortic Stenosis* dengan durasi 0,7 detik 46
Gambar 4.2 Gelombang suara *Aortic Stenosis* dengan durasi 1,5 detik 46
Gambar 4.3 *Codebook* 32 bit 48
Gambar 4.4 *Codebook 6*4 bit 49
Gambar 4.5 *Codebook* 128 bit 49

DAFTAR LAMPIRAN

1. Listing Program Pelabelan 57
2. Listing Program Pembuatan Codebook 61
3. Listing Program Pembuatan Parameter HMM..... 68
4. Listing Program Proses Pengenalan..... 78

DAFTAR SINGKATAN

AR	Aortic Regurgitation
AS	Aortic Stenosis
AV	Atrio Ventrikularis
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
HMM	Hidden Markov Model
LoP	Log of Probability
MFCC	Mel Frequency Cepstrum Coefficient
MR	Mitral Regurgitation
MS	Mitral Stenosis
VQ	Vector Quantization

BAB I

PENDAHULUAN

1.1 Latar Belakang

Jantung merupakan salah satu organ yang paling penting dalam tubuh manusia. Jantung merupakan pusat sirkulasi darah manusia karena jantung memiliki fungsi utama yaitu mengalirkan darah ke seluruh tubuh dan paru-paru agar tubuh manusia dapat berfungsi sebagaimana mestinya. Apabila jantung mengalami kerusakan sekecil apapun pada salah satu komponen-komponen penyusunnya, maka tubuh manusia juga akan terkena dampaknya.

Kegagalan fungsi jantung dapat mengakibatkan berbagai macam penyakit jantung, yang merupakan penyebab kematian terbesar pada manusia. Penyakit ini tidak lepas dari gaya hidup yang kurang sehat yang banyak dilakukan seiring dengan berubahnya pola hidup. Faktor-faktor pemicu serangan jantung ialah merokok, seringnya mengonsumsi makanan berkolestrol tinggi, kurang gerak, malas berolahraga, stres, dan kurang istirahat.

Kelainan pada irama detak jantung (*murmur*) juga dapat menunjukkan adanya penyakit jantung. *Murmur* ini memiliki siklus yang berulang dimana masing-masing siklus memiliki rentang waktu yang disebut dengan *durasi sampel*. Jenis penyakit jantung yang paling sering menyerang manusia karena kelainan tersebut adalah kelainan pada katup jantung dan kelainan bawaan sejak lahir (pada sekat jantung). Oleh karena itulah, pada skripsi ini akan dibahas bagaimana mengenali penyakit jantung melalui irama atau suara detak jantung. Pada umumnya, terdapat beberapa metode pengenalan suara yang biasa digunakan seperti *Hidden Markov Model* (HMM), *neural network*, dan *fuzzy logic*.

Teknik *fuzzy logic* merupakan teknik yang paling sederhana, namun hasil yang didapatkan kurang akurat dibanding metode lainnya. Sedangkan dengan menggunakan teknik *neural network* diperlukan proses pembelajaran dan iterasi yang sangat banyak dan panjang. Dengan teknik HMM, walaupun proses pembelajarannya lebih kompleks dibanding dengan dua metode lainnya, tetapi jumlah sampel yang dibutuhkan tidak terlalu banyak dan hasil yang diperoleh jauh

lebih optimal. Oleh karena itulah, dalam penelitian ini digunakan HMM sebagai metode pengenalan.

Penelitian ini diharapkan dapat mempermudah para ahli di bidang kedokteran dalam mendeteksi penyakit jantung dengan menggunakan suara detak jantung sebagai parameter pengamatan. Bahkan penelitian ini dapat dikembangkan lebih lanjut dalam pembuatan alat pendeteksi penyakit jantung yang lebih sederhana dan ekonomis sehingga dapat diaplikasikan langsung oleh masyarakat awam sekalipun.

1.2 Tujuan Penelitian

Tujuan dari penulisan skripsi ini adalah merancang perangkat lunak pendeteksi penyakit jantung dengan menggunakan metode HMM. Selain itu skripsi ini ditujukan untuk membandingkan akurasi perangkat lunak sistem pengenalan penyakit jantung terhadap variasi nilai durasi sampel, jumlah sampel, dan ukuran *codebook*.

1.3 Batasan Masalah

Sistem ini merupakan sistem *non-real time*, dimana data yang akan dikenali diolah terlebih dahulu dengan menggunakan *software* pengolah gelombang suara yang kemudian diproses untuk diidentifikasi. Pada penelitian ini, penyakit yang akan dideteksi hanya dibatasi pada penyakit jantung orang dewasa yang secara langsung dapat dikenali melalui suara denyutnya. Adapun penyakit yang dimaksud adalah penyakit dengan kelainan pada katup jantung, yaitu *Aortic Regurgitation* (AR), *Aortic Stenosis* (AS), *Mitral Regurgitation* (MR), dan *Mitral Stenosis* (MS). Masing-masing penyakit terdiri dari 10 (sepuluh) buah *database* dan 10 buah sampel uji yang diambil secara tidak langsung dalam format digital. Penelitian ini menggunakan durasi sampel 0,7 detik dan 1,5 detik. Sedangkan variasi ukuran *codebook* yang digunakan adalah 32, 64, dan 128.

1.4 Metode Penulisan

Metode-metode yang digunakan dalam penulisan skripsi ini antara lain:

(1) Studi kepustakaan

Mempelajari semua informasi mengenai jantung dan kelainannya termasuk kebocoran dan penyumbatan katup serta mengenai proses pengenalan suara dan HMM dari buku, jurnal, artikel dan literatur lain.

(2) Pencarian data

Mencari data-data yang diperlukan, dalam hal ini berupa sampel suara denyut jantung dalam bentuk digital yang diperoleh dari institusi-institusi terkait.

1.5 Sistematika Penulisan

Skripsi ini terdiri dari 5 (lima) bab.

BAB 1 merupakan pendahuluan. Bagian ini berisi tentang latar belakang, tujuan, batasan masalah dan sistematika penulisan untuk memberi gambaran tentang isi skripsi.

BAB 2 merupakan dasar teori. Bagian ini berisi tentang dasar-dasar teori yang mendukung penelitian dalam skripsi ini.

BAB 3 merupakan rancang bangun perangkat lunak. Bagian ini berisi diagram alir dan algoritma proses pembentukan database dan proses pengenalan penyakit.

BAB 4 merupakan hasil uji coba perangkat lunak. Bagian ini berisi hasil pengujian rancangan yang telah dibuat berdasarkan variasi parameter-parameter tertentu.

BAB 5 merupakan penutup. Bagian ini berisi kesimpulan yang diperoleh setelah dilakukan penelitian.

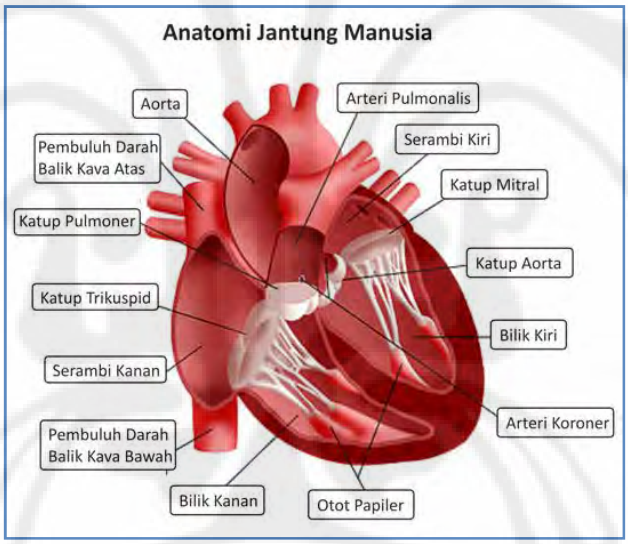
BAB II

DASAR TEORI

2.1 Anatomi Jantung [5]

Jantung, dalam terminologi sederhana, merupakan sebuah pompa yang terbuat dari otot. Istilah *kardiak* berarti berhubungan dengan jantung, dari bahasa Yunani *cardia* untuk jantung. Jantung merupakan salah satu organ terpenting dalam tubuh manusia yang berperan dalam sistem peredaran darah yang berfungsi untuk memompa darah ke paru-paru dan ke seluruh bagian tubuh dan terletak di rongga dada di antara kedua paru-paru.

2.1.1 Struktur Internal Jantung



Gambar 2.1 Anatomi jantung manusia

Jantung terbagi atas empat ruang utama, yaitu Atrium atau Serambi kiri-kanan dan Ventrikel atau Bilik kiri-kanan. Secara fungsional, jantung dibagi menjadi alat pompa kanan, yang memompa darah kotor menuju paru-paru melalui sirkulasi pulmonari, dan alat pompa kiri, yang memompa darah bersih ke seluruh tubuh manusia melalui sirkulasi sistemik. Dinding serambi jauh lebih tipis dibandingkan dinding bilik karena bilik harus melawan gaya gravitasi bumi untuk memompa dari bawah ke atas, khususnya di aorta, untuk memompa ke seluruh bagian tubuh yang memiliki pembuluh darah. Dua pasang rongga (bilik dan

serambi bersamaan) di masing-masing belahan jantung disambungkan oleh sebuah katup.

Secara umum jantung memiliki dua katup jenis utama, yaitu katup *atrioventrikularis* (katup AV) yang memisahkan atrium dengan ventrikel, dan katup *semilunaris* yang memisahkan ventrikel dengan pembuluh darah yang bersangkutan.

Pada bagian jantung kiri, katup AV dikenal dengan nama katup *mitral*, yang memisahkan atrium dan ventrikel kiri. Sedangkan untuk katup semilunarisnya, dikenal dengan nama katup *aorta*. Katup aorta ini memisahkan antara ventrikel kiri dengan aorta. Aorta merupakan pembuluh arteri terbesar pada sirkulasi sistemik, yang menghubungkan pembuluh arteri lain dengan jantung melalui ventrikel kiri.

Pada bagian jantung kanan, katup AV dikenal dengan nama katup *trikuspid*, yang memisahkan atrium dan ventrikel kanan. Sedangkan untuk katup semilunarisnya, dikenal dengan nama katup pulmonalis. Katup pulmonalis ini memisahkan antara ventrikel kanan dengan arteri pulmonalis. Dengan adanya katup AV, darah tidak akan mengalir kembali atrium ketika ventrikel berkontraksi, dan begitu juga dengan adanya katup semilunaris, darah dari aorta maupun arteri pulmonalis tidak akan kembali ke ventrikel sewaktu ventrikel dalam keadaaan istirahat (relaksasi).

2.1.2 Cara Kerja Jantung [12]

Pada saat berdenyut, setiap ruang jantung mengendur dan terisi darah (disebut diastol). Selanjutnya jantung berkontraksi dan memompa darah keluar dari ruang jantung yang disebut sistol. Kedua serambi mengendur dan berkontraksi secara bersamaan, begitu pula kedua bilik juga mengendur dan berkontraksi secara bersamaan.

Darah yang kehabisan oksigen (darah kotor) dan mengandung banyak karbondioksida dari seluruh tubuh mengalir melalui dua vena terbesar (vena kava)

menuju ke dalam serambi kanan. Setelah atrium kanan terisi darah, dia akan mendorong darah ke dalam bilik kanan. Dari bilik kanan, darah akan dipompa melalui katup pulmoner ke dalam arteri pulmonalis menuju ke paru-paru. Darah akan mengalir melalui pembuluh yang sangat kecil (kapiler) yang mengelilingi kantong udara di paru-paru, menyerap oksigen dan melepaskan karbondioksida yang selanjutnya dihembuskan.

Darah yang kaya akan oksigen (darah bersih) mengalir di dalam vena pulmonalis menuju ke serambi kiri. Peredaran darah di antara bagian kanan jantung, paru-paru dan atrium kiri disebut sirkulasi pulmoner. Di dalam serambi kiri darah akan didorong menuju bilik kiri, yang selanjutnya akan memompa darah bersih ini melewati katup aorta masuk ke dalam aorta (arteri terbesar dalam tubuh). Darah kaya oksigen ini disediakan untuk seluruh tubuh, kecuali paru-paru.

2.2 Penyakit Jantung

Serangan jantung masih menempati urutan pertama penyebab kematian dibanyak tempat di dunia. Salah satu hal yang dapat mengakibatkan terjadinya serangan jantung adalah adanya gejala abnormalitas pada bagian-bagian jantung yang dapat membuat jantung tidak dapat berfungsi dengan baik. Beberapa jenis kelainan yang terdapat jantung antara lain berupa kebocoran dan penyempitan pada katup.

2.2.1 Regurgitasi Katup Mitral [15]

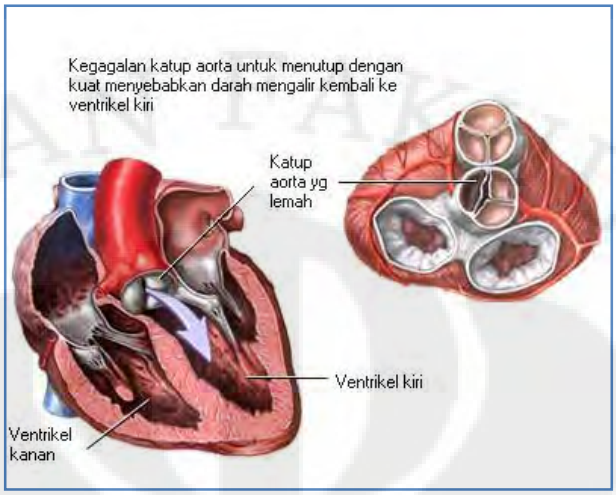
Regurgitasi Katup Mitral (*Inkompetensia Mitral, Insufisiensi Mitral*) adalah kebocoran aliran balik melalui katup mitral setiap kali ventrikel kiri berkontraksi. Pada saat ventrikel kiri memompa darah dari jantung menuju ke aorta, sebagian darah mengalir kembali ke dalam atrium kiri dan menyebabkan meningkatnya volume dan tekanan di atrium kiri. Terjadi peningkatan tekanan darah di dalam pembuluh yang berasal dari paru-paru, yang mengakibatkan penimbunan cairan (kongesti di dalam paru-paru).

Dahulu, demam rematik menjadi penyebab utama dari regurgitasi katup mitral. Tetapi saat ini, di negara-negara yang memiliki obat-obat pencegahan yang baik, demam rematik jarang terjadi. Misalnya di Amerika Utara dan Eropa Barat, penggunaan antibiotik untuk *strep throat* (infeksi tenggorokan karena streptokokus), bisa mencegah timbulnya demam rematik. Penyebab umum lainnya adalah *degenerasi miksomatous* (suatu keadaan dimana katup secara bertahap menjadi terkulai/terkelepai).

Regurgitasi katup mitral yang ringan bisa jadi tidak menunjukkan gejala apapun. Kelainannya bisa dikenali hanya jika dokter melakukan pemeriksaan dengan *stetoskop*, dimana terdengar *murmur* yang khas, yang disebabkan pengaliran kembali darah ke dalam atrium kiri ketika ventrikel kanan berkontraksi. Regurgitasi yang berat akan menyebabkan berkurangnya aliran darah sehingga terjadi *gagal jantung*, yang akan menyebabkan batuk, sesak nafas pada saat melakukan aktivitas dan pembengkakan tungkai.

Pada regurgitasi mitral, darah mengalir balik melalui katup mitral ke dalam atrium kiri selama fase sistol. Keadaan ini juga menimbulkan suara "seperti tiupan" berfrekuensi tinggi dan mendesis yang serupa dengan regurgitasi katup aorta, dan terutama dihantarkan keras ke atrium kiri. Namun atrium kiri terletak dalam sekali di rongga dada sehingga sukar sekali untuk mendengar suara ini tepat diatas atrium. Akibatnya, suara pada regurgitasi mitral dihantarkan ke dinding dada terutama melalui ventrikel kiri, dan biasanya terdengar paling baik di apek jantung. [10]

2.2.2 Regurgitasi Katup Aorta [6]



Gambar 2.2 Regurgitasi Katup Aorta

Regurgitasi Katup Aorta (*Aortic Regurgitation*) adalah kebocoran pada katup aorta yang terjadi setiap kali ventrikel mengalami relaksasi. Sebelum penggunaan antibiotik meluas, penyebab utama dari regurgitasi katup aorta di beberapa belahan dunia adalah demam rematik dan sifilis. Sama sepedrti pada regurgitasi katup mitral, demam rematik juga dahulu menjadi penyebab utama penyakit ini. Selain demam rematik, penyebab lainnya yang paling sering ditemukan adalah:

1. Melemahnya katup.
2. Bahan fibrosa akibat *degenerasi miksomatous*.
   Degenerasi miksomatou merupakan kelainan jaringan ikat yang diturunkan, yang memperlemah jaringan katup jantung dan membuatnya meregang secara tidak normal dan kadang sobek.
3. Kelainan bawaan
4. Infeksi bakteri
5. Cedera

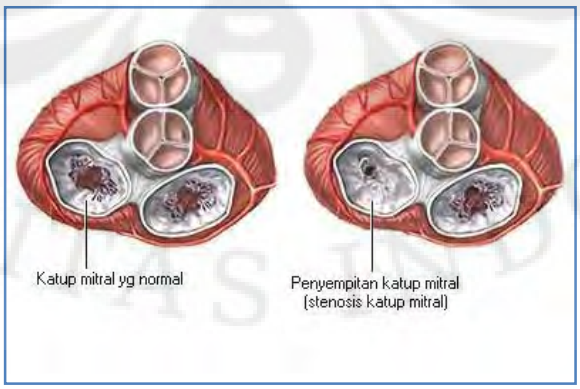
Regurgitasi katup aorta yang ringan tidak menimbulkan gejala selain *murmur* jantung yang khas yang timbul setiap kali ventrikel kiri mengalami relaksasi yang dapat didengar melalui *stetoskop*. Pada regurgitasi yang berat,

ventrikel kiri mengalirkan sejumlah besar darah, yang menyebabkan pembesaran ventrikel dan akhirnya menjadi *gagal jantung*.

Pada regurgitasi Aorta tidak terdengar selama fase sistol, tetapi selama fase diastole, darah mengalir balik dari aorta ke ventrikel kiri, menimbulkan murmur seperti "suara meniup". Yang relatif bernada tinggi dan mendesis, serta terdengar secara maksimal diatas ventrikel kiri. Murmur ini disebabkan oleh turbulen darah yang menyembur balik dengan darah yang telah berada dalam ventrikel kiri. [10]

2.2.3 Stenosis Katup Mitral [8]

Stenosis Katup Mitral (*mitral stenosis*) merupakan penyempitan pada lubang *katup mitral* yang akan menyebabkan meningkatnya tahanan aliran darah dari *atrium* kiri ke *ventrikel* kiri. Pasien dengan Stenosis Katup Mitral (SKM) secara khas memiliki daun katup mitral yang menebal, kommisura yang menyatu, dan korda tendineae yang menebal dan memendek. Diameter transversal jantung biasanya dalam batas normal, tetapi kalsifikasi dari katup mitral dan pembesaran atrium kiri dapat terlihat. Kondisi ini membuat tekanan vena pulmonal meningkat sehingga menyebabkan diversi darah, pada foto toraks terlihat pelebaran relatif pembuluh darah bagian atas paru dibanding pembuluh darah bawah paru. Penyempitan katup mitral menyebabkan katup tidak terbuka dengan tepat dan menghambat aliran darah antara ruang-ruang jantung kiri. Ketika katup mitral menyempit (stenosis), darah tidak dapat dengan efisien melewati jantung. Kondisi ini menyebabkan seseorang menjadi lemah dan nafas menjadi pendek serta gejala lainnya.



Gambar 2.3 Stenosis Katup Mitral

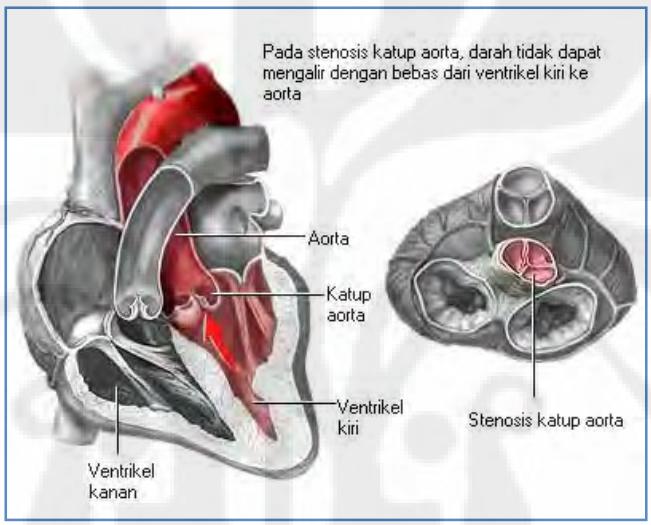
Stenosis katup mitral hampir selalu disebabkan oleh *demam rematik*, yang pada saat ini sudah jarang ditemukan di Amerika Utara dan Eropa Barat. Karena itu di wilayah tersebut, stenosis katup mitral terjadi terutama pada orang tua yang pernah menderita demam rematik pada masa kanak-kanak dan mereka tidak mendapatkan antibiotik. Di bagian dunia lainnya, demam rematik sering terjadi dan menyebabkan stenosis katup mitral pada dewasa, remaja dan kadang pada anak-anak. Yang khas adalah jika penyebabnya demam rematik, daun katup mitral sebagian bergabung menjadi satu. Stenosis katup mitral juga bisa merupakan suatu kelainan bawaan. Bayi yang lahir dengan kelainan ini jarang bisa bertahan hidup lebih dari 2 tahun, kecuali jika telah menjalani pembedahan. *Miksoma* (tumor jinak di atrium kiri) atau bekuan darah dapat menyumbat aliran darah ketika melewati katup mitral dan menyebabkan efek yang sama seperti stenosis katup mitral.

Pada stenosis katup mitral, darah mengalir susah payah melalui katup mitral yang mengalami stenosis dari atrium kiri ke ventrikel kiri, dank arena tekanan dalam atrium kiri jarang meningkat diatas 30 mmHg kecuali untuk jangka waktu pendek, selisih dari tekanan yang besar yang mendorong darah dari atrium kiri ke ventrikel kiri tidak pernah terjadi. Akibatnya bunyi abnormal yang terdengar pada stenosis katup mitral biasanya lemah dan dengan frekuensi sangat rendah sehingga sebagian besar spektrum suara berada di bawah frekuensi terendah dari pendengaran manusia.

Selama bagian awal diastol, ventrikel mengandung sedikit sekali darah dan dindingnya demikian lunak sehingga darah tidak memantul bolak balik diantara dinding-dinding ventrikel. Karena alas an ini, bahkan pada stenosis katup mitral yang hebat sekalipun, sama sekali tidak terdengar murmur selama sepertiga awal diastole. Kemudian, setelah sepertiga awal diastole berlalu ventrikel sudah cukup teregang sehingga darah dipantulkan bolak-balik, dan seringkali mulai terjadi murmur yang bergemuruh rendah. Pada stenosis ringan, murmur hanya berlangsung selama separuh pertama pada bagian kedua dari ketiga bagian diastole, tetapi pada stenosis berat, murmur bias lebih awal dan menetap selama sisa diastole.

2.2.4 Stenosis Katup Aorta [7]

Stenosis Katup Aorta (*Aortic Stenosis*) adalah penyempitan pada lubang *katup aorta*, yang menyebabkan meningkatnya tahanan terhadap aliran darah dari *ventrikel* kiri ke aorta. Di Amerika Utara dan Eropa Barat, stenosis katup aorta merupakan penyakit utama pada orang tua, yang merupakan akibat dari pembentukan jaringan parut dan penimbunan kalsium di dalam daun katup. Stenosis katup aorta seperti ini timbul setelah usia 60 tahun, tetapi biasanya gejalanya baru muncul setelah usia 70-80 tahun.



Gambar 2.4 Stenosis katup aorta

Stenosis katup aorta juga bisa disebabkan oleh *demam rematik* pada masa kanak-kanak. Pada keadaan ini biasanya disertai dengan kelainan pada katup *mitral* baik berupa stenosis, *regurgitasi* maupun keduanya. Pada masa bayi, katup aorta yang menyempit mungkin tidak menyebabkan masalah, masalah baru muncul pada masa pertumbuhan anak. Ukuran katup tidak berubah, sementara jantung melebar dan mencoba untuk memompa sejumlah besar darah melalui katup yang kecil. Katup mungkin hanya memiliki dua daun yang seharusnya tiga, atau memiliki bentuk abnormal seperti corong. Sehingga pembukaan katup tersebut, sering menjadi kaku dan menyempit karena terkumpulnya endapan kalsium.

Dinding ventrikel kiri menebal karena ventrikel berusaha memompa sejumlah darah melalui katup aorta yang sempit. Otot jantung yang membesar membutuhkan lebih banyak darah dari *arteri koroner*. Persediaan darah yang tidak mencukupi akhirnya akan menyebabkan terjadinya nyeri dada (*angina*) pada waktu penderita melakukan aktivitas. Penderita stenosis katup aorta yang berat bisa mengalami pingsan ketika melakukan aktivitas, karena katup yang sempit menghalangi ventrikel untuk memompa cukup darah ke arteri di otot, yang telah melebar untuk menerima darah yang kaya akan oksigen.

Pada stenosis aorta darah disemburkan dari ventrikel kiri melalui sebuah lubang yang sempit di katup aorta. Akibat tahanan terhadap semburan, kadang-kadang tekanan dalam ventrikel kiri meningkat sampai setinggi 300 mmHg, sedangkan tekanan di aorta tetap normal. Jadi, terbentuk pengaruh pipa semprot yang terjadi selama sistol, dengan darah yang disemburkan dengan kecepatan sangat tinggi melalui lubang kecil di katup. Keadaan ini menyebabkan turbulensi hebat pada darah di pangkal Aorta. Darah turbulen yang mengenai dinding aorta menimbulkan getaran yang hebat, dan murmur yang keras dihantarkan sepanjang Aorta bagian atas dan bahkan ke dalam arteri-arteri besar di leher. Suara ini kasar dan pada stenosis berat kadang-kadang demikian kerasnya sehingga dapat terdengar beberapa kaki dari pasien.

2.3 Isyarat Suara Jantung [11]

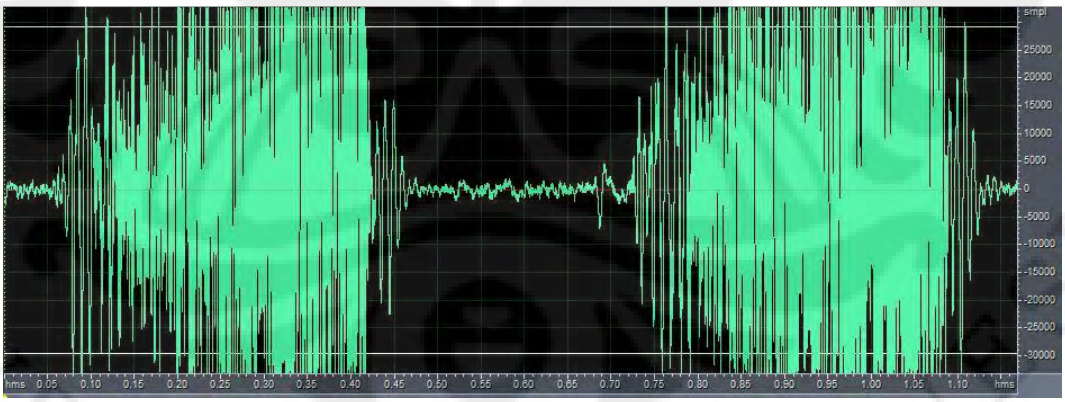
Suara jantung yang didengar oleh dokter dengan menggunakan stetoskop sebenarnya terjadi pada saat penutupan katup. Kejadian ini dapat menimbulkan anggapan yang keliru bahwa suara tersebut disebabkan oleh penutupan daun katup tersebut, tetapi sebenarnya disebabkan oleh efek arus pusar (*eddy*) dalam darah akibat penutupan katup tersebut (Carr, 2001). Suara jantung normal mempunyai rentang frekuensi antara 20 Hz hingga 200 Hz, sedangkan suara jantung abnormal mempunyai rentang frekuensi hingga 1000 Hz (Cromwell, 1980). Salah satu jenis *regurgitasi* menyebabkan *murmur* dalam rentang 100 hingga 600 Hz dan bahkanuntuk jenis *murmur* tertentu hingga 1000 Hz (Cromwell, 1980).

Detak jantung menghasilkan 2 suara yang berbeda yang dapat didengarkan pada stetoskop yang sering dinyatakan dengan *lub-dub*. Suara *lub* disebabkan oleh

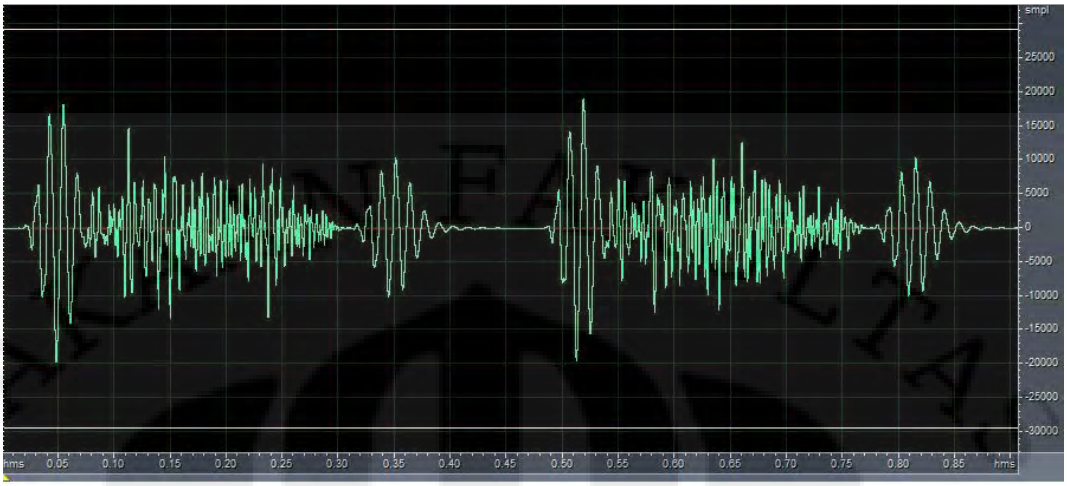
penutupan katup *tricuspid* dan *mitral* (*atrioventrikular*) yang memungkinkan aliran darah dari serambi jantung (*atria*) ke bilik jantung (*ventricle*) dan mencegah aliran balik. Umumnya hal ini disebut suara jantung pertama (S1), yang terjadi hampir bersamaan dengan timbulnya QRS dari elektrokardiogram dan terjadi sebelum periode jantung berkontraksi (*systole*).

Suara *dub* disebut suara jantung ke-dua (S2) dan disebabkan oleh penutupan katup *semilunar* (*aortic* dan *pulmonary*) yang membebaskan darah ke sistem sirkulasi paru-paru dan sistemik. Katup ini tertutup pada akhir *systole* dan sebelum katup *atrioventikular* membuka kembali. Suara S2 ini terjadi hampir bersamaan dengan akhir gelombang *T* dari EKG, suara jantung ke-tiga (S3) sesuai dengan berhentinya pengisian *atrioventikular*, sedangkan suara jantung ke-empat (S4) memiliki korelasi dengan kontraksi *atria.*

Jantung abnormal menghasilkan suara tambahan yang disebut *murmur* yang disebabkan oleh pembukaan katup yang tidak sempurna atau memaksa darah melewati bukaan sempit (*stenosis*) atau *regurgitasi* yang disebabkan oleh penutupan katup yang tidak sempurna dan mengakibatkan aliran balik darah, dalam masing-masing kasus suara yang timbul adalah akibat aliran darah dengan kecepatan tinggi yang melewati bukaan sempit.



Gambar 2.5 Contoh bentuk gelombang suara regurgitasi



Gambar 2.6 Contoh bentuk gelombang suara stenosis

Penyebab lain terjadinya *murmur* adalah adanya kebocoran *septum* yang memisahkan jantung bagian kiri dan kanan sehingga darah mengalir dari *ventrikel* kiri ke *ventrikel* kanan sehingga menyimpangkan sirkulasi sistemik (Anonim, 2004). Suara jantung normal mempunyai rentang frekuensi antar 20-200 Hz, sedangkan suara jantung abnormal mempunyai rentang frekuensi hingga 1000 Hz. Suara jantung S1 terdiri atas energi dalam rentang frekuensi 30-45 Hz, yang sebagian besar berada dibawah ambang dengar. Suara jantung S2 biasanya memiliki nada lebih tinggi dengan energi maksimum berada dalam rentang 50-70 Hz. Suara jantung S3 merupakan vibrasi yang sangat lemah dengan hampir semua energinya dibawah 30 Hz. Sedangkan *murmur* sering menghasilkan suara dengan nada yang lebih tinggi.

Beberapa jenis kelainan pada jantung beserta penjelasan singkatnya dapat dilihat pada Tabel 2.1 berikut.

Tabel 2.1 Jenis-jenis Jantung Abnormal

No	Nama suara jantung	Penjelasan
1	*Acute rheumatic fever*	Demam rematik akut, penyakit peradangan akut yang dapat menyertai faringitis yang disebabkan oleh *streptococcus beta-hemolyticus* grup A, cendrung berulang, dan sebagai penyebab terpenting penyakit jantung didapat pada anak dan dewasa muda
2	*Aortic insufficiency loud systolic ejection murmur, third sound*	Katup *aorta* tidak dapat menutup dengan sempurna dengan bising yang keras waktu fase *ejeksi sistolik* bunyi jantung ketiga
3	*Aortic stenosis opening snap of aortic early systolic ejection sound*	Pembukaan katup *aorta* tidak sempurna dengan bunyi pembukaan katup *aorta* pada pada fase *ejeksis sistolik* dini
4	*Atrial septal defect abnormal splitting of*	Bunyi jantung terbelah yang abnormal
5	*Coarctation of aorta systolic murmur*	*Coartasio aorta* dengan bising pada fase *sistolik*
6	*Complet heart block slow heart rate, varying first sound*	Blok jantung total dengan frekuensi jantung rendah, bunyi jantung pertama bermacam-macam.
7	*Mitral regurgiation holosystolic murmur*	*Mitral regurgitasi* dengan bising *holosistolik*
8	*Mitral regurgiation late systolic murmur crecendo type*	*Mitral regurgitasi* dengan bising diakhir sistolik tipe *crescendo*
9	*Mitral regurgiation mid systolic click and late systolic murmur*	*Mitral regurgitasi* dengan bunyi klik ditengah *sistolik* dan bising diakhir *sistolik*
10	*Mitral regurgiation systolic murmur, crescendo type*	Mitral *regurgitasi* dengan bising *sistolik* tipe *crescendo*
11	*Mitral regurgiationand mitral stenosis all sound features of mitral stenosis and mitral regurgitation*	Mitral *regurgitasi* dan *mitral stenosis* dengan menonjolkan semua bunyi semua bunyi *mitral stenosis* dan *mitral regurgitasi*
12	*Mitral regurgiattion systolic murmur, hight piched and blowing type*	*Mitral regurgitasi* dengan bising *sistolik tipe hight piched* dan *blowing*
13	*MItral regurgitation systolic Murmur*	*Mitral regurgitasi* dengan bising *sistolik*
14	*MItral regurguition third heart sound*	*Mitral stenosis* dengan bunyi jantung Ketiga
15	*Mitral stenosis accentuated first*	*Mitral stenosis* dengan menonjolkan bunyi jantung pertama
16	*Mitral stenosis opening snap*	*Mitral stenosis* dengan *opening snap*
17	*Mitral stenosis presystolic Murmur*	*Mitral stenosis* dengan bising *pansistolik*
18	*Mitral stenosis short middiastolic*	*Mitral stenosis* dengan tengah *diastolic* yang singkat
19	*Patent ductus arteriosus continuous machinery murmur*	bising khas seperti sebuah kereta api memasuki terowongan
20	*Pulmonary stenosis harsh systolic ejection murmur*	*Pulmonary* dengan bising *ejeksi sistolik* yang keras
21	*Right bundle branch block of first sound*	*Right bundle branch* blok dengan bunyi jantung pertama
22	*Systemic hypertension accentuated second sound*	*Hipertensi* sistemik dengan menonjolkan bunyi jantung kedua
23	*Tricuspid regurgitation holosystolic murmur*	*Triscupidal regurgitasi* dengan bising *holosistolik*
24	*Ventricular sepatal defect continuous murmur*	*Ventrikel septal defek* dengan bising yang bersifat kontinyu

2.4 Proses Pengolahan Sinyal Suara

Untuk dapat membedakan antara suatu penyakit jantung dengan penyakit jantung lainnya dengan menggunakan media suara harus melalui suatu proses.

Sinyal suara analog diubah menjadi sinyal dijital melalui proses sampling. Setelah itu informasi yang terkandung di dalam suara diekstraksi dan diubah ke dalam data-data vektor. Data vektor ini kemudian dikuantisasi dengan teknik VQ untuk melihat persebarannya pada *cluster*. Vektor pada *cluster* ini akan menentukan letak *centroid* yang kemudian dikumpulkan di dalam suatu *codeword*. Kumpulan beberapa *codeword* disebut *codebook*. Nilai-nilai *Codeword* inilah yang digunakan untuk menentukan parameter-parameter HMM. Proses lengkapnya akan dijelaskan pada bagian berikut ini.

2.4.1 *Sampling* [1]

Sampling adalah suatu proses untuk membagi-bagi suatu sinyal kontinyu dalam interval waktu yang telah ditentukan. *Sampling* ini dilakukan dengan mengubah sinyal analog menjadi sinyal dijital dalam fungsi waktu. Pengubahan bentuk sinyal ini bertujuan untuk mempermudah memproses sinyal masukan yang berupa analog karena sinyal analog memiliki kepekaan terhadap *noise* yang rendah, sehingga sulit untuk memproses sinyal tersebut.

Parameter-parameter yang menentukan hasil sampling adalah panjang interval yang digunakan. Frekuensi *sampling* yang digunakan mengikuti aturan Teorema Nyquist, bahwa besarnya nilai frekuensi *sampling* harus dua kali frekuensi tertinggi dari sinyal masukan, yang dinyatakan dengan :

$$f_s \geq 2\ x\ f_h \quad \text{.....} \quad (2.1)$$

Dimana : f_s adalah frekuensi sampling

f_h adalah frekuensi tertinggi sinyal masukan

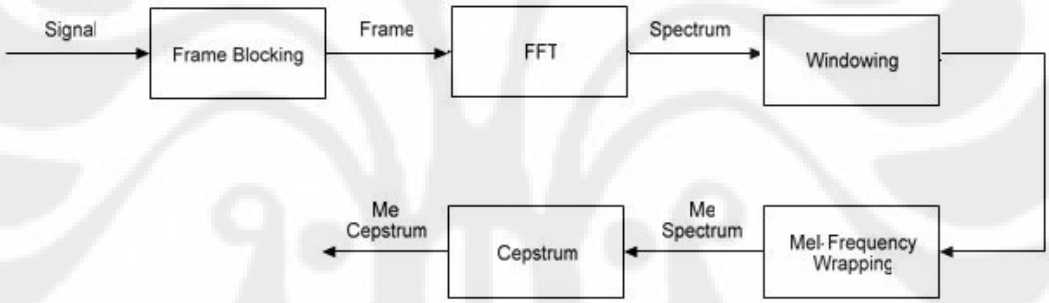
2.4.2 Ekstraksi [3]

Ekstraksi atau *feature extraction* merupakan proses dimana tiap-tiap *sample* sinyal akan diubah menjadi vektor-vektor data. Terdapat beberapa metode yang dapat digunakan untuk proses ini, antara lain *Linear Prediction Coding* (LPC) dan *Mel Frequency Cepstrum Coefficient* (MFCC). Metode yang akan digunakan pada proses ekstraksi dalam pengenalan suara ini adalah dengan menggunakan MFCC.

Mel Frequency Cepstrum Coefficient (MFCC) memiliki beberapa keunggulan dibandingkan dengan metode lainnya, antara lain :

1. Mampu menangkap informasi-informasi penting yang terkandung dalam sinyal suara.
2. Menghasilkan data seminimal mungkin, tanpa menghilangkan informasi-informasi penting yang ada.
3. Mengadaptasi organ pendengaraan manusia dalam melakukan persepsi terhadap sinyal suara.
4. Menghasilkan pendekatan yang lebih baik terhadap sistem pendengaran manusia karena menggunakan fungsi logaritmik dalam perhitungannya.

MFCC di sini bertujuan untuk menghasilkan *cepstrum* yang akan digunakan dalam membentuk *codeword*. Blok diagram dari MFCC ditunjukkan pada Gambar 2.7.



Gambar 2.7 Blok Diagram MFCC [4]

Proses MFCC diawali dengan membagi sinyal (suara) menjadi beberapa *frame* melalui proses *frame blocking*. Kemudian dari setiap frame dicari spektrum amplitudonya dengan terlebih dahulu mengubah masing-masing *frame* dari domain waktu ke domain frekuensi dengan menggunakan *Fast Fourier Transform* (FFT). Setelah itu dilakukan *windowing* pada setiap *frame*. *Windowing* bertujuan untuk meminimalisasi diskontinuitas sinyal dan distorsi spektral. Selanjutnya dilakukan proses *mel-frequency wrapping* untuk memperoleh sinyal spektrum dalam *mel-scale* dari hasil FFT. Langkah terakhir adalah mengubah hasil *log mel spectrum* ke dalam domain waktu dan menghasilkan MFCC sebagai hasil akhir.

Urutan dan cara kerja MFCC dapat dijelaskan sebagai berikut[13].

1. *Frame Blocking*

   Pada proses *frame blocking*, suatu sinyal suara yang diterima akan dibagi ke dalam N frame berdasarkan persamaan :

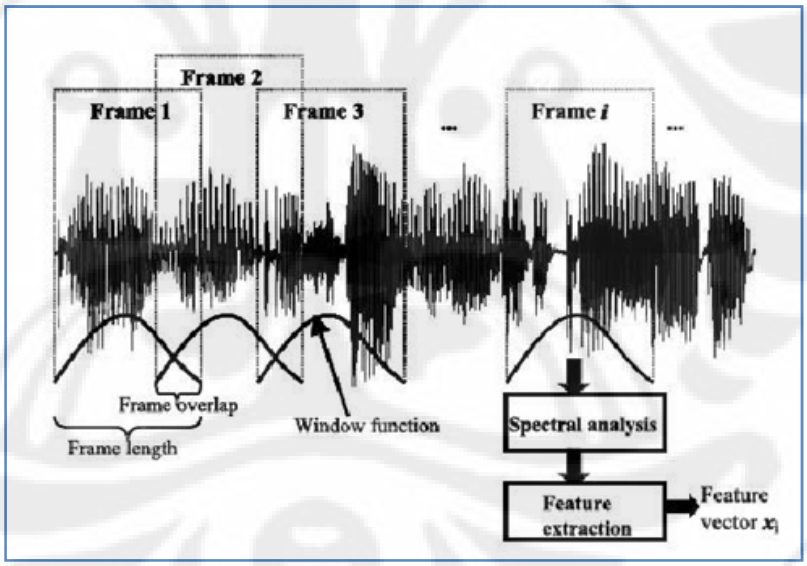
   t = N/*fs*.....(2.2)

   Dimana : N = banyak data per frame

   fs = frekuensi sampling

   t = panjang frame

   Panjang frame yang biasanya digunakan dalam pemrosesan sinyal adalah antara 10 ms – 30 ms. Proses *frame blocking* ini terus dilakukan sampai sinyal dapat diproses seluruhnya. Selain itu, proses ini umumnya dilakukan secara overlapping untuk setiap frame-nya. Panjang daerah overlap yang digunakan secara umum adalah 30 % sampai 50 % dari panjang frame-nya.



Gambar 2.8 Proses *frame blocking* dan *windowing*

2. *Discrete Fourier Transform* dan *Fast Fourier Transform* [1]

   Tujuan utama dari transformasi Fourier ini adalah untuk mengubah sinyal dari domain waktu menjadi spektrum pada domain frekuensi. Dalam pemrosesan suara, proses tersebut sangatlah menguntungkan karena sinyal pada domain frekuensi dapat diproses lebih mudah dibandingkan dengan

sinyal pada domain waktu. Hal ini dikarenakan pada domain frekuensi, kuat lemahnya suara tidak terlalu berpengaruh.

Discrete Fourier Transform (DFT) adalah suatu metode yang digunakan untuk mengubah domain suatu gelombang dari domain waktu ke domain frekuensi. Transformasi diskrit merupakan transformasi dimana masukan dan keluaran bernilai diskrit yang digunakan untuk manipulasi di komputer. Rumus DFT untuk mengubah N data dari domain waktu ke domain frekuensi dapat diperlihatkan sebagai berikut :

$$X(k) = F_D[x(nT)] = \sum_{n=0}^{N-1} x(nT)e^{-jk\Omega nT} \quad \text{..... (2.5)}$$

k = 0,1,2,...,N-1

Dimana : F_D adalah transformasi Fourier

X(nT) adalah sinyal input

T adalah interval waktu antar nilai diskrit

K adalah angka harmonik dari komponen transformasi

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 dengan mempergunakan persamaan :

$$X_n = \sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N} \quad \text{..... (2.6)}$$

k = 0,1,...,N-1

Faktor dari $e^{-j2\pi nk/N}$ dapat dituliskan sebagai W_N.

$$W_N = e^{-j2\pi nk/N} \quad \text{..... (2.7)}$$

Sehingga persamaan akan menjadi :

$$X_n = \sum_{n=0}^{N-1} x_n W_n^{kn} \quad \text{..... (2.8)}$$

dengan k = 0, 1, ... , N-1 ; dimana : X_n adalah sinyal hasil DFT

x_n adalah sinyal masukan

W_n^{kn} adalah *twidle factors*

Perhitungan DFT memerlukan operasi sebanyak M^2, sedangkan FFT dapat memenuhi hal yang sama dengan operasi sebanyak $M \log_2 M$.

Dengan demikian FFT merupakan *fast algorithm* untuk mengimplementasikan DFT.

3. *Windowing*

Windowing dilakukan untuk memperkecil penyimpangan pada sinyal yang diskontinu di awal dan di akhir masing-masing *frame*. Sinyal suara yang dipotong-potong menjadi beberapa frame akan menyebabkan efek diskontinuitas sehingga menyebabkan kesalahan data pada proses fourier transform. *Windowing* diperlukan untuk mengurangi efek diskontinuitas dari potongan-potongan sinyal. Dalam penelitian ini digunakan metode *Hamming Window*. Metode ini dapat menghasilkan *sidelobe level* yang tidak terlalu tinggi (kurang lebih -43 dB) dan noise yang dihasilkan tidak terlalu besar (1,36 BINS). Berikut ini adalah persamaannya :

$$w(n) = 0.54 - 0.46cos\left(\frac{2\pi n}{N-1}\right) \quad \text{(2.3)}$$

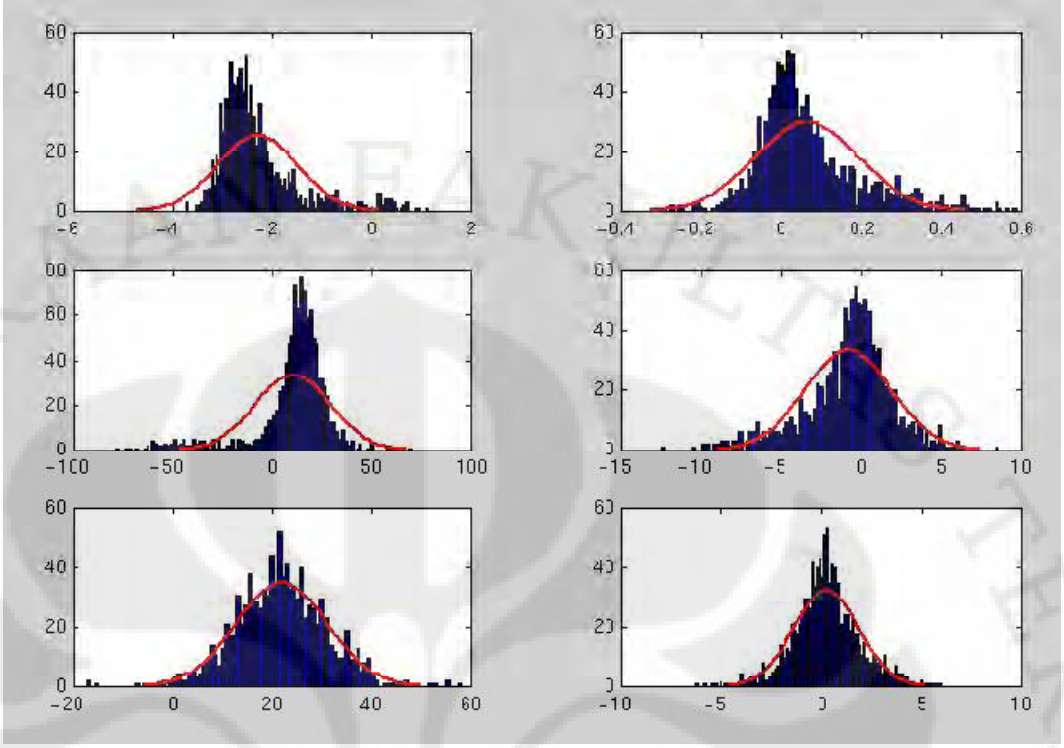
Dimana : N = lebar window

n = 0,1,...,(N-1)/2 , untuk N ganjil

= 0,1,...,(N/2)-1 , untuk N genap

Hasil dari proses *windowing* ini adalah berupa suatu sinyal dengan persamaan :

$$y_1(n) = x_i(n)w(n), \quad 0 \le n \le N-1 \quad \text{(2.4)}$$

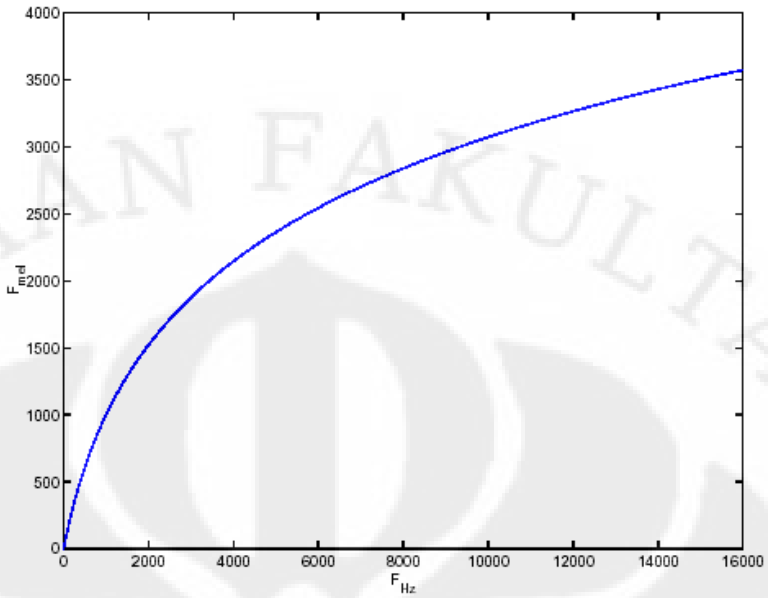


Gambar 2.9 Windowing pada spektrum frekuensi

4. Mel-Freq Warping

Persepsi manusia terhadap frekuensi dari sinyal suara tidak mengikuti skala linear. Frekuensi yang sebenarnya (dalam Hz) pada sebuah sinyal akan diukur manusia secara subyektif dengan menggunakan *mel scale*. Skala mel-frekuensi adalah pemetaan frekuensi secara linear untuk frekuensi dibawah 1 kHz dan logaritmik untuk frekuensi diatas 1 kHz. Sebagai titik referensi, *pitch* dari 1kHz, 40 dB diatas *perceptual hearing threshold*, didefinisikan sebagai 1000 mels. Oleh karena itu, dapat digunakan formula sebagai berikut untuk menghitung *mels* untuk frekuensi yang diberikan dalam Hz.

$$mel(f) = 2595 * \log_{10}(1 + f/700) \quad \text{..... (2.9)}$$



Gambar 2.10 Grafik mel-frekuensi versus frekuensi

5. *Cepstrum*

Langkah terakhir dalam *feature extraction* yaitu mengubah kembali *log mel spectrum* ke dalam domain waktu. Hasilnya disebut *mel frequensi ceptrum coeffecient (MFCC).* Representatif *spectral* dari *speech spectrum* memberikan representatif yang baik untuk *local spectral properties* dari sinyal suara untuk analisa *frame* yang diberikan. Karena *mel spectrum coeffecient* (dan logaritmiknya) adalah angka real, kita dapat mengubahnya ke time domain menggunakan *Discrete Cosine Transform (DCT).* Oleh karena itu *mel power spectrum coefficient* tersebut merupakan hasil dari langkah terakhir yang dinotasikan dengan $\tilde{S}_k$, dimana $k = 1,2,...,K$, maka MFCC, $\tilde{c}_n$ dapat dihitung dengan persamaan :

$$\tilde{c}_n = \sum_{k=1}^{K} (\log \tilde{S}_k) \cos\left[n\left(k - \frac{1}{2} \right) \frac{\pi}{K} \right] \; ; \; k = 1, 2, \ldots, K \quad \text{.....} \quad (2.10)$$

K = jumlah koefisien yang diharapkan

$\tilde{S}_k$ = keluaran dari proses Mel-Freq Wrapping

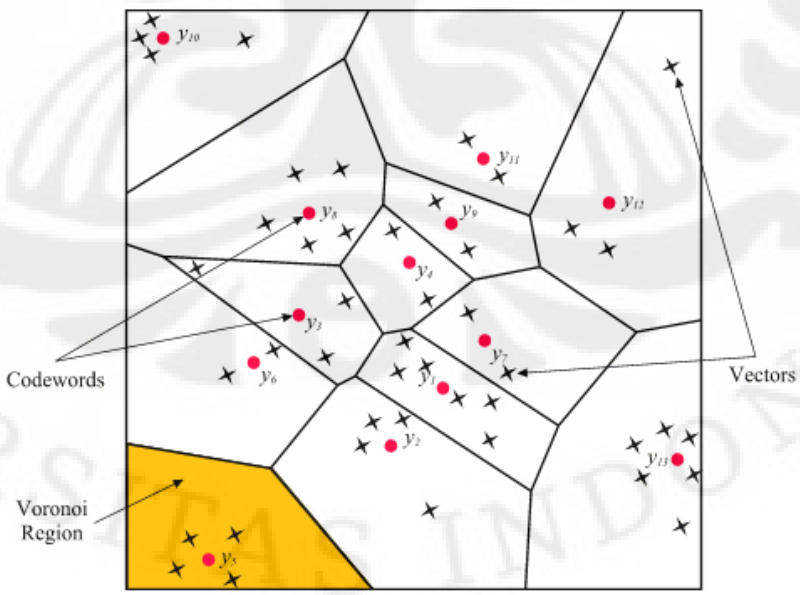
2.4.3 Kuantisasi Vektor [3]

Kuantisasi vektor merupakan teknik kuantisasi klasik dimana dilakukan pemodelan dari fungsi kepadatan probabilitas dengan distribusi vektor. Kuantisasi vektor memetakan vektor dengan dimensi k pada ruang vektor R_k menjadi suatu bentuk vektor berhingga $Y = \{y_i : i = 1, 2, ..., n\}$. Vektor y_i disebut sebagai vektor kode. Vektor-vektor ini merupakan vektor-vektor data yang diperoleh dari hasil ekstraksi yang disebut dengan *codeword*. Kumpulan dari *codeword* ini disebut dengan *codebook*.

Gambar 2.11 menggambarkan vektor pada suatu ruang dengan garis horizontal menunjukkan nilai real dan garis vertikal menunjukkan nilai imajiner dari vektor. Setiap cluster dari vektor menunjukkan *centroid*-nya, dan setiap *codeword* berada pada daerah *Voroinoi*-nya masing-masing. Vektor masukan ditandai dengan x sedangkan *centroid* ditandai dengan bulatan berwarna merah. Representasi *centroid* ditentukan berdasarkan jarak *Euclidian* terdekat dari vektor masukan. Jarak *Euclidian* didefinisikan dengan persamaan berikut :

$$d(x, y_i) = \sqrt{\sum_{j=1}^{k}(x_j - y_{ij})^2} \quad \text{..... (2.11)}$$

dimana x_j adalah komponen ke-j dari vektor masukan dan y_{ij} adalah komponen ke-j dari *centroid* y_i.



Gambar 2.11 Contoh *codeword* pada ruang dua dimensi [13]

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

1. Mendesain vektor *codebook* yang merupakan *centroid* dari keseluruhan hasil pelatihan vektor.
2. Melipatgandakan ukuran dari *codebook* dengan membagi masing-masing codebook C_n menurut aturan :

   $C_n^+ = C_n(1+\varepsilon)$ ..... (2.14)

   $C_n^- = C_n(1+\varepsilon)$ ..... (2.15)

   dimana n bervariasi dari satu sampai dengan *current size codebook* dan epsilon adalah parameter *splitting*. (epsi = 0,01)
3. *Nearest Neighbour Search*

   Mengelompokkan *training* vektor yang mengumpul pada blok tertentu. Selanjutnya menentukan *centroid* dalam *current codebook* yang terdekat dan memberikan tanda vektor yaitu *cell* yang diasosiasikan dengan *centroid-centroid* yang terdekat.
4. *Centroid Update*

   Menentukan *centroid* baru yang merupakan *codeword* yang baru pada masing-masing *cell* dengan menggunakan *training* vektor pada *cell* tsb.
5. Iterasi 1

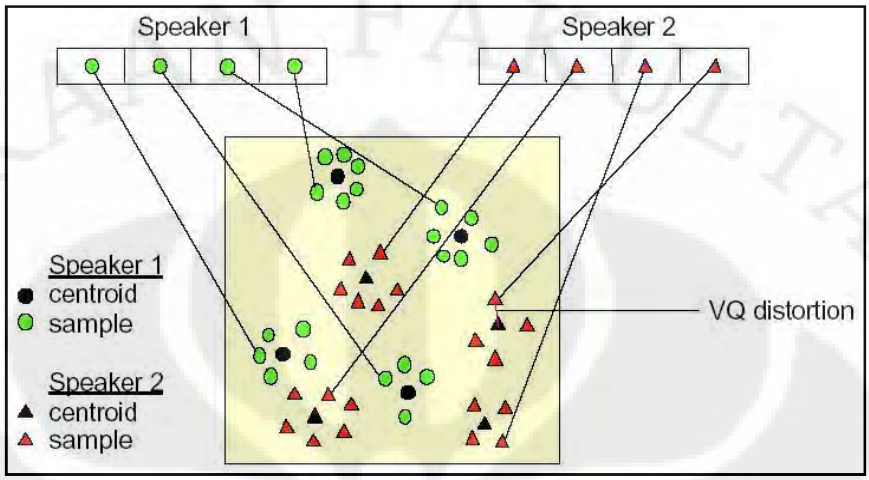
   Mengulang step 3 dan 4 sampai jarak rata-rata dibawah *present treshold*.
6. Iterasi 2

   Mengulang step 2, 3, dan 4 sampai *codebook* berukuran M.

Jarak suatu vektor ke *centroid* terdekat disebut dengan distorsi. Pada proses pengenalan, total distorsi yang paling kecil antara *codeword* dari database dan *codebook* VQ dari input merupakan hasil identifikasi.

Gambar 2.12 menunjukan konseptual diagram untuk mengilustrasikan proses *recognition*. Gambar ini hanya mengilustrasikan 2 (dua) buah suara dari 2 (dua) *speaker* dalam ruang akustik dua dimensi. Lingkaran menunjukan vektor data dari suara 1, sedangkan segitiga adalah vektor data dari suara 2. Hasil

codeword-nya ditunjukkan dengan lingkaran dan segitiga hitam untuk suara 1 dan 2. Jarak dari sebuah vektor ke *codeword* terdekat disebut distorsi.

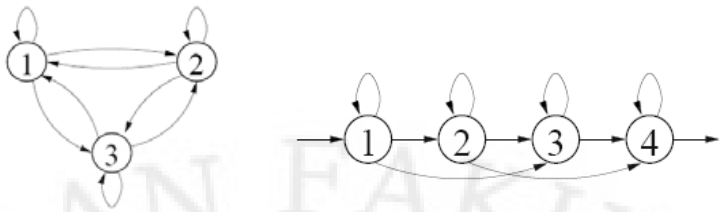


Gambar 2.12 Diagram konsep pembentukan *codebook* dengan *vector quantization*.

2.5 Hidden Markov Model

Hidden Markov Model (HMM) adalah suatu teknik untuk membentuk model statistik berdasarkan prinsip probabilitas. Model tersebut digunakan untuk memprediksi suatu keluaran berdasarkan data-data yang telah dimasukkan dan proses pelatihan yang telah dilakukan. Model statistik ini merupakan suatu sistem yang diasumsikan sebagai proses Markov dengan parameter-parameter yang belum diketahui dan parameter-parameter yang tersembunyi tersebut harus ditentukan dari parameter yang dapat diamati (*observable*). Parameter model yang diambil kemudian dapat digunakan untuk keperluan analisa selanjutnya, misalnya untuk aplikasi pengenalan gelombang suara.

Bentuk umum dari rantai Markov adalah bentuk *ergodic* seperti yang dapat dilihat pada Gambar 2.9(a). Namun dapat juga dimodelkan dengan model *left- right Markov* seperti pada Gambar 2.9(b).



(a) (b)

Gambar 2.13 Model Markov (a) *Ergodic* (b) *Left-right* [13]

HMM memiliki 3 (tiga) parameter utama yang harus dicari nilainya terlebih dahulu. Ketiga parameter tersebut adalah sebagai berikut.

1. **Parameter A** disebut sebagai probabilitas transisi, merupakan probabilitas kedudukan suatu *state* terhadap semua *state* yang ada, termasuk kedudukan terhadap *state* itu sendiri. Penggunaan probabilitas transisi dapat ditunjukkan pada gambar berikut :

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

$$A = a_{ij} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

2. **Parameter B** disebut sebagai probabilitas *state*, merupakan probabilitas kemunculan suatu *state* dalam deretan seluruh *state* yang ada.

   Parameter B dalam HMM dituliskan dalam bentuk matriks kolom dengan ukuran Mx1 dimana M merupakan jumlah seluruh *state* yang ada. Misalnya terdapat n buah *state* dalam suatu kondisi, maka matriks B yang terbentuk ditunjukkan oleh persamaan berikut :

$$B = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{bmatrix}$$

3. **Parameter $\boldsymbol{\pi}$** disebut sebagai probabilitas awal, yaitu probabilitas kemunculan suatu *state* di awal.

   Sama halnya dengan parameter B, parameter π juga dituliskan dalam bentuk matriks kolom dengan ukuran Mx1 dimana M adalah jumlah *state*-nya. maka parameter π yang dihasilkan akan ditunjukkan seperti pada persamaan berikut :

$$\pi = \begin{bmatrix} c_1 \\ c_2 \\ . \\ . \\ c_n \end{bmatrix}$$

Elemen π, A, dan B merupakan parameter-parameter markov dalam HMM yang tidak diketahui atau tersembunyi (*hidden*). Ketiga parameter tersebut digabungkan menjadi sebuah parameter HMM dan dapat dituliskan dalam bentuk $\lambda = (A, B, \pi)$.

Jika diberikan suatu model $P(O|\lambda)$ dengan probabilitas urutan observasi $O = O_1, O_2, \ldots, O_T$, maka untuk mengetahui nilai probabilitas observasinya diperlukan suatu urutan *state* yang tetap, misalkan

$$Q = q_1\ q_2\ \ldots\ q_T \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.16)$$

di mana q_1 adalah *initial-state*. Maka probabilitas urutan observasi *O* untuk urutan *state* persamaan (2.27) adalah

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(Q_t|q_t,\lambda) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.17)$$

Sehingga diperoleh $P(O|Q, \lambda) = bq_1(O_1)\ .\ bq_2(O_2)\ \ldots\ bq_T(O_T)$ (2.18)

Probabilitas dari urutan *state Q* maka dapat ditulis sebagai berikut

$$P(Q|\lambda) = \pi_{q_1}\ a_{q_1\ q_2} a_{q_2\ q_3} \ldots\ a_{q_T - 1 q_T} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.19)$$

Probabilitas gabungan *O* dan *Q* yang merupakan probabilitas saat *O* dan *Q* muncul bersamaan adalah hasil perkalian dari keduanya atau dapat ditulis sebagai berikut :

$$P(O, Q|\lambda) = P(O|Q, \lambda)\ P(Q, \lambda) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.20)$$

Probabilitas observasi *O* diperoleh dengan menjumlahkan probabilitas gabungan dari semua kemungkinan urutan *state q*, yaitu :

$$P(O|\lambda) = \sum_{allQ} P(O|Q,\lambda)\ \mathrm{P}\ (\mathrm{Q}|\lambda) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.21)$$

atau dapat juga dituliskan sebagai berikut :

$P(O|\lambda) = \sum_{q_1 q_2 \ldots q_T} \pi_{q_1} b_{q_1}(O_1)\ a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(O_T)$ .....(2.22)

Log of Probability (LoP) = Log $P(O|\lambda)$ ..... (2.23)

Jika terdapat keadaan dimana :

State 1 : waveform segment 1 (W_1)

State 2 : waveform segment 2 (W_2)

State 3 : waveform segment 3 (W_3)

State 4 : waveform segment 4 (W_4)

State 5 : waveform segment 5 (W_5)

Maka, probabilitas dari observasi HMM :

Suara 1 → (w_1,w_2,w_2, w_1, w_1) = $c_1 * a_{12} * b_2 * a_{22} * b_2 * a_{21} * b_1 * a_{11} * b_1$

Suara 2 → (w_1, w_2, w_1, w_3, w_1) = $c_1 * a_{12} * b_2 * a_{21} * b_1 * a_{13} * b_3 * a_{31} * b_1$

⋮

Suara x → (w_4, w_5, w_4, w_5, w_4) = $c_4 * a_{45} * b_5 * a_{54} * b_4 * a_{45} * b_5 * a_{54} * b_4$

proses yang terjadi adalah :

1. Gelombang yang telah terbagi menjadi gelombang-gelombang kecil pada *frame blocking* akan dikenali melalui *codebook* yang dimiliki. Pada proses pencocokan dengan *codebook* akan dihitung jarak dari tiap gelombang dengan *centroid-centroid*. Jarak terdekat akan menentukan urutan kode observasi.
2. Gelombang yang telah dikenali berdasarkan codebook akan membentuk suatu *state*. Dari *state* ini akan dicari nilai masing-masing parameter HMM-nya, yang perhitungannya dicocokkan dengan nilai pada parameter HMM *database*.

Dari contoh di atas, dapat diketahui bahwa suara 1 terbentuk dari gelombang w_1, gelombang w_2, gelombang w_2, gelombang w_1, dan gelombang w_1. Tiap suara dibentuk oleh susunan gelombang yang berbeda-beda. Susunan-susunan gelombang tersebut memiliki probabilitas transisi yang bergantung terhadap perubahan gelombangnya.

BAB III

RANCANG BANGUN PENELITIAN

Penelitian ini menggunakan piranti lunak *MATLAB R2008a* sebagai media untuk melakukan simulasi pengenalan penyakit jantung melalui suara denyut jantung dan software pendukung *Adobe Audition 1.0* sebagai pengolah gelombang suara denyut jantung. Sedangkan untuk spesifikasi komputer yang digunakan adalah sebagai berikut:

Processor : AMD Turion 64 X2 Mobile Technology TL-60 2.00 GHz
Memory (RAM) : 2.00 GB
Sistem Operasi : Windows Vista Home Premium
VGA Card : ATI Radeon

Dalam perancangan sistem pengenalan penyakit jantung ini, terdapat dua proses utama yang harus ditempuh, yaitu pembuatan *database* dan pengenalan suara (denyut jantung).

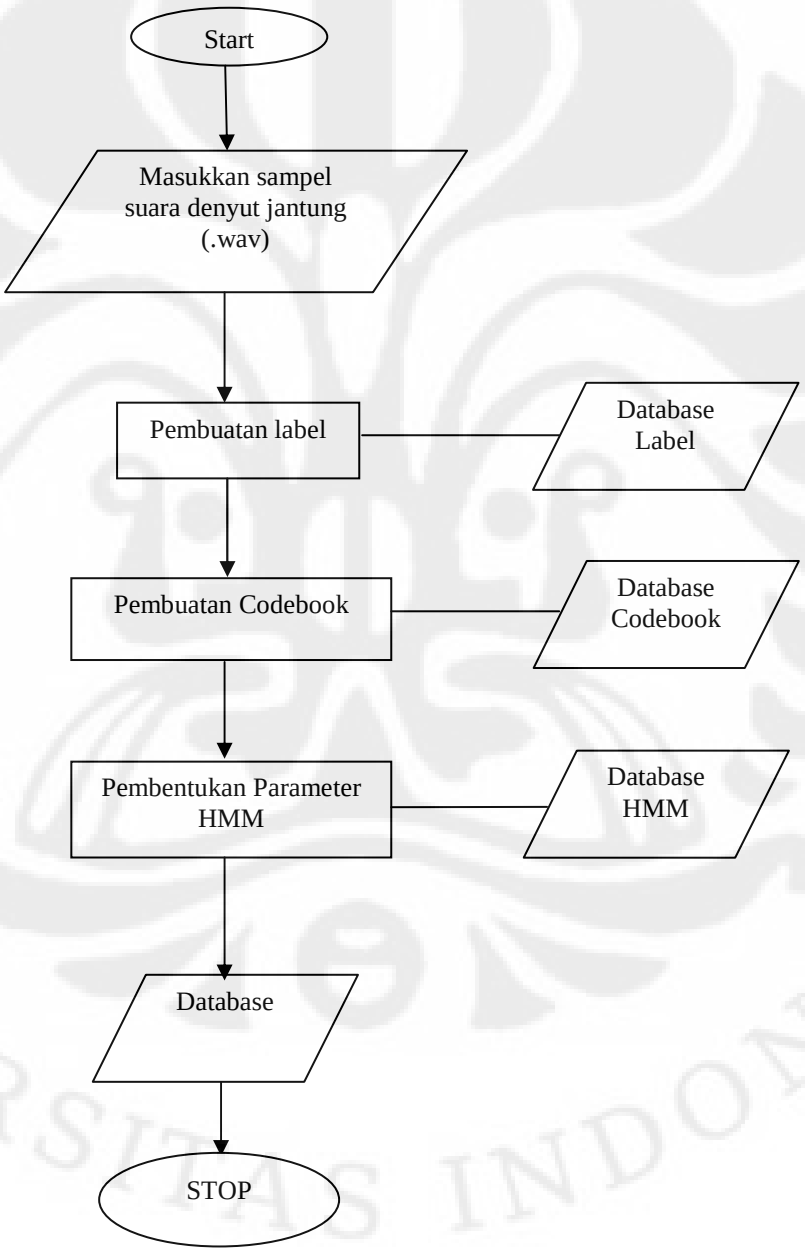
3.1 Pembuatan *Database*

Pada proses ini, tiap-tiap sampel suara denyut jantung yang merupakan sinyal masukan, akan mengalami tiga proses utama secara berurutan, yaitu tahap pembuatan label, tahap pembuatan *codebook*, dan tahap pembentukan parameter-parameter HMM. Secara umum akan digambarkan melalui *flowchart* pada gambar 3.1.

3.1.1 Pelabelan

Pada proses pembuatan label ini, tiap-tiap sampel suara penyakit akan didaftarkan pada suatu label yang diberi nama sesuai dengan nama penyakit yang dimaksud, sehingga jumlah label sama dengan jumlah penyakit. Nama label inilah yang nantinya akan menjadi keluaran akhir pada simulasi ini. File-file sampel suara tersebut akan mengalami proses *sampling* dengan frekuensi sampling sebesar 6 kHz. Berikut ini adalah algoritma proses pembuatan label :

```
Untuk i = 1 sampai banyaknya penyakit
     Masukkan nama penyakit;
     Masukkan jumlah sampel yang didaftarkan ;
     Masukkan nilai durasi sinyal sampel;
     Baca file (.wav) sampel;
     Nama label [i] = nama penyakit ;
Kembali
```



Gambar 3.1 Diagram alir pembuatan database jantung

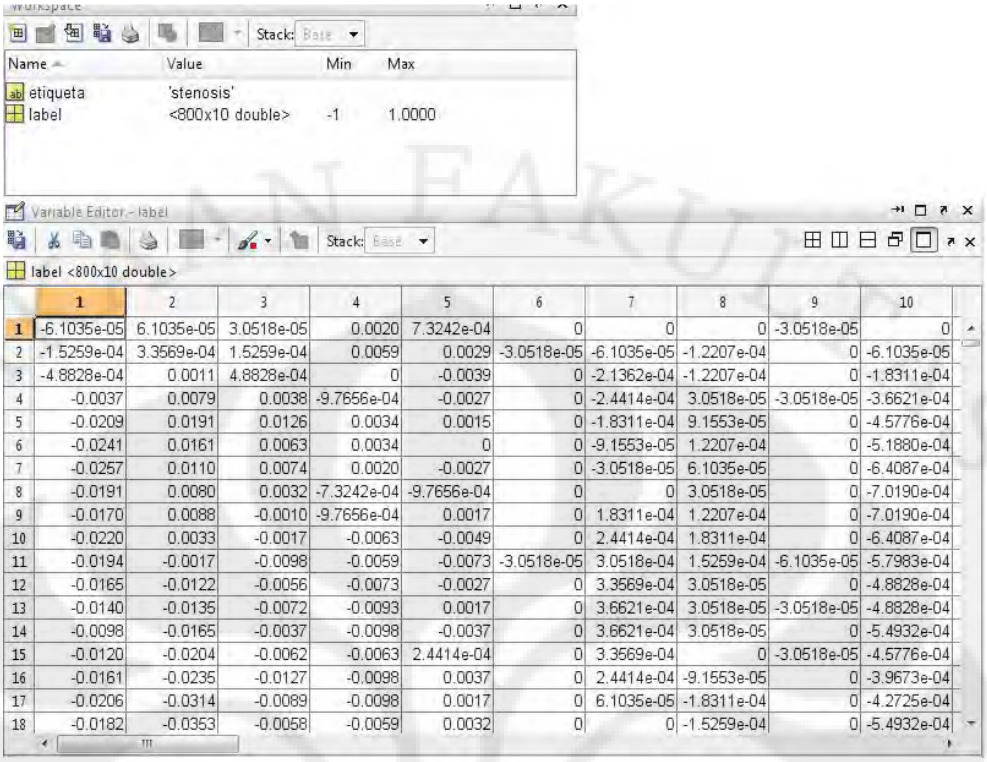
Berikut ini adalah tampilan software untuk proses pelabelan :

Gambar 3.2 Tampilan Program Pembuatan Label

Pada software ini, terdapat tiga jenis inputan, yaitu : *Label Num, Training, dan Label Name*.

1. *Label Num* menunjukkan urutan label yang akan dibuat, dan nantinya file yang akan tersimpan dalam bentuk format "(Label+Label Num).mat".
2. *Duration* menunjukkan lamanya durasi gelombang suara sampel.
3. *Training* menunjukkan banyaknya sampel yang akan diproses untuk setiap labelnya. Besarnya dapat ditentukan sesuai dengan keinginan user.
4. *Label Name* menunjukkan nama file sampel suara yang akan diproses. *Label Name* ini juga yang nantinya akan menjadi keluaran akhir dari proses simulasi ini.

Sebagai contoh, *Label Num* = 1, *Training* = 10 dan *Label Name* = stenosis. Maka file sampel suara yang akan diproses sebanyak 13 buah dan harus diberi nama "stenosis1.wav" – "stenosis13.wav", sedangkan file label yang akan terbentuk bernama Label1.mat.



Name	Value	Min	Max
etiqueta	'stenosis'		
label	<800x10 double>	-1	1.0000

label <800x10 double>

	1	2	3	4	5	6	7	8	9	10
1	-6.1035e-05	6.1035e-05	3.0518e-05	0.0020	7.3242e-04	0	0	0	-3.0518e-05	0
2	-1.5259e-04	3.3569e-04	1.5259e-04	0.0059	0.0029	-3.0518e-05	-6.1035e-05	-1.2207e-04	0	-6.1035e-05
3	-4.8828e-04	0.0011	4.8828e-04	0	-0.0039	0	-2.1362e-04	-1.2207e-04	0	-1.8311e-04
4	-0.0037	0.0079	0.0038	-9.7656e-04	-0.0027	0	-2.4414e-04	3.0518e-05	-3.0518e-05	-3.6621e-04
5	-0.0209	0.0191	0.0126	0.0034	0.0015	0	-1.8311e-04	9.1553e-05	0	-4.5776e-04
6	-0.0241	0.0161	0.0063	0.0034	0	0	-9.1553e-05	1.2207e-04	0	-5.1880e-04
7	-0.0257	0.0110	0.0074	0.0020	-0.0027	0	-3.0518e-05	6.1035e-05	0	-6.4087e-04
8	-0.0191	0.0080	0.0032	-7.3242e-04	-9.7656e-04	0	0	3.0518e-05	0	-7.0190e-04
9	-0.0170	0.0088	-0.0010	-9.7656e-04	0.0017	0	1.8311e-04	1.2207e-04	0	-7.0190e-04
10	-0.0220	0.0033	-0.0017	-0.0063	-0.0049	0	2.4414e-04	1.8311e-04	0	-6.4087e-04
11	-0.0194	-0.0017	-0.0098	-0.0059	-0.0073	-3.0518e-05	3.0518e-04	1.5259e-04	-6.1035e-05	-5.7983e-04
12	-0.0165	-0.0122	-0.0056	-0.0073	-0.0027	0	3.3569e-04	3.0518e-05	0	-4.8828e-04
13	-0.0140	-0.0135	-0.0072	-0.0093	0.0017	0	3.6621e-04	3.0518e-05	-3.0518e-05	-4.8828e-04
14	-0.0098	-0.0165	-0.0037	-0.0098	-0.0037	0	3.6621e-04	3.0518e-05	0	-5.4932e-04
15	-0.0120	-0.0204	-0.0062	-0.0063	2.4414e-04	0	3.3569e-04	0	-3.0518e-05	-4.5776e-04
16	-0.0161	-0.0235	-0.0127	-0.0098	0.0037	0	2.4414e-04	-9.1553e-05	0	-3.9673e-04
17	-0.0206	-0.0314	-0.0089	-0.0098	0.0017	0	6.1035e-05	-1.8311e-04	0	-4.2725e-04
18	-0.0182	-0.0353	-0.0058	-0.0059	0.0032	0	0	-1.5259e-04	0	-5.4932e-04

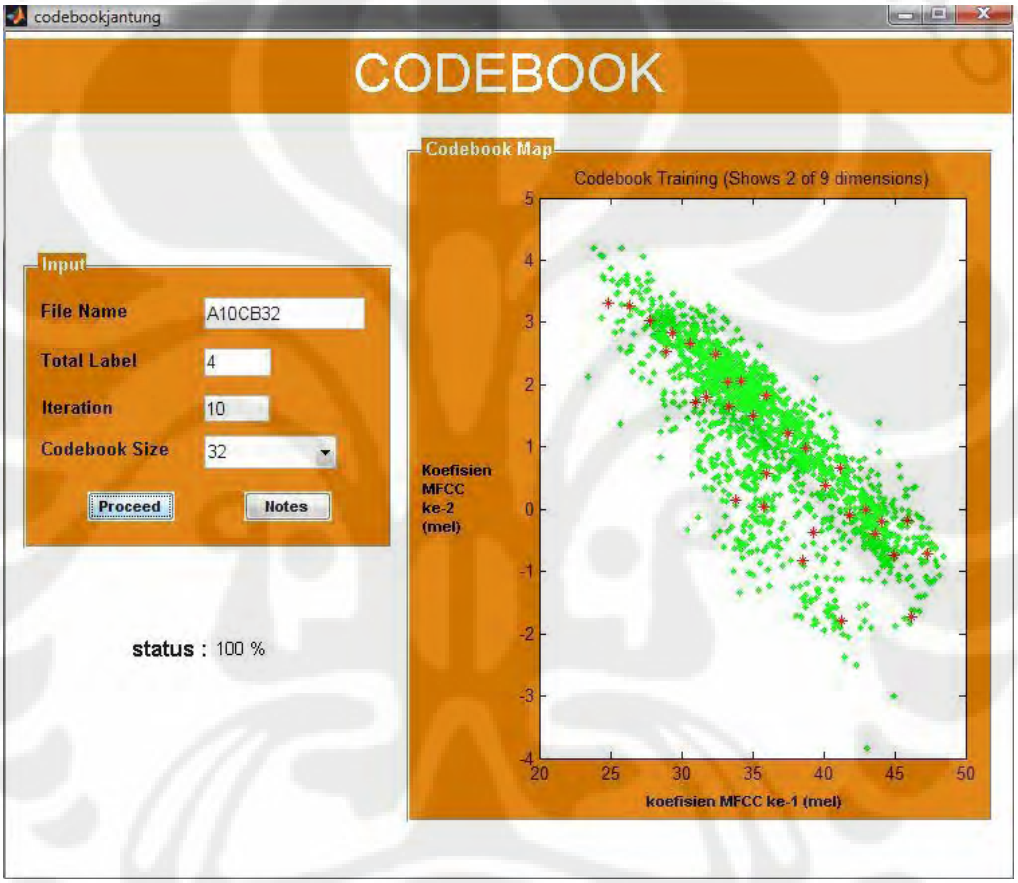
Gambar 3.3 Keluaran proses pelabelan

Setelah melakukan eksekusi program, maka keluaran yang dihasilkan pada proses pembuatan label ini adalah berupa matriks yang disimpan dalam format ".mat". Jumlah baris pada matriks ini menunjukkan banyaknya jumlah sampel yang dipotong berdasarkan perhitungan persamaaan (2.2). Sedangkan jumlah kolom pada matriks ini menunjukkan banyaknya sampel suara denyut jantung yang diproses.

3.1.2 Pembuatan *Codebook*

Proses pembuatan *database* selanjutnya adalah pembuatan *codebook*. Pada tahap ini dilakukan proses penggabungan dari semua label hasil proses sebelumnya ke dalam sebuah *file codebook* dengan format ".mat". Proses ini dimulai dengan ekstraksi sampel-sampel suara jantung (telah dijelaskan pada bab sebelumnya), yang akan menghasilkan titik-titik vektor melalui proses FFT. Titik-titik ini kemudian dipetakan pada suatu grafik dengan teknik kuantisasi vektor (VQ). VQ merupakan proses pemetaan vektor dari ruang vektor yang besar menjadi daerah yang terbatas (*cluster*). Setiap *cluster* ini direpresentasikan

oleh sebuah titik *centroid* yang disebut *codeword*. Kumpulan semua *codeword* disebut dengan *codebook*. Titik-titik sampel yang berdekatan dikuantisasikan ke satu titik vektor sehingga diperoleh beberapa titik vektor atau *centroid*. Kemudian nilai-nilai *centroid* dari setiap *sample* untuk tiap jenis penyakit yang diperoleh dari proses pembelajaran tersebut akan disimpan menjadi sebuah *codebook*.



Gambar 3.4 Tampilan program pembuatan *codebook*

Algoritma pembuatan *codebook* secara umum dapat dituliskan sebagai berikut :

```
definisikan besar vektor
untuk i = 1 sampai jumlah sample
      ekstraksi sampel [i];
      hitung FFT untuk setiap sample [i];
      sample point [i] = nilai FFT;
      plot grafik;
```

```
kembali

definisikan ukuran codebook dan iterasi;
untuk j = 1 sampai jumlah cluster
      hitung centroid sebanyak iterasi;
      simpan centroid [j] berdasarkan urutan labelnya;
kembali
```

Pada program ini terdapat empat buah inputan, yaitu :

1. *File name* diisi dengan nama file yang diinginkan yang nantinya akan tersimpan dalam format ".mat".
2. *Codebook size* merupakan ukuran *codebook* yang akan digunakan. Pada program ini tersedia ukuran *codebook* 32, 64 dan 128.
3. *Iteration* merupakan banyaknya proses pengulangan yang dilakukan dalam menentukan *centroid* agar mendapatkan *centroid* yang cukup presisi. Semakin besar jumlah iterasinya, maka akan semakin presisi letak *centroid* yang didapatkan, namun dengan mengambil iterasi yang sangat tinggi proses pembuatan *codebook* akan berjalan sangat lambat, oleh karena itu iterasi yang dilakukan juga tidak perlu terlalu besar. Dalam skripsi ini ditentukan *default* untuk besarnya iterasi adalah 10 dengan harapan letak centroid yang diperoleh cukup presisi dan waktu proses lebih cepat.
4. *Total label* dimasukkan sesuai dengan jumlah label yang telah dibuat sebelumnya pada proses pelabelan.

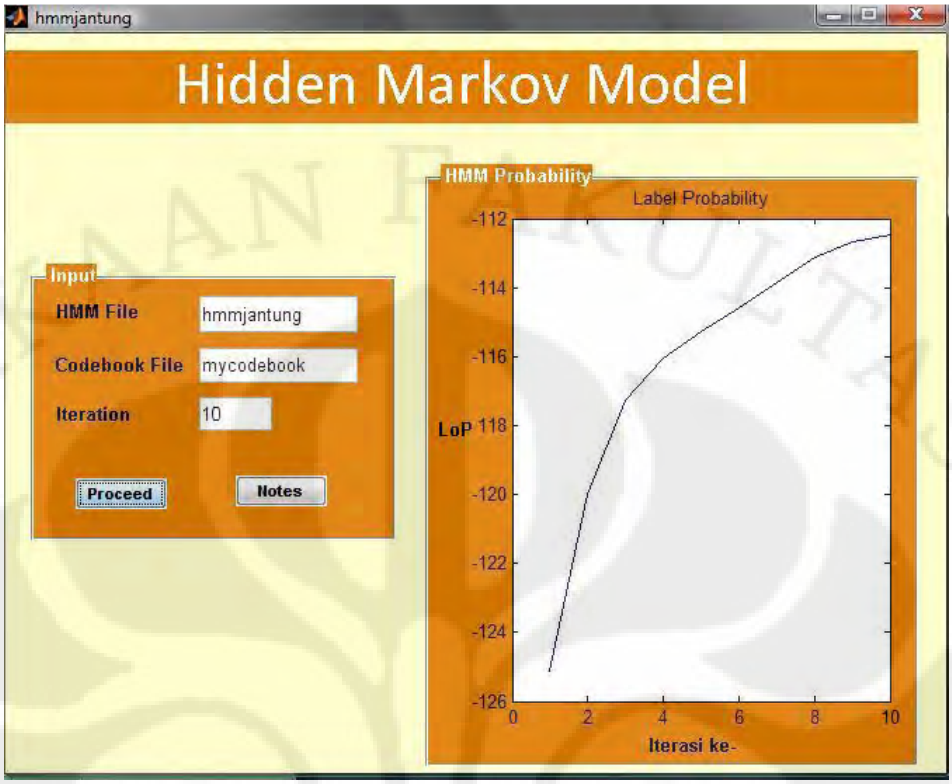
Setelah memasukkan *input-input* yang diperlukan dan mengeksekusi program, maka akan dihasilkan keluaran berupa grafik dan *file* matriks berekstensi ".mat". Grafik ini merupakan tampilan pemetaan titik-titik vektor untuk semua sampel yang telah terdaftar pada proses pelabelan dan posisi *codeword* yang dicari. Sedangkan *file* matriks terdiri dari dua macam, yaitu matriks kode (*matrix codes*) dan matriks nama (*matrix names*). Matriks kode berisi nilai-nilai (posisi) *codeword* untuk masing-masing label dan matriks nama berisi nama-nama penyakit untuk setiap label.

3.1.3 Pembentukan Parameter HMM

Proses ini bertujuan untuk mencari parameter-parameter HMM yang dibutuhkan dalam proses pengenalan. Untuk mendapatkan parameter-parameter tersebut, dibutuhkan suatu masukan yang dikenal sebagai *state* dalam HMM. Keluaran dari proses pembuatan *codebook* yang berupa nilai-nilai (posisi) *centroid*, merupakan *state* bagi proses ini. *Centroid* ini akan membentuk suatu urutan yang mewakili urutan penggalan masing-masing sampel. Urutan *centroid* inilah yang dijadikan urutan *state* dalam pembentukan parameter HMM. Selanjutnya adalah melakukan proses pembelajaran HMM yaitu perhitungan *log of probability* (LoP) untuk 10 iterasi pada tiap-tiap label. Berikut ini adalah algoritma proses pembentukan parameter HMM :

```
Untuk i = 1 sampai banyaknya label
        hitung jumlah centroid;
        state = jumlah centroid;
        hitung nilai probabilitas transisi;
        hitung nilai probabilitas kemunculan state;
        hitung nilai probabilitas observasi;
kembali
hitung nilai Log of probability tiap-tiap label;
```

Sedangkan tampilan program untuk proses pembentukan parameter HMM dapat dilihat pada gambar 3.5.



Gambar 3.5 Tampilan *software* pembentukan parameter HMM

Pada software tersebut terdapat tiga buah masukan, yaitu :

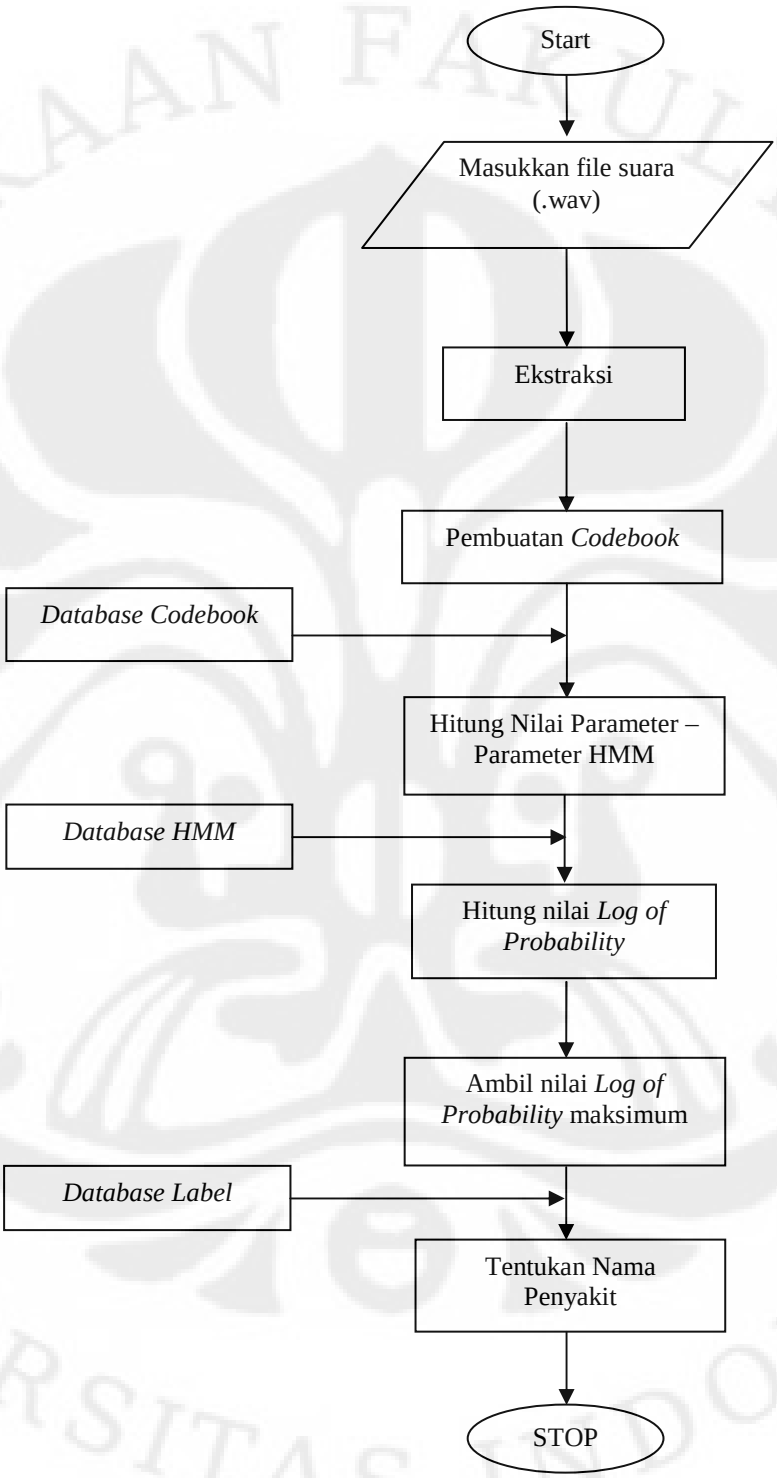
1. *HMM File* diisi dengan nama file HMM yang diinginkan.
2. *Codebook File* diisi dengan nama file *codebook* yang dihasilkan oleh proses sebelumnya yang akan diekstrak.
3. *Iteration* diisi dengan jumlah iterasi yang diinginkan, direkomendasikan sebanyak 10 kali.

Setelah pengeksekusian program, maka akan menghasilkan keluaran berupa grafik nilai LoP terhadap iterasi yang telah dilakukan. Nilai-nilai ini akan tersimpan dalam bentuk matriks dengan nama *file* yang telah diisi pada *HMM File*.

3.2 Proses Pengenalan

Proses pengenalan merupakan bagian terpenting dari seluruh simulasi pengenalan penyakit jantung ini. Sebagian langkah-langkah yang dilakukan pada proses ini mirip dengan langkah-langkah yang terdapat pada proses pengenalan *database* yang telah dilakukan sebelumnya, hanya saja pada bagian pengenalan

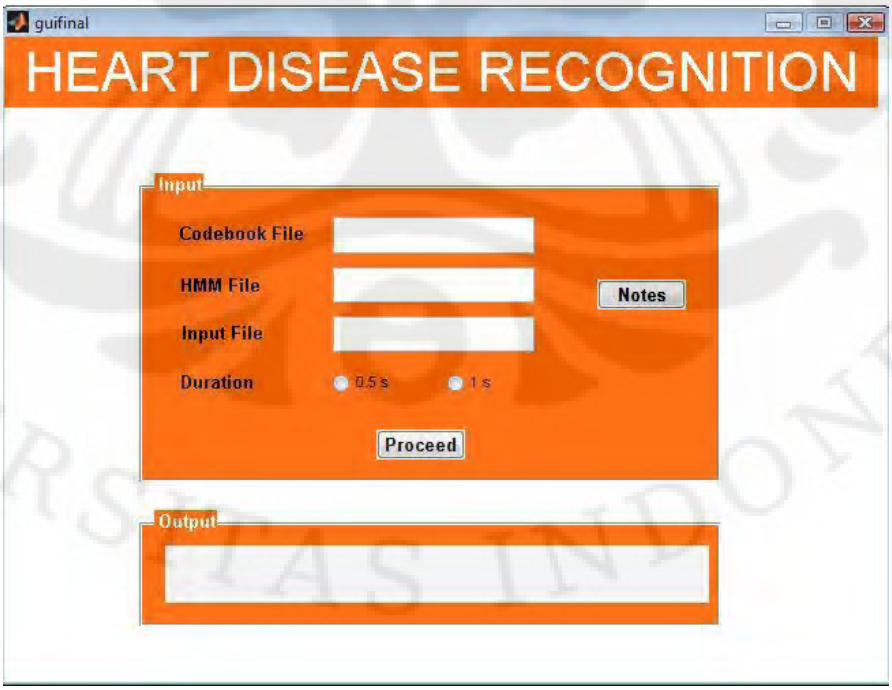
tidak lagi dilakukan proses pembelajaran. Gambar 3.6 menunjukkan diagram alir proses pengenalan.



Gambar 3.6 Diagram alir proses pengenalan penyakit jantung

File suara jantung baru (dalam format ".wav") yang ingin diidentifikasi, dibaca sebagai masukan dari program. *File* input ini akan diekstraksi menjadi beberapa sampel dan selanjutnya akan dikonversi ke dalam domain frekuensi dengan transformasi FFT yang menghasilkan vektor-vektor data. Vektor-vektor data ini seolah-olah dipetakan pada *codebook* yang sama dengan *database codebook* yang telah dilatih sebelumnya karena pada *database*, letak *centroid*-nya telah presisi. Oleh karena itu, apabila vektor-vektor data sampel baru ini dilakukan proses demikian, maka dapat ditentukan vektor-vektor data ini lebih dekat ke *centroid* mana pada *database*. Sehingga dapat ditentukan letak *centroid* atau *codeword*-nya. Setelah nilai dan posisi *centroid* pada sampel diketahui, maka dapat ditentukan kombinasi urutan *centroid* sebagai urutan *state* yang nantinya akan digunakan untuk menentukan parameter-parameter HMM.

Dari beberapa parameter HMM ini (yang telah dijelaskan dalam Bab 2), akan ditentukan nilai *Log of Probability* dari sampel terhadap label-label pada *database*, sehingga akan didapat beberapa nilai *Log of Probability* sebanyak jumlah labelnya. Nilai LoP yang paling tinggi merupakan karakteristik yang mewakili sampel yang kemudian menentukan jenis penyakit sebagai hasil keluaran dari program.



Gambar 3.7 Tampilan program pengenalan penyakit jantung

Pada tampilan proses pengenalan ini, terdapat tiga jenis masukan, yaitu :

1. *HMM File* diisi dengan nama *file* HMM hasil dari proses sebelumnya.
2. *Codebook File* diisi dengan nama *file codebook* hasil dari proses sebelumnya.
3. *Input File* diisi dengan nama *file* sampel yang akan di ujicoba (dalam bentuk ".wav").
4. *Duration* menunjukkan lamanya durasi gelombang suara sampel.

Untuk mengeksekusi program ini, tekan *proceed* dan keluaran dari program akan muncul pada layar. Keluaran inilah berupa nama penyakit yang merupakan keluaran akhir dari simulasi pengenalan penyakit jantung berdasarkan metode HMM.

Algoritma Proses Pengenalan

```
mulai
     baca file sampeljantung.wav;
     untuk i = 1 sampai jumlah penggal
          ekstraksi sampeljantung.wav;
          hitung FFT sebanyak h;
          cari centroid sampeluji berdasarkan database ;
     definisikan urutan centroid sebagai state HMM;
     untuk h = 1 sampai jumlah_label
          Hitung parameter-parameter HMM berdasarkan
          database;
          Hitung log of probability (LoP) untuk semua label;
          LoP [jumlah_label] = tertinggi
          Kembali
     Kembali
     Ambil nilai LoP tertinggi untuk satu label ;
     Nama penyakit = Nama Label;
selesai
```

BAB IV

UJI COBA DAN ANALISIS PERANGKAT LUNAK PENGENALAN PENYAKIT JANTUNG DENGAN METODE HMM

4.1 Daftar Penyakit dan Jenis Percobaan

Pada penelitian ini, uji coba hanya dapat dilakukan pada penyakit jantung yang dapat dideteksi melalui suara detak jantung saja. Oleh karena itu, penyakit jantung yang akan dijadikan bahan uji coba adalah jenis kebocoran dan penyempitan pada beberapa katup jantung. Penyakit tersebut terdiri dari 4 (empat) jenis, yang dapat dilihat pada tabel 4.1.

Tabel 4.1 Nama File dan Jenis Penyakit Jantung

No.	Jenis Penyakit	Nama File
1.	*Aortic Regurgitation*	AR11 – AR20
2.	*Aortic Stenosis*	AS11 – AS20
3.	*Mitral Regurgitation*	MR11 – MR20
4.	*Mitral Stenosis*	MS11 – MS20

Uji coba pada penelitian ini dilakukan dengan memvariasikan tiga buah parameter yaitu durasi tiap sampel, ukuran *codebook*, dan jumlah *database* pelatihan. Secara lengkap uji coba yang dilakukan meliputi :

1. Uji coba sampel untuk durasi 0,7s dengan ukuran *codebook* 32 dan jumlah *database* 5 buah
2. Uji coba sampel untuk durasi 0,7s dengan ukuran *codebook* 32 dan jumlah *database* 10 buah
3. Uji coba sampel untuk durasi 0,7s dengan ukuran *codebook* 64 dan jumlah d*atabase* 5 buah
4. Uji coba sampel untuk durasi 0,7s dengan ukuran *codebook* 64 dan jumlah *database* 10 buah
5. Uji coba sampel untuk durasi 0,7s dengan ukuran *codebook* 128 dan jumlah *database* 5 buah
6. Uji coba sampel untuk durasi 0,7s dengan ukuran *codebook* 128 dan jumlah *database* 10 buah

7. Uji coba sampel untuk durasi 1,5s dengan ukuran *codebook* 32 dan jumlah *database* 5 buah
8. Uji coba sampel untuk durasi 1,5s dengan ukuran *codebook* 32 dan jumlah *database* 10 buah
9. Uji coba sampel untuk durasi 1,5s dengan ukuran *codebook* 64 dan jumlah *database* 5 buah
10. Uji coba sampel untuk durasi 1,5s dengan ukuran *codebook* 64 dan jumlah *database* 10 buah
11. Uji coba sampel untuk durasi 1,5s dengan ukuran *codebook* 128 dan jumlah *database* 5 buah
12. Uji coba sampel untuk durasi 1,5s dengan ukuran *codebook* 128 dan jumlah *database* 10 buah

4.2 Hasil Uji Coba

Setelah melakukan pengujian terhadap sampel-sampel yang telah ditentukan beserta variasi parameter-parameter yang telah dijelaskan sebelumnya, maka didapatkan data hasil pengujian tersebut. Warna merah menunjukkan program "salah" mendeteksi penyakit dan warna hitam menunjukkan program "benar" mendeteksi penyakit.

Tabel 4.2 Hasil uji coba penyakit *Aortic Regurgitation* untuk durasi 1,5s

No.	*Codebook* / Repetisi / Nama File	32		64		128	
		5	10	5	10	5	10
1.	AR11	AR	AR	AR	AR	AR	AR
2.	AR12	MR	MR	MR	AR	MS	AR
3.	AR13	AR	AR	AR	AR	AR	AR
4.	AR14	AR	AR	AR	AR	AR	AR
5.	AR15	AR	AR	AR	AR	AR	AR
6.	AR16	MR	MR	MR	MS	MS	MS
7.	AR17	MS	AS	AS	AS	AR	AS
8.	AR18	AR	AR	AR	AR	AR	AR
9.	AR19	AR	AR	AR	AR	AR	AR
10.	AR20	AR	AR	AR	AR	AR	AR

Tabel 4.3 Hasil uji coba penyakit *Aortic Stenosis* untuk durasi 1,5s

No.	*Codebook* / Repetisi / Nama File	32		64		128	
		5	10	5	10	5	10
1.	AS11	AS	AS	AS	AS	AS	AS
2.	AS12	AS	AS	AS	AS	AS	AS
3.	AS13	MR	AS	MR	MR	MR	MR
4.	AS14	AS	AS	AS	AS	AS	AS
5.	AS15	AR	AR	AR	AR	AR	AR
6.	AS16	AS	AS	AS	AS	AS	AS
7.	AS17	AS	AS	AS	AS	AS	AS
8.	AS18	MS	AS	MS	AS	MS	AS
9.	AS19	AS	MR	AS	AS	AS	AS
10.	AS20	AS	AS	AS	AS	AS	AS

Tabel 4.4 Hasil uji coba penyakit *Mitral Regurgitation* untuk durasi 1,5s

No.	*Codebook* / Repetisi / Nama File	32		64		128	
		5	10	5	10	5	10
1.	MR11	AS	MR	AS	MR	AR	MR
2.	MR12	AR	AR	MR	MR	MR	AR
3.	MR13	MS	MR	MS	MR	AR	MR
4.	MR14	MR	MR	MR	MR	MR	MR
5.	MR15	MR	MR	MR	MR	MR	MR
6.	MR16	MR	MR	MR	MR	MR	MR
7.	MR17	MR	MR	MR	MR	MR	MR
8.	MR18	MR	MR	MR	MR	MR	MR
9.	MR19	MR	MR	MR	MR	MR	MR
10.	MR20	MR	MR	MR	MR	MS	MR

Tabel 4.5 Hasil uji coba penyakit *Mitral Stenosis* untuk durasi 1,5s

No.	*Codebook* / Repetisi / Nama File	32		64		128	
		5	10	5	10	5	10
1.	MS11	AS	AS	MR	AR	MS	AR
2.	MS12	MS	AR	MS	MS	MS	MS
3.	MS13	AS	MS	MR	MR	MS	MS
4.	MS14	MS	MS	MR	MS	MS	MS
5.	MS15	AS	MS	MS	AS	AS	MS
6.	MS16	MS	MS	MS	MS	MS	MS
7.	MS17	AS	MS	MS	MS	AR	MS
8.	MS18	MS	MR	MS	MS	MS	MS
9.	MS19	MR	MS	MR	MS	MS	MS

10.	MS20	MS	MS	MS	MS	MS	MS

Tabel 4.6 Hasil uji coba penyakit *Aortic Regurgitation* untuk durasi 0,7s

No.	*Codebook*	32		64		128	
	Repetisi / Nama File	5	10	5	10	5	10
1.	AR11	AR	MS	AR	AR	AR	AR
2.	AR12	MS	AR	MS	MR	MS	MS
3.	AR13	AR	AR	AR	AR	AR	AR
4.	AR14	AR	AR	AR	AR	AR	AR
5.	AR15	AR	MR	AR	AR	AR	AR
6.	AR16	MS	MS	MS	MS	MS	MS
7.	AR17	AS	AR	AS	AS	AS	AR
8.	AR18	AR	AR	AR	AR	AR	AR
9.	AR19	MR	AR	AR	AR	AR	AR
10.	AR20	AR	AR	AR	AR	AR	AR

Tabel 4.7 Hasil uji coba penyakit *Aortic Stenosis* untuk durasi 0,7s

No.	*Codebook*	32		64		128	
	Repetisi / Nama File	5	10	5	10	5	10
1.	AS11	AS	AS	AS	AS	AS	AS
2.	AS12	AS	AS	AS	AS	AS	AS
3.	AS13	MR	MR	MR	AS	MR	AS
4.	AS14	AR	AS	AS	AS	AS	AR
5.	AS15	MS	AR	AR	AR	AR	AR
6.	AS16	AS	AS	AR	AS	AS	AS
7.	AS17	AS	AS	AS	AS	AS	AS
8.	AS18	MS	AS	AS	AS	AS	AS
9.	AS19	AS	AS	AS	AS	AS	AS
10.	AS20	MR	AS	AS	AS	AS	AS

Tabel 4.8 Hasil uji coba penyakit *Mitral Regurgitation* untuk durasi 0,7s

No.	*Codebook*	32		64		128	
	Repetisi / Nama File	5	10	5	10	5	10
1.	MR11	MR	MR	AR	MR	AR	MR
2.	MR12	MR	MR	MR	MR	MR	MR
3.	MR13	MS	MR	AR	MR	AR	MR
4.	MR14	MS	MS	MS	MR	MS	MR
5.	MR15	MR	MR	MR	MR	MR	MR
6.	MR16	MR	MR	MR	MR	MR	MR
7.	MR17	MR	MR	MR	MR	MR	MR

8.	MR18	MR	MS	MR	MR	MR	MR
9.	MR19	AS	MR	MR	MR	MR	MR
10.	MR20	MS	MS	MR	MS	MS	AR

Tabel 4.9 Hasil uji coba penyakit *Mitral Stenosis* untuk durasi 0,7s

No.	*Codebook* / Repetisi / Nama File	32		64		128	
		5	10	5	10	5	10
1.	MS11	MR	MS	MR	MR	MR	AS
2.	MS12	MS	AR	MS	MS	MS	MS
3.	MS13	AS	AS	MR	MS	MS	MS
4.	MS14	MS	MS	MR	MS	MS	MS
5.	MS15	AS	MS	MS	MS	MS	MS
6.	MS16	MS	MS	MS	MS	MS	MS
7.	MS17	MS	MS	MS	MS	MS	AS
8.	MS18	MS	MS	MS	MS	MS	MS
9.	MS19	MS	MS	MS	MS	MS	MS
10.	MS20	MS	MS	MS	MS	MS	MS

4.3 Persentase Akurasi

Berikut ini adalah persentase akurasi program terhadap berdasarkan hasil uji coba yang telah diperoleh.

Tabel 4.10 Persentase akurasi semua penyakit untuk durasi 1,5s

Parameter / Penyakit	Codebook 32		Codebook 64		Codebook 128	
	5	10	5	10	5	10
AR	70%	70%	70%	80%	80%	80%
AS	70%	80%	70%	80%	70%	80%
MR	70%	90%	80%	100%	70%	90%
MS	50%	70%	60%	70%	80%	90%

Tabel 4.11 Persentase akurasi semua parameter untuk durasi 1,5s

Codebook / Repetisi	32	64	128
5	65%	70%	75%
10	77,5%	82,5%	85%

Tabel 4.12 Persentase akurasi semua penyakit untuk durasi 0,7s

Parameter	Codebook 32		Codebook 64		Codebook 128	
	5	10	5	10	5	10
AR	60%	70%	70%	70%	70%	80%
AS	50%	80%	70%	90%	80%	80%
MR	60%	70%	70%	90%	60%	100%
MS	70%	80%	70%	90%	90%	80%

Tabel 4.13 Persentase akurasi semua parameter untuk durasi 0,7s

Codebook / Repetisi	32	64	128
5	60%	65%	75%
10	75%	85%	85%

4.4 Analisis Hasil Percobaan

Pada sistem pengenalan ini dibandingkan antara gelombang yang ingin diidentifikasi dengan gelombang yang ada pada *database*. Hanya ada dua kondisi yang mungkin keluar sebagai hasil identifikasi, yaitu :

1. Sistem dapat mengidentifikasi penyakit jantung

   Kondisi ini terjadi jika hasil pengenalan sesuai dengan masukan. *Label* atau jenis penyakit dengan probabilitas tertinggi merupakan jenis gelombang yang sesuai dengan masukan.

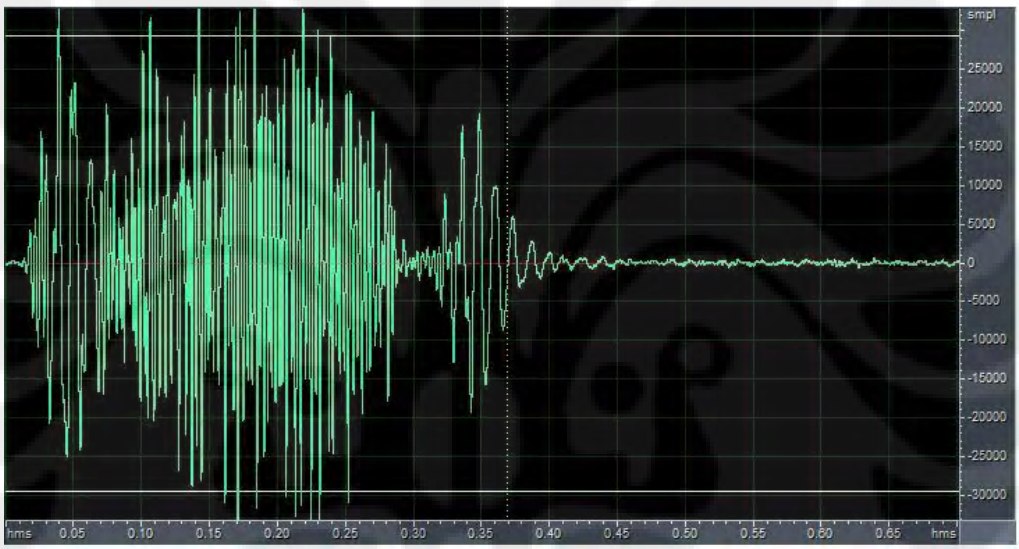
2. Sistem salah mengindentifikasi penyakit jantung

   Kondisi ini terjadi jika hasil pengenalan berbeda dengan masukan. *Label* atau jenis penyakit dengan probabilitas tertinggi bukan merupakan jenis gelombang yang sesuai dengan masukan. Dalam hal ini, sistem akan menjadikan *label* dengan nilai *log of probability* yang tertinggi sebagai hasil pengenalan.

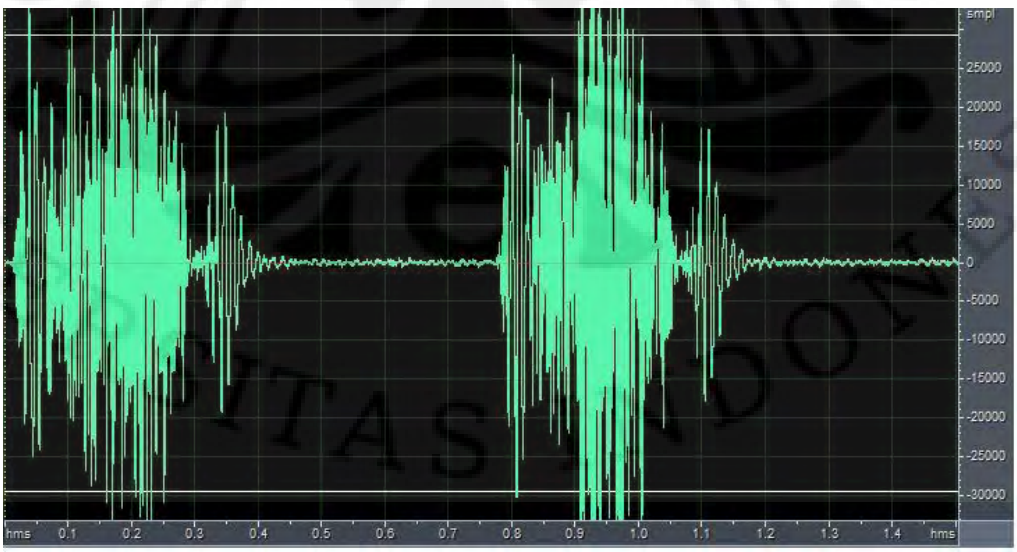
Keberhasilan atau tidaknya sistem dalam mengidentifikasi gelombang perubahan fase tergantung dari beberapa faktor, diantaranya adalah ukuran *codebook*, jumlah *database* dan durasi sinyal.

4.4.1 Analisis Pengaruh Variasi Durasi Sampel

Pada penelitian ini, durasi sampel ditentukan berdasarkan siklus *murmur* tiap sampel yang diproses baik sebagai sampel uji maupun sebagai *database. Murmur* ini memiliki siklus yang berulang dimana masing-masing siklus memiliki rentang waktu yang disebut dengan durasi sampel. Berdasarkan bentuk-bentuk gelombang suara detak jantung yang diperoleh, maka dalam penelitian ini digunakan durasi sampel 0,7 detik dan 1,5 detik. Durasi sampel 0,7 detik secara umum mewakili satu siklus murmur dan durasi sampel 1,5 detik mewakili dua siklus murmur untuk masing-masing penyakit. Berikut ini adalah contoh sampel penyakit *Aortic Stenosis* untuk masing-masing durasi sampel.



Gambar 4.1 Gelombang suara *Aortic Stenosis* dengan durasi 0,7 detik



Gambar 4.2 Gelombang suara *Aortic Stenosis* dengan durasi 1,5 detik

Perubahan durasi sampel memberikan pengaruh terhadap perubahan jumlah *frame* tiap-tiap sampel berdasarkan persamaan (2.2). Semakin besar durasi sampel yang ditentukan, semakin banyak jumlah *frame* yang dihasilkan. Hal ini mengakibatkan semakin banyaknya jumlah titik-titik vektor data yang diproses sehingga memakan waktu lebih lama. Namun, semakin tinggi nilai durasi sampel, maka semakin banyak jumlah titik vektor data yang dihasilkan. Sehingga jarak antara vektor data dengan *centroid* menjadi semakin dekat dan lebih teliti dibandingkan dengan nilai durasi sampel yang lebih kecil ([1]-[4]).

Kesalahan pendeteksian penyakit dapat dipengaruhi parameter durasi sampel ini. Hal ini dikarenakan terdapat beberapa sampel yang tidak memenuhi 1 (satu) siklus murmur untuk durasi sampel 0,7 detik dan 2 (dua) siklus murmur untuk durasi sampel 1,5 detik. Hal ini dapat menyebabkan sinyal tersebut terpotong dan mungkin membentuk suatu potongan sampel yang mirip antara satu penyakit dengan penyakit yang lain, sehingga dapat terjadi kesalahan pendeteksian penyakit.

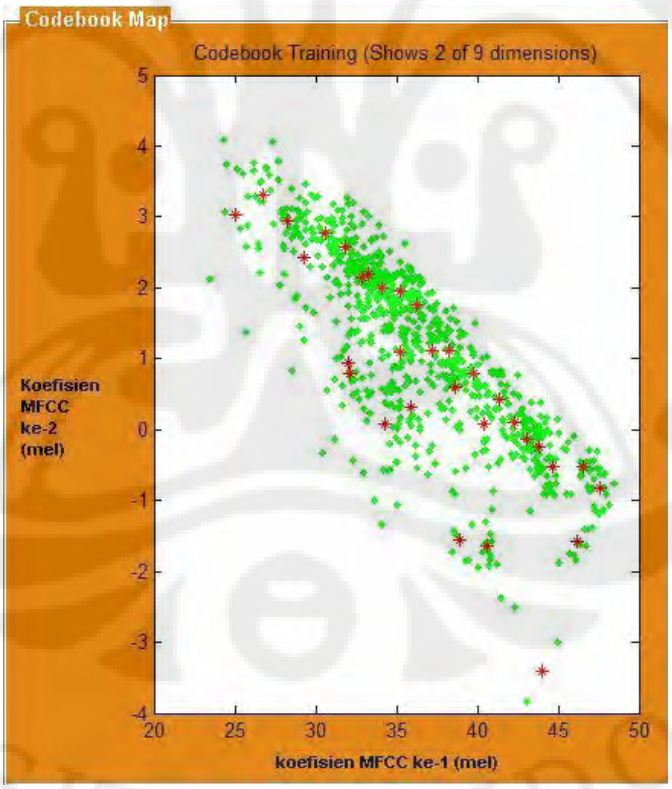
Berdasarkan Tabel 4.10 sampai 4.13, terlihat bahwa peningkatan nilai durasi sampel tidak terlalu berpengaruh terhadap peningkatan akurasi. Durasi sampel 1,5 detik memiliki kelebihan, yaitu walaupun dengan jumlah sampel sedikit, persentase akurasi yang dicapai lebih baik dibandingkan durasi sampel 0,7 detik dengan jumlah sampel yang sama. Sedangkan durasi sampel 0,7 detik memiliki persentase akurasi rata-rata yang lebih tinggi untuk sampel yang lebih banyak dibandingkan durasi sampel 1,5 detik.

Persentase akurasi pada durasi sampel 0,7 detik mengalami peningkatan yang lebih signifikan dibanding durasi sampel 1,5 detik. Selain itu, durasi sampel 0,7 detik diproses dengan waktu yang lebih singkat dibanding durasi sampel 1,5 detik. Hal ini dikarenakan durasi sampel 0,7 detik sudah cukup mewakili siklus *murmur* yang akan diidentifikasi penyakitnya sehingga akurasi yang dicapai lebih tinggi untuk jumlah database yang lebih banyak dan waktu pemrosesan jauh lebih singkat dibandingkan dengan durasi sampel 1,5 detik. Sehingga menunjukkan bahwa durasi sampel 0,7 detik lebih efisien dibandingkan dengan durasi ampel 1,5 detik.

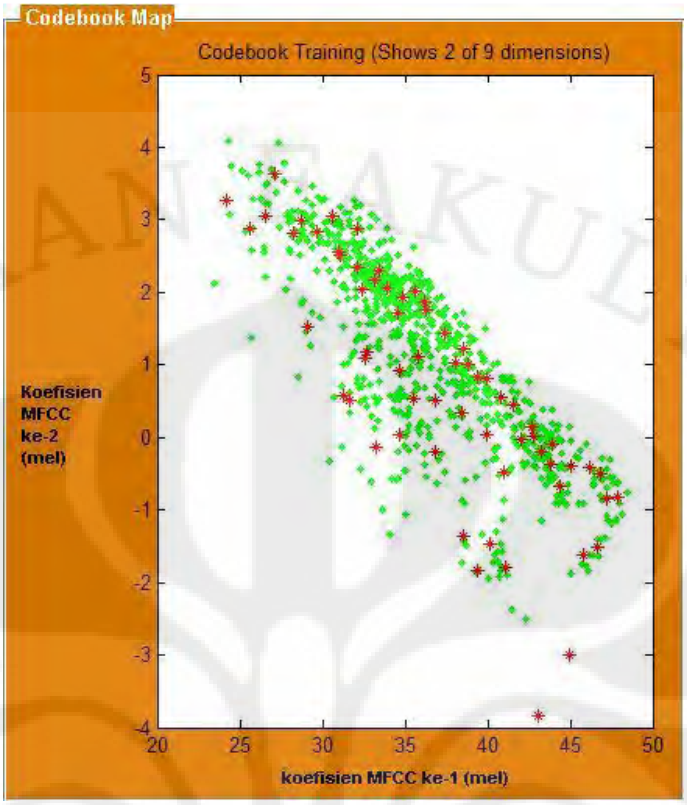
4.4.2 Analisis Pengaruh Variasi Ukuran *Codebook*

Pengaruh ukuran *codebook* dapat dilihat pada Tabel 4.10 sampai Tabel 4.13 Secara keseluruhan, semakin besar ukuran *codebook*, maka semakin besar pula tingkat keberhasilan program dalam mendeteksi penyakit.

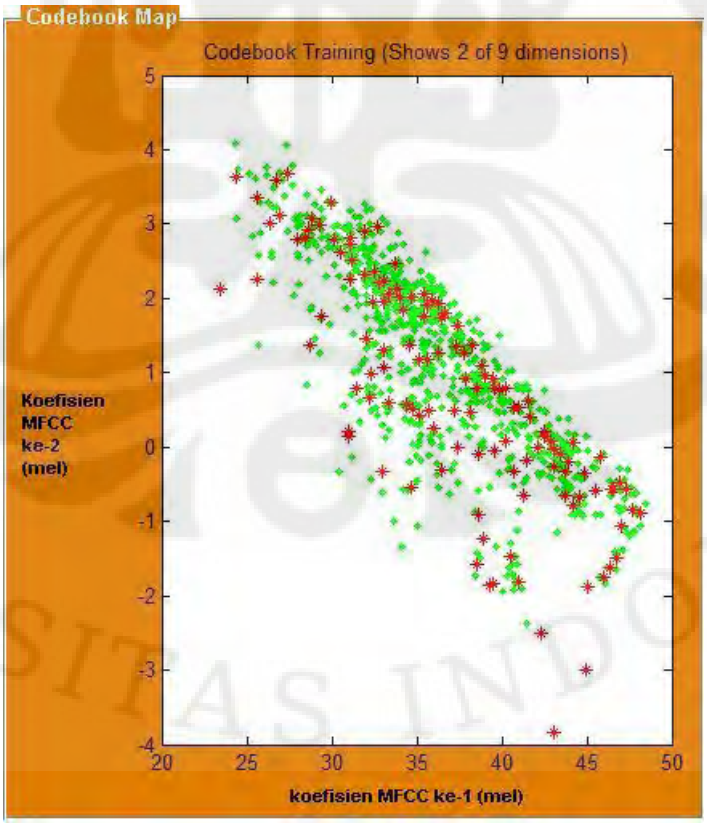
Dengan meningkatnya ukuran *codebook*, maka jumlah *cluster* yang terbentuk semakin banyak, sehingga terbentuk *centroid* yang semakin banyak pula. Semakin banyak jumlah *centroid*, maka panjang *codeword* akan semakin besar dan tentunya *codeword* ini akan semakin mewakili sinyal masukan aslinya. Selain itu, ukuran *cluster* akan semakin kecil yang dapat menyebabkan sistem mengkuantisasi vektor data lebih teliti karena jarak antar-vektor data serta antara vektor data dengan *centroid* semakin kecil. Untuk lebih jelasnya, dapat dilihat pada gambar 4.3 – 4.5 berikut ini.



Gambar 4.3 *Codebook* 32 bit



Gambar 4.4 *Codebook* 64 bit



Gambar 4.5 *Codebook* 128 bit

Berdasarkan hasil data percobaan secara keseluruhan, terlihat bahwa semakin besar ukuran *codebook*, maka tingkat akurasi program semakin meningkat. Namun, terdapat kondisi dimana tingkat akurasi tidak mengalami kenaikan maupun penurunan. Hal ini disebabkan karena sinyal suara memiliki rentang data yang cukup besar, sehingga cukup sulit untuk membandingkan data-data yang secara kasat mata terlihat mirip. Selain itu, dari gambar di atas juga terlihat bahwa terlihat beberapa *centroid* yang tidak melingkupi daerah persebaran vektor data, yang menyebabkan beberapa *centroid* tersebut tidak mewakili sinyal masukan ([1]-[4]).

Namun, walaupun dengan nilai persentase akurasi yang sama ataupun meningkat, kesalahan pendeteksian tidak sama untuk setiap penyakit baik untuk durasi sampel 0,7 s maupun durasi sampel 1,5 s. Hal ini terlihat pada Tabel 4.2 sampai Tabel 4.9. Hal ini disebabkan karena perubahan ukuran *codebook*, maka berubah pula jumlah *centroid* sehingga walaupun jenis penyakitnya sama, urutan *centroid* (*codeword*) yang dibentuk untuk tiap *codebook* berbeda. Oleh karena itu, meskipun dengan masukan penyakit yang sama, maka urutan *centroid* yang terbentuk akan berbeda untuk ukuran *codebook* yang berbeda dan pendeteksian penyakit dapat berbeda pula.

Selain itu, semakin besar ukuran *codebook*, semakin lama pula waktu pemrosesannya karena semakin banyak jumlah *centroid* yang akan dicari. Oleh karena itu pada penelitian ini hanya digunakan ukuran *codebook* 32, 64 dan 128. Menurut beberapa penelitian sinyal suara [1] [2],untuk *codebook* dengan ukuran lebih besar dari 128, hasil yang didapatkan tidak terlalu berbeda dengan hasil yang didapatkan pada uji coba dengan ukuran *codebook* 128. Namun waktu pemrosesan menjadi jauh lebih lama sehingga menjadi tidak efisien. Oleh karena itulah hanya digunakan *codebook* berukuran 32, 64, dan 128 dalam pengujian.

Ketika sampel baru diuji dan akan dicari letak *centroid*-nya, maka pencarian tersebut mengacu kepada letak *centroid* yang terdapat pada database *codebook*. Pada jenis penyakit yang sama (tapi beda sampel) pun dapat terjadi kesalahan pengenalan penyakitnya. Hal ini dikarenakan distorsi yang dihasilkan pada setiap sampel tidak selalu sesuai dengan distorsi yang terdapat pada *database* untuk penyakit yang sama. Sehingga terdapat kemungkinan letak sampel baru ini lebih

mendekati letak *centroid* jenis penyakit lainnya sehingga kombinasi urutan centroid yang dihasilkan agak berbeda dengan penyakit yang sebenarnya.

Penurunan nilai akurasi sistem pada penelitian ini tidak dapat dihindari yang dapat dilihat pada tabel 4.10 dan tabel 4.12. Dengan jumlah *database* yang sama, nilai akurasi menurun seiring dengan semakin besarnya nilai *codebook*. Hal ini disebabkan karena adanya kemiripan karakteristik penyakit jantung antara yang satu dengan lainnya untuk label yang berbeda. Sehingga *codeword* hasil uji coba untuk ukuran *codebook* semakin besar semakin mendekati *codeword* dari sampel penyakit untuk label lain. Tetapi hal tersebut tidak terlalu berpengaruh karena secara keseluruhan, semakin besar ukuran *codebook* semakin besar pula nilai akurasinya yang dapat dilihat pada tabel 4.11 dan tabel 4.13.

4.4.3 Analisis Pengaruh Variasi Jumlah *Database*

Parameter berikutnya yang berpengaruh terhadap hasil uji coba adalah jumlah *database*. Secara keseluruhan, semakin besar jumlah *database,* maka semakin tinggi tingkat akurasi yang dicapai. Hal ini dapat dilihat pada Tabel 4.10 sampai 4.13. Pada penelitian ini, jumlah sampel yang dijadikan *database* divariasikan sebanyak 5 (lima) dan 10 (sepuluh) buah sampel. Penggunaan jumlah *database* yang sedikit ini disebabkan karena keterbatasan sampel yang ada.

Semakin banyak sampel yang dimasukkan ke dalam *database* untuk masing-masing penyakit, maka proses *recognition* akan lebih mudah karena dengan banyaknya *database* untuk masing-masing penyakit, masalah kemiripan atau perbedaan satu penyakit terhadap penyakit lainnya akan berkurang. Karena jumlah *database* yang lebih banyak maka secara tidak langsung akan menambah karakteristik label yang tersimpan pada *database.* Dengan demikian, data yang dimasukkan lebih banyak, maka karakteristik yang tadinya belum ada pada saat proses pembentukan *database* dilakukan dengan jumlah database yang lebih kecil, akan terpenuhi dengan jumlah *database* yang lebih besar ([1]-[4]).

Dengan banyaknya variasi data untuk masing-masing penyakit pada database, maka letak *centroid* untuk tiap-tiap penyakit juga akan semakin bervariasi, sehingga penyakit-penyakit yang mirip sekalipun akan memiliki letak

centroid yang berbeda koordinatnya, tidak terlalu berdekatan apalagi berhimpit. Dengan demikian proses pengenalan akan semakin mudah dan tingkat keberhasilan yang dicapai akan semakin tinggi.

4.4.4 Analisis Pengaruh Hubungan Ukuran *Codebook*, Jumlah *Database* dan Durasi Sampel

Untuk mencapai kondisi sistem pengenalan penyakit jantung yang efisien, maka diperlukan kombinasi parameter-parameter untuk membentuk kinerja sistem yang optimal. Setelah dilakukan uji coba terhadap sampel-sampel pada variasi ukuran *codebook*, jumlah *database* dan nilai durasi sampel, maka dapat dilihat pada Tabel 4.2 – Tabel 4.13 variasi parameter yang mana yang memberikan hasil yang optimal.

Seperti yang telah dijelaskan sebelumnya, semakin besar ukuran *codebook* yang digunakan, maka kinerja sistem menjadi lebih optimal. Jika dihubungkan dengan parameter tersebut, maka semakin banyak jumlah *database*, semakin banyak pula vektor data yang mengalami proses vektor kuantisasi. Hal ini sangat berpengaruh pada waktu pemrosesan pembuatan *codebook* yang menjadi semakin bertambah. Namun, hal ini tidak menjadi suatu kendala karena jumlah *database* yang semakin banyak akan berpengaruh positif terhadap sistem. Jika dihubungkan lagi dengan parameter durasi sampel, maka durasi sampel yang optimal adalah durasi sampel yang bernilai paling kecil, tapi harus memenuhi batas minimum persyaratan *input* gelombang yang dibutuhkan.

Berdasarkan tabel 4.11 dan tabel 4.13, terdapat nilai maksimum akurasi sebesar 85%. Namun, karena terdapat nilai akurasi yang sama untuk tiga parameter, maka ukuran *codebook* yang optimal pada penelitian ini adalah 64, Jumlah *database* yang optimal sebesar 10 (sepuluh) buah, dan rentang waktu sampel yang optimal adalah 0,7 detik. Jika dibandingkan dengan kedua parameter lainnya, parameter ini membutuhkan waktu pemrosesan data yang lebih cepat sehingga sistem lebih efisien.

BAB V

KESIMPULAN

Berdasarkan hasil uji coba dan analisis yang telah dilakukan, maka dapat diambil beberapa kesimpulan antara lain sebagai berikut.

1. Nilai akurasi keseluruhan yang dicapai adalah antara 60% sampai 85%.
2. Semakin besar ukuran *codebook*, maka semakin besar tingkat akurasi yang dicapai sistem dalam mendeteksi penyakit jantung.
3. Semakin besar jumlah *database*, semakin besar pula tingkat keberhasilan sistem dalam mendeteksi penyakit jantung.
4. Durasi sampel yang optimal adalah durasi sampel yang bernilai paling kecil, tapi harus memenuhi batas minimum persyaratan masukan gelombang suara yang dibutuhkan.
5. Pada penelitian ini, ukuran *codebook* yang optimal adalah 64, jumlah *database* yang optimal sebesar 10 (sepuluh) buah, dan rentang waktu sampel yang optimal adalah 0,7 detik.

DAFTAR ACUAN

[1] Alfarisi, Lutfie Salman. "Speech Recognition dengan Hidden Markov Model menggunakan DSP Starter Kit TMS320C6713". Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2007.

[2] Amin, Ahmad Mujadid. "Simulasi dan Analisis Program Speech To Text Menggunakan Metode Hidden Markov Model". Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2007.

[3] Lestari, Afita Putri. "Rancang Bangun Pengenalan Penyakit Darah Menggunakan Hidden Markov Model". Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2008.

[4] Diponegoro, Arman Djohan. "Analisis penentuan jenis kawanan ikan berdasarkan deteksi fasa pantulan gelombang akustik dan penerapan Hidden Markov Model". 2006. Disertasi, Program Pascasarjana Fakultas Ilmu Kelautan dan Perikanan Institut Pertanian Bogor, Bogor, 2006.

[5] "_____", *Cara Kerja Jantung*, diakses Maret 2009. http://jantung.klikdokter.com/subpage.php?id=1&sub=67

[6] "_____", *Regurgitasi Katup Aorta (Inkompetensia Aorta, Insuffisiensi Aorta, Aortic Regurgitation)*, diakses Maret 2009. http://medicastore.com/penyakit/114/Regurgitasi_Katup_AortaInkompetensia_Aorta_Insuffisiensi_Aorta_Aortic_Regurgitation.html

[7] "_____", *Stenosis Katup Aorta (Aortic Stenosis)*, diakses Maret 2009. http://medicastore.com/penyakit/115/Stenosis_Katup_Aorta_Aortic_Stenosis.html

[8] "_____", *Stenosis Katup Mitral*, diakses Maret 2009.

http://medicastore.com/penyakit/113/Stenosis_Katup_Mitral.html

[9] "_____", *Prolaps Katup Mitral (Mitral Valve Prolapse (MVP)*, diakses Maret 2009. http://medicastore.com/penyakit/112/Prolaps_Katup_Mitral_Mitral_Valve_Prolapse_MVP.html

[10] "Zap Andriyana", *Cara Kerja Jantung*, diakses Maret 2009. http://dr-zapra.blogspot.com/2007/12/murmur-jantung-yang-disebabkan-oleh.html

[11] Yul Antonisfia, Romi Wiryadinata. "Ekstraksi Ciri Pada Isyarat Suara Jantung Menggunakan Power Spectral Density Berbasis Metode Welch". Media Informatika, Vol. 6, No. 1, 2008.

[12] "_____", *Jantung*, diakses April 2009. http://wapedia.mobi/id/Jantung?t=3

[13] "_____", *Dasar Teori*, diakses April 2009. http://digilib.petra.ac.id/viewer.php?page=1&submit.x=0&submit.y=0&qual=high&fname=/jiunkpe/s1/info/2005/jiunkpe-ns-s1-2005-26402028-7608-kenal_suara-chapter2.pdf

[14] "_____", *Regurgitasi Katup Mitral*, diakses Februari 2009. http://medicastore.com/penyakit/111/Regurgitasi_Katup_Mitral.html

DAFTAR PUSTAKA

Diponegoro, Arman Djohan. "Analisis penentuan jenis kawanan ikan berdasarkan deteksi fasa pantulan gelombang akustik dan penerapan Hidden Markov Model". 2006. Disertasi, Program Pascasarjana Fakultas Ilmu Kelautan dan Perikanan Institut Pertanian Bogor, Bogor, 2006.

Alfarisi, Lutfie Salman. "Speech Recognition dengan Hidden Markov Model menggunakan DSP Starter Kit TMS320C6713". Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2007.

Amin, Ahmad Mujadid. "Simulasi dan Analisis Program Speech To Text Menggunakan Metode Hidden Markov Model". Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2007.

Lestari, Afita Putri. "Rancang Bangun Pengenalan Penyakit Darah Menggunakan Hidden Markov Model". Skripsi, Program Sarjana Fakultas Teknik Universitas Indonesia, Depok, 2008.

Rabiner, L. R., R. W. Schafer. *Digital Processing of Speech Signal*. Prentice-Hall: New Jersey. 1978.

Rabiner, L. R, B. H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall: New Jersey. 1993.

LAMPIRAN

1. Listing Program Pelabelan

```
     function varargout = pelabelan(varargin)
% PELABELAN M-file for pelabelan.fig
%   PELABELAN, by itself, creates a new PELABELAN or raises the existing
%   singleton*.
%
%   H = PELABELAN returns the handle to a new PELABELAN or the handle to
%   the existing singleton*.
%
%   PELABELAN('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in PELABELAN.M with the given input
%   arguments.
%
%   PELABELAN('Property','Value',...) creates a new PELABELAN or raises
%   the existing singleton*.  Starting from the left, property value
%   pairs are applied to the GUI before pelabelan_OpeningFcn gets called.
%   An unrecognized property name or invalid value makes property
%   application stop.  All inputs are passed to pelabelan_OpeningFcn via
%   varargin.
%
%   *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help pelabelan

% Last Modified by GUIDE v2.5 01-Jun-2009 16:03:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pelabelan_OpeningFcn, ...
                   'gui_OutputFcn',  @pelabelan_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before pelabelan is made visible.
function pelabelan_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pelabelan (see VARARGIN)

% Choose default command line output for pelabelan
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes pelabelan wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = pelabelan_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
    h=guidata(gcbo);
    repetisi=get(h.repetisi,'value');
%     repetisi = str2num(repetisi1);
%     repetisi = 35;
    if get(h.radiobutton1,'value') == 1
        time = 0.7
    else
        time = 1.5
    end
    make_labels(repetisi,time);
%frekuensisampling menggunakan 8000Hzdan waktu yang diambil hanyalah 1s


function index_label_Callback(hObject, eventdata, handles)
% hObject    handle to index_label (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of index_label as text
%        str2double(get(hObject,'String')) returns contents of
index_label as a double


% --- Executes during object creation, after setting all properties.
function index_label_CreateFcn(hObject, eventdata, handles)
% hObject    handle to index_label (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object deletion, before destroying properties.
function text4_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to text4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
function nama_label_Callback(hObject, eventdata, handles)
% hObject    handle to nama_label (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nama_label as text
%        str2double(get(hObject,'String')) returns contents of nama_label
as a double


% --- Executes during object creation, after setting all properties.
function nama_label_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nama_label (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in repetisi.
function repetisi_Callback(hObject, eventdata, handles)
% hObject    handle to repetisi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns repetisi contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from
repetisi


% --- Executes during object creation, after setting all properties.
function repetisi_CreateFcn(hObject, eventdata, handles)
% hObject    handle to repetisi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%==========================================================================
function make_labels(rep,time)
    h=guidata(gcbo);
    index_label=get(h.index_label,'String');
    index_label=str2num(index_label);
    nama_label=get(h.nama_label,'String');
    etiqueta=nama_label;
    makelabel(['label' int2str(index_label)],etiqueta,rep,time);
%==========================================================================


%==========================================================================
function makelabel(filename,etiqueta,times,time)
%   fs=11025;
```

```
    fs=6000;
    label=zeros(time*6000,times);
    number=1;
    while (number<=times)
       nama_file = [etiqueta int2str(number)];
       etiqueta2 = nama_file;%no11...no15

       %eval(['load' ' ' etiqueta2]);
       %speech =  eval(etiqueta2);

       [speech]=wavread(etiqueta2,time*6000);
       label(:,number)=speech;
       number=number+1;
    end
    eval(['save' ' ' filename ' ' 'label etiqueta'])
    h=guidata(gcbo);
    index_label=get(h.index_label,'String');
    index_label=str2num(index_label);
    index_label = index_label + 1;
    set(h.index_label,'String',index_label);
%==========================================================================

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
helppelabelan


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
h=guidata(gcbo);
set(h.radiobutton2,'value',0);
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
h=guidata(gcbo);
set(h.radiobutton1,'value',0);
% Hint: get(hObject,'Value') returns toggle state of radiobutton2


% --- Executes during object creation, after setting all properties.
function radiobutton1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes during object creation, after setting all properties.
function radiobutton2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

2. Listing Program Pembuatan Codebook

```
function varargout = codebookjantung(varargin)
% CODEBOOKJANTUNG M-file for codebookjantung.fig
%      CODEBOOKJANTUNG, by itself, creates a new CODEBOOKJANTUNG or
raises the existing
%      singleton*.
%
%      H = CODEBOOKJANTUNG returns the handle to a new CODEBOOKJANTUNG or
the handle to
%      the existing singleton*.
%
%      CODEBOOKJANTUNG('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in CODEBOOKJANTUNG.M with the given input
arguments.
%
%      CODEBOOKJANTUNG('Property','Value',...) creates a new
CODEBOOKJANTUNG or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before codebookjantung_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to codebookjantung_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help codebookjantung

% Last Modified by GUIDE v2.5 01-Jun-2009 17:14:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @codebookjantung_OpeningFcn, ...
                   'gui_OutputFcn',  @codebookjantung_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before codebookjantung is made visible.
function codebookjantung_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to codebookjantung (see VARARGIN)

% Choose default command line output for codebookjantung
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes codebookjantung wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = codebookjantung_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


 function ukuran_codebook_Callback(hObject, eventdata, handles)
% hObject    handle to ukuran_codebook (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ukuran_codebook as
text
%        str2double(get(hObject,'String')) returns contents of
ukuran_codebook as a double


% --- Executes during object creation, after setting all properties.
function ukuran_codebook_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ukuran_codebook (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%==========================================================================
function pushbutton1_Callback(hObject, eventdata, handles)
    [names,speech]=crear_speech;
    h=guidata(gcbo);
    ukuran_codebook=get(h.ukuran_codebook,'Value');
    switch ukuran_codebook
        case 1
            ukuran_codebook = 32;
        case 2
            ukuran_codebook = 64;
        case 3
            ukuran_codebook = 128;
        case 4
            ukuran_codebook = 256;
        case 5
```

```
            ukuran_codebook = 512;
        case 6
            ukuran_codebook = 1024;
    end;
    filename=get(h.nama_filet,'String');
    jumlah_iterasi=get(h.jumlah_iterasi,'String');
    jumlah_iterasi=str2num(jumlah_iterasi);
    Code=VQ_training(speech,ukuran_codebook,jumlah_iterasi);
    eval(['save ' filename ' Code names']);
%==========================================================================

function jumlah_iterasi_Callback(hObject, eventdata, handles)
% hObject    handle to jumlah_iterasi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of jumlah_iterasi as text
%        str2double(get(hObject,'String')) returns contents of
jumlah_iterasi as a double


% --- Executes during object creation, after setting all properties.
function jumlah_iterasi_CreateFcn(hObject, eventdata, handles)
% hObject    handle to jumlah_iterasi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function jumlah_label_Callback(hObject, eventdata, handles)
% hObject    handle to jumlah_label (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of jumlah_label as text
%        str2double(get(hObject,'String')) returns contents of
jumlah_label as a double


% --- Executes during object creation, after setting all properties.
function jumlah_label_CreateFcn(hObject, eventdata, handles)
% hObject    handle to jumlah_label (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function nama_filet_Callback(hObject, eventdata, handles)
% hObject    handle to nama_filet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of nama_filet as text
%        str2double(get(hObject,'String')) returns contents of nama_filet
as a double


% --- Executes during object creation, after setting all properties.
function nama_filet_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nama_filet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%==========================================================================
function [names,A]=crear_speech;
    h=guidata(gcbo);
    jumlah_label=get(h.jumlah_label,'String');
    jumlah_label=str2num(jumlah_label);
    A=[];
    names=cell(4,1);
    for i=1:jumlah_label
       load(['label' int2str(i)])
       names{i}=[etiqueta];
        [a,b]=size(label);
        for i=1:b,
        sp=label(1:a,i);
        A=[A;sp];
    end;
    end;
    %close all
%==========================================================================


%==========================================================================
function F=extraction(speech,length_frame,overlap,fs)
    nmax_frame=floor(length(speech)/length_frame);
    sp=[zeros(1,overlap) speech' zeros(1,overlap)]';
    frame=0;
    for i=overlap+1:length_frame:nmax_frame*length_frame,
       frame=frame+1;
       sframe=sp((i-overlap):(i+length_frame-1+overlap));

[spec,logSpec,f,Pcoeff1,F1(frame,:),F2(frame,:)]=mfcc3(sframe.*hamming(le
ngth(sframe)),fs);
    end;
    F=F2;
%==========================================================================


%==========================================================================
function [spec,logSpec,f,Pcoeff1,coeffs1,coeffs2]=mfcc3(signal,fs)
    spec=fft(signal);
    logSpec=log(abs(spec(1:floor((length(spec)+1)/2))));
    nbpts=length(logSpec);
    i=1;
    freq(i)=1;
    f(i)=1;
```

```
    while freq(i)<nbpts
       i=i+1;
       f(i)=(exp(log(2)*(150*(i-1))/1000)-1)*1000;
       freq(i)=floor((exp(log(2)*(150*(i-1))/1000)-1)*1000*2/fs*nbpts);
    end
    freq(i+1)=floor((exp(log(2)*(150*i)/1000)-1)*1000*2/fs*nbpts);
    NbFilters=length(freq)-2;
    filters=zeros(NbFilters,freq(end)-freq(end-1));
    for k=1:NbFilters
       filters(k,1:freq(k+2)-freq(k))=triang(freq(k+2)-freq(k))';
    end
    ZerosVect=zeros(freq(end)-1-length(logSpec),1);
    logSpec=[logSpec;ZerosVect];
    spec=[spec(1:floor((length(spec)+1)/2));ZerosVect];
    for j=1:NbFilters
       Pcoeff1(j)=sum(filters(j,1:freq(j+2)-
freq(j))'.*logSpec(freq(j):freq(j+2)-1));
       Pcoeff2(j)=sum(filters(j,1:freq(j+2)-
freq(j))'.*abs(spec(freq(j):freq(j+2)-1)));
    end
    NbCoeffs=floor(NbFilters/2)+1;
    MAX=max(10*log10(Pcoeff2));
    for k=0:NbCoeffs-1

coeffs1(k+1)=1/NbFilters*sum(Pcoeff1.*cos(2*pi*k/NbFilters*[0:NbFilters-
1]));
       coeffs2(k+1)=1/NbFilters*sum((10*log10(Pcoeff2)-
MAX+50).*cos(2*pi*k/NbFilters*[0:NbFilters-1]));
    end
%=========================================================================

%=========================================================================
function Cfinal=VQ_training(speech,M,iteration);
    F=extraction(speech,100,78,6000);
    save feat_default F
    [a,b]=size(F);
    Cfinal=split2(F,floor(log(M)/log(2)),iteration);
    Code=Cfinal;
    save codebook Code
%=========================================================================

%=========================================================================
function [C,V,k]=split2(X,bits,iteration);
    sigma=std(X);
    [N,L]=size(X);
    C(1,:)=mean(X);
    h = guidata(gcbo);
    for s=1:bits,
       [M,L]=size(C);
       for c=1:M,
          C(c,:)=C(c,:)-0.00000001*sigma;
          C(c+M,:)=C(c,:)+0.00000001*sigma;
       end;
       set(h.status,'String',[ int2str(floor(100*s/bits)) ' ' '%' ])
       [C,V,k]=gla2(X,C,iteration);
    end;
    set(h.status, 'String', '100 %')
    %figure(1)
    %plot(X(:,1),X(:,2),'g.')
    %hold on
    %voronoi(C(:,1),C(:,2));
    %hold off
%=========================================================================

%=========================================================================
```

```
function [CN,V,k]=gla2(X,C,iteration);
    [M,L]=size(C);
    [N,L]=size(X);
    Diff=zeros(M,N);
    V=zeros(N,1);
     plot(X(:,1),X(:,2),'g.');
     hold on
     plot(C(:,1),C(:,2),'r*');
     hold off
     pause(1)
     CN=C;
    for it=1:iteration,
          k=zeros(N,1);
          V=zeros(N,1);
          for x=1:N,
            for c=1:M,
               Diff(c,x)=norm(X(x,:)-CN(c,:));
            end;
          end;
          [val,V]=min(Diff);
          k=zeros(M,1);
          CN=zeros(M,L);
          for x=1:N,
             CN(V(x),:)=CN(V(x),:)+X(x,:);
             k(V(x))=k(V(x))+1;
          end;
          for c=1:M,
             if k(c)>0
                CN(c,:)=CN(c,:)/k(c);
             else
                CN(c,:)=C(c,:);
             end;
          end;
          plot(X(:,1),X(:,2),'g.');
          hold on
          plot(CN(:,1),CN(:,2),'r*');
          title('Codebook Training');
          hold off
          pause(1)
            dis=0;
        for i=1:N,
            dis=norm(X(i,:)-CN(V(i),:))^2+dis;
          end;
    end;
%=========================================================================

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
helpcodebook


function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to nama_filet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nama_filet as text
%        str2double(get(hObject,'String')) returns contents of nama_filet
as a double

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nama_filet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to status (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of status as text
%        str2double(get(hObject,'String')) returns contents of status as
a double


% --- Executes during object creation, after setting all properties.
function status_CreateFcn(hObject, eventdata, handles)
% hObject    handle to status (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

3. Listing Program Pembuatan Parameter HMM

```
function varargout = hmmjantung(varargin)
% HMMJANTUNG M-file for hmmjantung.fig
%   HMMJANTUNG, by itself, creates a new HMMJANTUNG or raises the
%   existing singleton*.
%
%   H = HMMJANTUNG returns the handle to a new HMMJANTUNG or the handle
%   to the existing singleton*.
%
%   HMMJANTUNG('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in HMMJANTUNG.M with the given input
%   arguments.
%
%   HMMJANTUNG('Property','Value',...) creates a new HMMJANTUNG or raises
%   the existing singleton*.  Starting from the left, property value
%   pairs are applied to the GUI before hmmjantung_OpeningFcn gets
%   called.  An unrecognized property name or invalid value makes
%   property application stop.  All inputs are passed to
%   hmmjantung_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help hmmjantung

% Last Modified by GUIDE v2.5 23-May-2009 00:06:05

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @hmmjantung_OpeningFcn, ...
                   'gui_OutputFcn',  @hmmjantung_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before hmmjantung is made visible.
function hmmjantung_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to hmmjantung (see VARARGIN)

% Choose default command line output for hmmjantung
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes hmmjantung wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = hmmjantung_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function file_hmm_Callback(hObject, eventdata, handles)
% hObject    handle to file_hmm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of file_hmm as text
%        str2double(get(hObject,'String')) returns contents of file_hmm
as a double

% --- Executes during object creation, after setting all properties.
function file_hmm_CreateFcn(hObject, eventdata, handles)
% hObject    handle to file_hmm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function file_codebook_Callback(hObject, eventdata, handles)
% hObject    handle to file_codebook (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of file_codebook as text
%        str2double(get(hObject,'String')) returns contents of
file_codebook as a double


% --- Executes during object creation, after setting all properties.
function file_codebook_CreateFcn(hObject, eventdata, handles)
% hObject    handle to file_codebook (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in proceed.
```

```
function proceed_Callback(hObject, eventdata, handles)
h=guidata(gcbo);
    file_hmm=get(h.file_hmm,'String');
    file_codebook=get(h.file_codebook,'String');
    iterasi=get(h.jumlah_iterasi,'String');
    iterasi = str2num(iterasi);
    make_HMM(file_hmm,file_codebook,iterasi);

% --- Executes on button press in notes.
function notes_Callback(hObject, eventdata, handles)
helphmm


function jumlah_iterasi_Callback(hObject, eventdata, handles)
% hObject    handle to jumlah_iterasi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of jumlah_iterasi as text
%        str2double(get(hObject,'String')) returns contents of
jumlah_iterasi as a double


% --- Executes during object creation, after setting all properties.
function jumlah_iterasi_CreateFcn(hObject, eventdata, handles)
% hObject    handle to jumlah_iterasi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%=========================================================================
function make_HMM(model_file,codebook,iteration)
    load(codebook);
    [M,a]=size(Code);
    SAVING_OBSERVATION_new(codebook,'fileobservation');
    TRAIN_PART_demo(model_file,'fileobservation',iteration,M);
%=========================================================================


%=========================================================================
function SAVING_OBSERVATION_new(codebook2,fileobservation256)
    label=1;
    %label 1
    [O_1]=EXTRACTING_OBSERVATION_2(label,codebook2);
    label=label+1;
    eval(['save' ' ' fileobservation256 ' O_1']);

    %label 2
    [O_2]=EXTRACTING_OBSERVATION_2(label,codebook2);
    label=label+1;
    eval(['save' ' ' fileobservation256 ' O_1 O_2']);

    %label 3
    [O_3]=EXTRACTING_OBSERVATION_2(label,codebook2);
    label=label+1;
    eval(['save' ' ' fileobservation256 ' O_1 O_2 O_3']);
```

```
    %label 4
    [O_4]=EXTRACTING_OBSERVATION_2(label,codebook2);
    label=label+1;
    eval(['save' ' ' fileobservation256 ' O_1 O_2 O_3 O_4']);
%=========================================================================

%=========================================================================
function [O2]=EXTRACTING_OBSERVATION_2(label1,codebook2)
    eval(['load' ' label' int2str(label1)]); %call variable label
    [rows,total]=size(label);
    continue1=1;
    finish=0;
    total_seq2=[];
    eval(['load' ' ' codebook2]);
    for i=1:total
       speech=label(:,i);
       [a,b]=size(label);
       F=extraction(speech(1:a),100,78,6000);
       [seq2,Y,Distortion]=vq(F,Code);
       total_seq2=[total_seq2 seq2 0];
    end;
    O2=matrix_func(total_seq2);
%=========================================================================


%=========================================================================
function F=extraction(speech,length_frame,overlap,fs)
    nmax_frame=floor(length(speech)/length_frame);
    sp=[zeros(1,overlap) speech' zeros(1,overlap)]';
    frame=0;
    for i=overlap+1:length_frame:nmax_frame*length_frame,
       frame=frame+1;
       sframe=sp((i-overlap):(i+length_frame-1+overlap));

[spec,logSpec,f,Pcoeff1,F1(frame,:),F2(frame,:)]=mfcc3(sframe.*hamming(le
ngth(sframe)),fs);
    end;
    F=F2;
%=========================================================================


%=========================================================================
function [spec,logSpec,f,Pcoeff1,coeffs1,coeffs2]=mfcc3(signal,fs)
    %spec=fft(signal);
    spec=fft(signal,256);
    spec = spec/1000;
    logSpec=log(abs(spec(1:floor((length(spec)+1)/2))));
    nbpts=length(logSpec);
    i=1;
    freq(i)=1;
    f(i)=1;
    while freq(i)<nbpts
       i=i+1;
       f(i)=(exp(log(2)*(150*(i-1))/1000)-1)*1000;
       freq(i)=floor((exp(log(2)*(150*(i-1))/1000)-1)*1000*2/fs*nbpts);
    end
    freq(i+1)=floor((exp(log(2)*(150*i)/1000)-1)*1000*2/fs*nbpts);
    NbFilters=length(freq)-2;
    filters=zeros(NbFilters,freq(end)-freq(end-1));
    for k=1:NbFilters
       filters(k,1:freq(k+2)-freq(k))=triang(freq(k+2)-freq(k))';
    end
    ZerosVect=zeros(freq(end)-1-length(logSpec),1);
    logSpec=[logSpec;ZerosVect];
    spec=[spec(1:floor((length(spec)+1)/2));ZerosVect];
```

```
    for j=1:NbFilters
       Pcoeff1(j)=sum(filters(j,1:freq(j+2)-
freq(j))'.*logSpec(freq(j):freq(j+2)-1));
       Pcoeff2(j)=sum(filters(j,1:freq(j+2)-
freq(j))'.*abs(spec(freq(j):freq(j+2)-1)));
    end
    NbCoeffs=floor(NbFilters/2)+1;
    MAX=max(10*log10(Pcoeff2));
    for k=0:NbCoeffs-1

coeffs1(k+1)=1/NbFilters*sum(Pcoeff1.*cos(2*pi*k/NbFilters*[0:NbFilters-
1]));
       coeffs2(k+1)=1/NbFilters*sum((10*log10(Pcoeff2)-
MAX+50).*cos(2*pi*k/NbFilters*[0:NbFilters-1]));
    end
%=========================================================================


%=========================================================================
function TRAIN_PART_demo(model_file,obs_256,iteration,M);
    N=4;
    load(obs_256);
    %Create the file where will be stored the matrix HMMs
    initialize_models(model_file,N,M);
    load(model_file);
    %label 1
    tic
    [A,B,p0]=generate_random_model(N,M);
    [A1,B1,p01]=baum(A,B,p0,O_1,iteration);
    B1=arrangefunc(B1,10^-5);
    available=available-1;
    eval(['save' ' ' model_file ' ' 'A1 A2 A3 A4 B1 B2 B3 B4 p01 p02 p03
p04 available N M'])
    time=toc
    %label 2
    [A,B,p0]=generate_random_model(N,M);
    [A2,B2,p02]=baum(A,B,p0,O_2,iteration);
    B2=arrangefunc(B2,10^-5);
    available=available-1;
    eval(['save' ' ' model_file ' ' 'A1 A2 A3 A4 B1 B2 B3 B4 p01 p02 p03
p04 available N M'])

    %label 3
    [A,B,p0]=generate_random_model(N,M);
    [A3,B3,p03]=baum(A,B,p0,O_3,iteration);
    B3=arrangefunc(B3,10^-5);
    available=available-1;
    eval(['save' ' ' model_file ' ' 'A1 A2 A3 A4 B1 B2 B3 B4 p01 p02 p03
p04 available N M'])

    %label 4
    [A,B,p0]=generate_random_model(N,M);
    [A4,B4,p04]=baum(A,B,p0,O_4,iteration);
    B4=arrangefunc(B4,10^-5);
    available=available-1;
    eval(['save' ' ' model_file ' ' 'A1 A2 A3 A4 B1 B2 B3 B4 p01 p02 p03
p04 available N M'])

%=========================================================================

%=========================================================================
function [O,Y,Distortion]=vq(X,C);
    [N,L]=size(X);
    [M,L]=size(C);
    Distortion=0;
```

```
    for t=1:N,
        for c=1:M,
            dist(c)=norm(X(t,:)-C(c,:)); %you can use only the square
distance without use square root
        end;
        [val,O(t)]=min(dist);
        Distortion=Distortion+val^2;
        Y(t,:)=C(O(t),:);
    end;
    Distortion=Distortion/N;
%=========================================================================


%=========================================================================
function O=matrix_func(X);
    A=[0 find(X==0)]; %Returnes the index of the zeros in a vector.
    N=length(A)-1;
    L=zeros(1,N);
    for i=1:N,
       L(i)=A(i+1)-A(i);
    end
    M=max(L);
    O=zeros(N,M);
    for j=1:N,
       O(j,:)=[X(A(j)+1:A(j+1)-1) zeros(1,(M-L(j)+1))];
    end
%=========================================================================


%=========================================================================
function initialize_models(model,N,M);
    available=4;
    %model 1
    A1=zeros(N);
    B1=zeros(N,M);
    p01=zeros(N,1);
    %model 2
    A2=zeros(N);
    B2=zeros(N,M);
    p02=zeros(N,1);
    %model 3
    A3=zeros(N);
    B3=zeros(N,M);
    p03=zeros(N,1);
    %model 4
    A4=zeros(N);
    B4=zeros(N,M);
    p04=zeros(N,1);

    %create the memory file
    eval(['save' ' ' model ' ' 'A1 A2 A3 A4 B1 B2 B3 B4 p01 p02 p03 p04
available N M'])
%=========================================================================


%=========================================================================
function [A,B,p0]=generate_random_model(N,M);
    delta=1;
    A=triu(rand(N,N));
    for i=1:N,
       for j=1:N,
          if j>i+delta
             A(i,j)=0;
          end;
       end;
```

```
    end;
    B=rand(N,M);
    p0=zeros(N,1);
    for i=1:N,
       A(i,:)=A(i,:)/sum(A(i,:));
       B(i,:)=B(i,:)/sum(B(i,:));
    end;
    p0(1)=1;
%=========================================================================


%=========================================================================
function [A_new,B_new,p0_new]=baum(A,B,p0,O,iteration);
    A_new=A;
    B_new=B;
    p0_new=p0;
    for i=1:iteration,

[A_new,B_new,p0_new,T]=HMM_multiple_training3(A_new,B_new,p0_new,O);
       K=length(T);
       Pall(i)=0;
       for k=1:K,
          [alfa,c]=HMM_forward(A_new,B_new,p0_new,O(k,1:T(k)));
          P(i,k)=HMM_probability(c);
          Pall(i)=Pall(i)+P(i,k);
       end;
    end;
    plot(Pall);
    title('Label Probability');
    pause(0.1)
%=========================================================================


%=========================================================================
function [A_new,B_new,p0_new,T]=HMM_multiple_training3(A,B,p0,O);
    [K,Tmax]=size(O);
    T=zeros(1,K);
    pl=zeros(1,K+1);
    pl(1)=0;
    for k=2:K+1,
       for t=1:Tmax,
          if O(k-1,t)~=0
             T(k-1)=T(k-1)+1;
          else
             break
          end;
       end;
       pl(k)=sum(T(1:k-1));
    end;
    for k=1:K,

[alfa(:,pl(k)+1:pl(k+1)),c(pl(k)+1:pl(k+1))]=HMM_forward(A,B,p0,O(k,1:T(k
)));

beta(:,pl(k)+1:pl(k+1))=HMM_backward(A,B,p0,O(k,1:T(k)),c(pl(k)+1:pl(k+1)
));
    end;
    [n,m]=size(B);
    A_new=zeros(n,n);
    A_temp=zeros(n,K*n);
    for k=1:K,
       alfaki=alfak(alfa,pl,k);
       betaki=betak(beta,pl,k);
       for i=1:n,
          for j=1:n,
```

```
            for t=1:T(k)-1,%start sum
                A_temp(i,n*(k-1)+j)=A_temp(i,n*(k-
1)+j)+alfaki(i,t)*A(i,j)*B(j,O(k,t+1))*betaki(j,t+1);
                end; %end sum
          end;
       end;
    end;
    for i=1:n,
       for j=1:n,
          num=0;
          den=0;
          for k=1:K,
             num=num+A_temp(i,n*(k-1)+j);
          end;
          for k=1:K,
             for s=1:n,
                den=den+A_temp(i,n*(k-1)+s);
             end;
          end;
          A_new(i,j)=num/den;
       end;
    end;
    %gama=[gama1 gama2 ... gamaK]
    for k=1:K,
       alfaki=alfak(alfa,pl,k);
       betaki=betak(beta,pl,k);
       for i=1:n,
            for t=1:T(k),
             gama(i,pl(k)+t)=alfaki(i,t)*betaki(i,t);
             den=0;
             for s=1:n,
                den=den+alfaki(s,t)*betaki(s,t);
             end;
            gama(i,pl(k)+t)=gama(i,pl(k)+t)/den;
        end;
      end;
    end;
    for l=1:m,
       for j=1:n,
          num=0;
          den=0;
          for k=1:K,
             gammaki=gammak(gama,pl,k);
             for t=1:T(k),
                if O(k,t)==l
                   num=num+gammaki(j,t);
                end;
             end;
             for t=1:T(k),
                den=den+gammaki(j,t);
             end;
          end;
          B_new(j,l)=num/den;
       end;
    end;
    p0_new=p0;
%=========================================================================


%=========================================================================
function [alfa,c]=HMM_forward(A,B,p0,O);
    T=length(O);
    %t=1
    alfa_temp=p0.*B(:,O(1));
    c(1)=1/sum(alfa_temp);
```

```
    alfa(:,1)=c(1)*alfa_temp;

    for t=2:T,
      alfa_temp=B(:,O(t)).*(A'*alfa(:,t-1));
      c(t)=1/sum(alfa_temp);
      alfa(:,t)=c(t)*alfa_temp;
    end;
%=========================================================================


%=========================================================================
function beta=HMM_backward(A,B,p0,O,c);
    T=length(O);
    [n,m]=size(B);
    %t=T;
    beta_temp=ones(n,1);
    beta(:,T)=c(T)*beta_temp;

    for t=(T-1):-1:1,
      beta_temp= A*(beta(:,t+1).*B(:,O(t+1)));
      beta(:,t)=c(t)*beta_temp;
    end;
%=========================================================================


%=========================================================================
function alfaki=alfak(alfa,pl,k);
    alfaki=alfa(:,pl(k)+1:pl(k+1));
%=========================================================================


%=========================================================================
function betaki=betak(beta,pl,k);
    betaki=beta(:,pl(k)+1:pl(k+1));
%=========================================================================

%=========================================================================
function gammaki=gammak(gamma,pl,k);
    gammaki=gamma(:,pl(k)+1:pl(k+1));
%=========================================================================


%=========================================================================
function logp=HMM_probability(c);
    logp=-sum(log(c));
%=========================================================================


%=========================================================================
function B_new=arrangefunc(B,E)
    [N,M]=size(B);
    for i=1:N,
       pos=find(B(i,:)<E);
       numbzer=(length(pos));
       E_new=E*sum(B(i,:).*pulse(pos,M))/(1-E*numbzer);
       for m=1:numbzer,
          B(i,pos(m))=E_new;
       end;
       B_new(i,:)=B(i,:)./sum(B(i,:));
    end;
%=========================================================================


%=========================================================================
function vect=pulse(pos,M);
```

```
    vect=ones(1,M);
    for i=1:length(pos),
       vect(pos(i))=0;
    end;
%=========================================================================
```

4. Listing Program Proses Pengenalan

```
function varargout = guifinal(varargin)
% GUIFINAL M-file for guifinal.fig
%      GUIFINAL, by itself, creates a new GUIFINAL or raises the existing
%      singleton*.
%
%      H = GUIFINAL returns the handle to a new GUIFINAL or the handle to
%      the existing singleton*.
%
%      GUIFINAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUIFINAL.M with the given input
arguments.
%
%      GUIFINAL('Property','Value',...) creates a new GUIFINAL or raises
%      the existing singleton*.  Starting from the left, property value
%      pairs are applied to the GUI before guifinal_OpeningFcn gets called.
%      An unrecognized property name or invalid value makes property
%      application stop.  All inputs are passed to guifinal_OpeningFcn via
%      varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help guifinal

% Last Modified by GUIDE v2.5 07-Jun-2009 19:57:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guifinal_OpeningFcn, ...
                   'gui_OutputFcn',  @guifinal_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before guifinal is made visible.
function guifinal_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to guifinal (see VARARGIN)

% Choose default command line output for guifinal
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guifinal wait for user response (see UIRESUME)
```

```
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = guifinal_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function file_inputan_Callback(hObject, eventdata, handles)
% hObject    handle to file_inputan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of file_inputan as text
%        str2double(get(hObject,'String')) returns contents of
file_inputan as a double


% --- Executes during object creation, after setting all properties.
function file_inputan_CreateFcn(hObject, eventdata, handles)
% hObject    handle to file_inputan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function file_codebook_Callback(hObject, eventdata, handles)
% hObject    handle to file_codebook (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of file_codebook as text
%        str2double(get(hObject,'String')) returns contents of
file_codebook as a double


% --- Executes during object creation, after setting all properties.
function file_codebook_CreateFcn(hObject, eventdata, handles)
% hObject    handle to file_codebook (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function file_hmm_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to file_hmm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of file_hmm as text
%        str2double(get(hObject,'String')) returns contents of file_hmm
as a double


% --- Executes during object creation, after setting all properties.
function file_hmm_CreateFcn(hObject, eventdata, handles)
% hObject    handle to file_hmm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
    h=guidata(gcbo);
    if get(h.radiobutton1,'value') == 1
        time = 0.7
    else
        time = 1.5
    end
    file_hmm=get(h.file_hmm,'String');
    file_codebook=get(h.file_codebook,'String');
    file_inputan=get(h.file_inputan,'String');
    recognition(file_hmm,file_codebook,time,file_inputan);


% --- Executes on button press in notes.
function notes_Callback(hObject, eventdata, handles)
helpguifinal

function recognition(model,book,time,file_inputan);
    % frekuensi sampling
    % fs=11025;
    fs = 6000;
    eval(['load' ' ' book]);

%     eval(['load' ' ' file_inputan]);
%     speech =  eval(file_inputan);
    try
    [speech]=wavread(file_inputan,time*6000);
        catch ME
   %Get last segment of the error message identifier.
   %idSegLast = regexp(ME.identifier, '(?<=:)\w+$', 'match');

   %Did the read fail because the file could not be found?
   %if strcmp(idSegLast, 'InvalidFid') && ~exist(filename, 'file')

         h=guidata(gcbo);
         set(h.hasil,'String','file does not exist')

       %end
       return
    end
```

```
    F=extraction(speech,100,78,fs);
    %F
    %VQ
    [O,Y,Distortion]=vq(F,Code);
    %O
    %decision
    eval(['load' ' ' model]);
    %model 1
    number=4-available;
    P=-inf*ones(1,4);
    if number>=1
    [alfa,c]=HMM_forward(A1,B1,p01,O);
    P(1)=HMM_probability(c);
    alfa;`
    c;
    end;
    %model 2
    if number>=2
    [alfa,c]=HMM_forward(A2,B2,p02,O);
    P(2)=HMM_probability(c);
    end;
    %model 3
    if number>=3
    [alfa,c]=HMM_forward(A3,B3,p03,O);
    P(3)=HMM_probability(c);
    end;
    %model 4
    if number>=4
    [alfa,c]=HMM_forward(A4,B4,p04,O);
    P(4)=HMM_probability(c);
    end;

    P;
    %choose the maximun
    [val,index]=max(P);
    measure=val-mean(P);
    [Y,I]=sort(-P);
    fprintf('\n');
     fprintf('Label   Log of Probability\n')
    fprintf('-------------------------\n')
    for j=1:4,
      fprintf(' %i      %f\n',I(j), -Y(j));

   end;
   fprintf('\n');
    nam=names{index};

   fprintf('jenis penyakit :   %s\n',nam);
    h=guidata(gcbo);
    set(h.hasil,'String',nam);
%=========================================================================


%=========================================================================
function F=extraction(speech,length_frame,overlap,fs)
    nmax_frame=floor(length(speech)/length_frame);
    sp=[zeros(1,overlap) speech' zeros(1,overlap)]';
    frame=0;
    for i=overlap+1:length_frame:nmax_frame*length_frame,

       frame=frame+1;
       sframe=sp((i-overlap):(i+length_frame-1+overlap));
       sframe.*hamming(length(sframe));
```

```
[spec,logSpec,f,Pcoeff1,F1(frame,:),F2(frame,:)]=mfcc3(sframe.*hamming(le
ngth(sframe)),fs);
    end;
    F=F2;
%====================================================================


%====================================================================
function [spec,logSpec,f,Pcoeff1,coeffs1,coeffs2]=mfcc3(signal,fs)
    %spec=fft(signal);
    spec=fft(signal,256);
    spec = spec/1000;
    logSpec=log(abs(spec(1:floor((length(spec)+1)/2))));
    nbpts=length(logSpec);

    % Creation of the filter bank
    %% creation of a vector of mels spaced frequencies
    i=1;
    freq(i)=1;
    f(i)=1;
    while freq(i)<nbpts
       i=i+1;
       f(i)=(exp(log(2)*(150*(i-1))/1000)-1)*1000;
       freq(i)=floor((exp(log(2)*(150*(i-1))/1000)-1)*1000*2/fs*nbpts);
    end
    freq(i+1)=floor((exp(log(2)*(150*i)/1000)-1)*1000*2/fs*nbpts);

    %freq
    %% creation of the triangular filters
    NbFilters=length(freq)-2;
    filters=zeros(NbFilters,freq(end)-freq(end-1));
    for k=1:NbFilters
       filters(k,1:freq(k+2)-freq(k))=triang(freq(k+2)-freq(k))';
    end
    %filters

    % Adding zeros to the spectrum to match the filter bank's length
    ZerosVect=zeros(freq(end)-1-length(logSpec),1);
    logSpec=[logSpec;ZerosVect];
    spec=[spec(1:floor((length(spec)+1)/2));ZerosVect];

    % Computation of the power coefficients
    for j=1:NbFilters
       Pcoeff1(j)=sum(filters(j,1:freq(j+2)-
freq(j))'.*logSpec(freq(j):freq(j+2)-1));
       Pcoeff2(j)=sum(filters(j,1:freq(j+2)-
freq(j))'.*abs(spec(freq(j):freq(j+2)-1)));
    end
    %Pcoeff2
    % Computation of the mel cepstral coefficients
    NbCoeffs=floor(NbFilters/2)+1;
    MAX=max(10*log10(Pcoeff2));
    for k=0:NbCoeffs-1

coeffs1(k+1)=1/NbFilters*sum(Pcoeff1.*cos(2*pi*k/NbFilters*[0:NbFilters-
1]));
       coeffs2(k+1)=1/NbFilters*sum((10*log10(Pcoeff2)-
MAX+50).*cos(2*pi*k/NbFilters*[0:NbFilters-1]));
    end
%====================================================================


%====================================================================
function [O,Y,Distortion]=vq(X,C);
```

```
    [N,L]=size(X);
    [M,L]=size(C);
    Distortion=0;
    for t=1:N,
        for c=1:M,
            dist(c)=norm(X(t,:)-C(c,:)); %you can use only the square
distance without use square root
        end;
        %O(t)
        [val,O(t)]=min(dist);
        Distortion=Distortion+val^2;
        Y(t,:)=C(O(t),:);
    end;
    Distortion=Distortion/N;
%=====================================================================


%=====================================================================
function [alfa,c]=HMM_forward(A,B,p0,O);
    T=length(O);
    %t=1
    alfa_temp=p0.*B(:,O(1));
    c(1)=1/sum(alfa_temp);
    %c(1)
    alfa(:,1)=c(1)*alfa_temp;
    %alfa(:,1)

    for t=2:T,
      alfa_temp=B(:,O(t)).*(A'*alfa(:,t-1));
      c(t)=1/sum(alfa_temp);
      %c(t)
      alfa(:,t)=c(t)*alfa_temp;
      %alfa(:,t)
    end;
%=====================================================================


%=====================================================================
function logp=HMM_probability(c);
    logp=-sum(log(c));
%=====================================================================

% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
h=guidata(gcbo);
set(h.radiobutton2,'value',0);

% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
h=guidata(gcbo);
set(h.radiobutton1,'value',0);
```