HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Abdul Aziz Muslim**
NPM	**: 0606073663**
Tanda Tangan	**:**
Tanggal	**: 15 Juni 2010**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Abdul Aziz M
NPM	: 0606073663
Program Studi	: Teknik Elektro
Judul Skripsi	: Simulasi Sistem Query by Singing/Humming untuk Musik Dangdut dengan Menggunakan Pitch Sebagai Fitur Melodi

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

DEWAN PENGUJI

Pembimbing: Prof. Dr. Ir. Dadang Gunawan M.Eng. ( )

Penguji : Dr. Ir. Arman D. Diponegoro ( )

Penguji : Filbert Hilman Juwono S.T., M.T. ( )

Ditetapkan di : Depok

Tanggal : 23 Juni 2010

UCAPAN TERIMA KASIH

Alhamdulillah, segala puji dan syukur saya ucapkan kehadirat Allah SWT karena berkat rahmat, ridho, hidayah, inayah dan kasih sayang-Nya, penulisan skripsi ini bisa selesai di waktu yang tepat. Shalawat dan salam selalu saya haturkan kepada baginda Rasulullah Muhammad SAW, karena berkat jasa beliaulah kita dapat hidup di zaman yang terang benderang ini. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya ingin mengucapkan terima kasih kepada :

1. Prof. Dr. Ir. Dadang Gunawan, M.Eng selaku pembimbing skripsi yang telah menyediakan waktu, tenaga dan pikiran memberikan arahan dalam penyusunan skripsi ini;
2. Indrabayu Amirullah, ST, MT, M.Bus, Sys selaku pembimbing harian yang telah banyak memberikan motivasi ,waktu serta ilmunya selama proses pembuatan skripsi ini ;
3. Teddy Febrianto dan Ewaldo Zihan yang membantu proses pembelajaran serta penulisan skripsi ini. Tanpa bantuan, dorongan dan semangat kalian berdua saya tidak mungkin dapat menyelesaikan skripsi ini;
4. Ibunda tercinta, Sumingsih, yang telah memberikan kasih sayang, do'a dan dukungan. Dan sebagai panutan yang telah mengajarkan makna sebuah perjuangan seorang *single parent* yang ingin mewujudkan kesuksesan anaknya;
5. Aa Adi dan Mba Atri yang telah membantu biaya kuliah serta membangunkan tiap malam. Ceu Hikmah, Sandi Ajda , Azki, Hijaz, Anis dan Gaza. Serta semua anggota keluarga besar Bapak Damuri;
7. Bapak Muslim yang telah mengajarkan ilmu ma'rifatullah, terimakasih atas wejangan dan do'anya;
6. dr. Yuli Supriatin yang telah memberikan semangat dan motivasinya;
7. Rony Wijaya sebagai teman seperjuangan yang telah mengajarkan makna kekuatan ikhlas. Terimakasih atas pinjaman dananya untuk membeli laptop;

8. Ardyan Indra, Ahmad Ari Syakbani, Indra G, Aneu Dwi, Indah Riyanti dan Cindy Chairunissa. Terimakasih atas sumbangan suara merdunya untuk sampel data. Mudah-mudahan kalian bisa menjadi bintang KDI TPI masa depan (^_^);
9. Nur Arifin, mahasiswa STIS semester 6, yang telah mengajarkan ilmu statistika. Terimakasih atas pencerahannya Yip!;
10. Bripda Yogi Ryan Hidayat dan Hakim Agung Ramadhan yang telah memberikan semangat dan motivasinya;
11. Teman-teman asisten Laboratorium Telekomunikasi Teknik Elektro UI;
12. Teman-teman Teknik Elektro UI angkatan 2006;
13. Teman-teman kostan Faisal, Krisna dan Adri. Semoga kita semua bisa sukses di kehidupan masing-masing. Terima kasih juga untuk Pak Buyung dan Bu Indra yang telah member nasihat dan motivasi.
14. Para peneliti sebelum ini yang menjadi referensi dalam penulisan laporan skripsi ini.

Akhir kata, semoga Allah SWT berkenan membalas kebaikan semua pihak yang telah membantu. Jazakumullah khaiaran katsiran. Semoga skripsi ini bermanfaat bagi perkembangan ilmu pengetahuan.

Depok, 15 Juni 2010

Abdul Aziz Muslim

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Abdul Aziz Muslim

NPM : 0606073663

Program Studi : Teknik Elektro

Departemen : Teknik Elektro

Fakultas : Teknik

Jenis Karya : Skripsi

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

SIMULASI SISTEM QUERY BY SINGING/HUMMING UNTUK MUSIK DANGDUT DENGAN MENGGUNAKAN PITCH SEBAGAI FITUR MELODI

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 : 15 Juni 2010

Yang Menyatakan

(Abdul Aziz Muslim)

ABSTRAK

Nama : Abdul Aziz Muslim
Departemen : Teknik Elektro
Judul : Simulasi Sistem Query by Singing/Humming untuk Musik Dangdut dengan Menggunakan Pitch Sebagai Fitur Melodi

Sistem *Query by Singing/Humming* (QbSH) merupakan bidang MIR yang menawarkan teknologi pencarian lagu berdasarkan melodi. Sistem QbSH yang ada sekarang terbatas untuk lagu *western* saja. Skripsi ini membahas perancangan simulasi sistem QbSH untuk musik dangdut. Sistem QbSH yang dirancang menggunakan parameter *pitch* sebagai representasi melodi. Metode yang digunakan adalah *Auto-Correlation Function* (ACF) dan *Hidden Markov Model* (HMM). Sistem yang dibuat menggunakan *database* lagu *monophonic* MIDI. Sampel *query* yang diuji-cobakan terdiri dari 3 orang laki-laki dan 3 orang perempuan. Masing-masing orang menyenandungkan 10 lagu (*query*) yang ada dalam *database.* Optimasi hasil dilakukan dengan menggunakan variasi normalisasi *pitch* (berdasarkan energy, rata-rata dan maksimum), durasi pengambilan *pitch* (5 detik, 8 detik dan 10 detik) dari lagu dan durasi perekaman *query* (5 detik dan 10 detik).
Dari hasil percobaan didapatkan nilai MRR yang optimum dengan menggunakan normalisasi rata-rata, durasi pengambilan *pitch* 5 detik dan durasi perekaman *query* 10 detik. Sistem QbSH yang dirancang untuk musik dangdut ini memiliki nilai MRR 0,40 yang diuji-cobakan pada 60 sampel *query*.

Kata Kunci: QbSH, MIR, Pitch, Dangdut, ACF, HMM, MRR

ABSTRACT

Name : Abdul Aziz Muslim
Department : Electrical Engineering
Title : Simulation of Query by Singing/Humming System for Dangdut Music using Pitch As Melody Feature

Query by Singing/Humming (QbSH) system is one of MIR focus which offer music searching technology based on melody. The limitation of QbSH system is designed for western music. This thesis is talking about simulation design of QbSH system for dangdut music. QbSH system is designed using pitch as melody representation. The implementation method are Auto-Correlation Function (ACF) dan Hidden Markov Model (HMM). Database which created must be Monophonic MIDI . Sample of query which tested include of 3 male dan 3 female sample. Each sample humming 10 piece of songs in database. The result could be optimize using variation of pitch normalization (based on energy, mean and maximmum), duration of taking song pitch (5 second, 8 second and 10 second) and duration of recording query (5 second and 10 second).
The experiment result show that optimum MRR got using mean normalization, duration of taking pitch song 5 second and duration of recording query 10second. QbSH system which designed for dangdut music having MRR 0,40 and has been tested 60 query sample.

Keywords: QbSH, MIR, Pitch, Dangdut, ACF, HMM, MRR

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

ACF	*Auto-Correlation Function*
CHMM	*Continuous Hidden Markov Model*
GUI	*Graphical User Interface*
MIDI	*Musical Instrument Digital Interface*
MIR	*Music Information Retrieval*
MRR	*Mean Reciprocal Rank*
QbSH	*Query by Singing/Humming*

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi menyebabkan distribusi dan koleksi musik dalam format digital kian hari semakin berkembang pesat. Kehadiran musik dalam format digital sangat bermanfaat bagi manusia. Hal ini dimungkinkan dapat diorganisir, dikoleksi dan dibuat suatu *playlist* musik dengan mudah. Karenanya, kini kian banyak orang yang mempelajari musik, dan bahkan lebih jauh lagi mempelajari cara-cara pengambilan informasi dari musik itu sendiri. Cara-cara tersebut didapatkan dengan mempelajari *Music Information Retrieval* (MIR) yang berkaitan dengan representasi, penyimpanan, pengorganisasian dan perolehan informasi dari musik itu sendiri.

Dengan mempelajari MIR maka jenis-jenis musik yang beragam dan jumlah koleksi musik yang besar dapat tertangani dengan baik. Dapat dengan mudah diklasifikasi musik berdasarkan *genre* (pop, jazz, rock). Bahkan hal yang menarik pada MIR itu sendiri adalah ketika *user* lupa akan meta data dari lagu itu seperti judul lagu, *genre*, *artist* atau album maka *user* masih dapat mencari lagu yang diinginkan dengan menyenandungkan melodi dari potongan lagu yang dicari hanya dalam beberapa detik saja. Sistem ini dikenal dengan sebutan *Query by Singing/Humming* (QbSH).

Pada hakikatnya sistem ini dibuat dengan mencari kemiripan melodi dari senandungan (*query*) dengan melodi dari lagu yang ada di *database*. Dengan mengekstraksi fitur melodi berupa *pitch, contour, timbre, onset* ataupun *rythm* dari *query* dan lagu pada *database*, maka dapat dicocokkan dengan menggunakan *matching engine.* Sehingga bisa didapatkan urutan *list* lagu sesuai dengan kemiripan melodi *query*. Terdapat beberapa teknik ekstraksi fitur melodi *query* yang dikenal sebagai *Pitch Tracking.* Diantaranya adalah *Auto-Correlation Function* (ACF), *Average Magnitude Difference Function* (AMDF), *Maximmum Likelihood* dan *Cepstrum Analysis.* Teknik *pitch tracking* ini digunakan untuk

mendapatkan frekuensi dasar *query* yang disenandungkan *user*. Sedangkan untuk mencocokkan representasi melodi *query* dengan lagu pada *database* maka digunakan *Matching Engine.* Terdapat beberapa metode yang dapat digunakan untuk mencocokkan representasi melodi tersebut. Diantaranya adalah *Dynamic Time Wrapping* (DTW), *Hidden Markov Model* (HMM), *Linear Scaling*, *Note Interval Matching* dan *N-Gram Matching* [1]. Namun terdapat *trade-off* pada tiap *matching engine* yaitu akurasi dan waktu yang dibutuhkan selama proses pencocokkan melodi.

Telah banyak penelitian yang sudah dilakukan mengenai sistem QbSH ini. Mulai dari perepresentasian melodi dengan menggunakan *relative pitch* 3 level [2], penggunaaan HMM untuk melodi *matching* [3], dan kombinasi beberapa metode *matching engine* seperti *N-Gram* dengan *Note Interval* [1]. Akan tetapi kebanyakan peneliti di bidang MIR khususnya penelitian mengenai sistem QbSH terbatas untuk lagu-lagu *western.* Padahal masih banyak jenis musik lainnya yang tidak kalah menarik untuk diuji-cobakan pada sistem QbSH. Di Indonesia sendiri penelitian mengenai sistem QbSH masih sangat jarang. Apalagi penelitian mengenai sistem QbSH dengan menggunakan *database* musik dangdut masih langka. Padahal dangdut merupakan musik khas warisan budaya Indonesia yang kian lama semakin ditinggalkan. Dangdut memiliki karakteristik keunikan tersendiri dibandingkan musik *western.* Jadi belum tentu sistem QbSH yang dibuat untuk musik *western* dapat memiliki tingkat akurasi yang sama jika diaplikasikan untuk musik dangdut.

Oleh karena itu, skripsi ini mencoba merancang simulasi sistem QbSH untuk musik dangdut. Sistem QbSH yang dibuat menggunakan *Auto-Correlation Function* (ACF) sebagai teknik *pitch tracking* dan *Hidden Markov Model* (HMM) sebagai *matching engine.* Metode ACF dipilih karena simple, cepat dan sangat cocok digunakan untuk pengambilan frekuensi dasar suara manusia. Sedangkan HMM digunakan karena dapat menangani berbagai kemungkinan *error* yang terjadi seperti tempo *query* yang disenandungkan *user* terlalu cepat, perubahan tempo ataupun ketidaktepatan frekuensi dasar *query user.* Disamping itu metode HMM memiliki tingkat akurasi yang cukup tinggi dan tidak membutuhkan waktu

proses pencocokkan yang lama. Sehingga diharapkan kedua teknik ini dapat diaplikasikan untuk sistem QbSH dengan menggunakan *database* musik dangdut.

1.2 Tujuan Penulisan

Tujuan dari skripsi ini adalah merancang simulasi sistem QbSH dengan menggunakan parameter *pitch* sebagai fitur representasi melodi. Simulasi sistem QbSH ini dibuat dengan menggunakan teknik *Pitch Tracking Auto-Correlation Function* (ACF) dan metode *Matching Engine Hidden Markov Model* (HMM). Selain itu, skripsi ini ditujukan untuk meningkatkan nilai *Mean Reciprocal Rank* (MRR) dengan memvariasikan normalisasi dan durasi (waktu) pengambilan *pitch.*

1.3 Batasan Masalah

Permasalahan yang akan dibahas pada skripsi ini terbatas pada :

1. Masukan (*query*) yang disenandungkan *user* hanya diperkenankan untuk bagian awal lagu dan direkam selama 10 detik.
2. *Database* lagu dangdut yang digunakan berjumlah 15 lagu dengan format MIDI yang hanya memiliki satu *channel* melodi (monoponik).

1.4 Metode Penulisan

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

a) Studi Pustaka

Mempelajari informasi mengenai sistem QbSH baik dari buku, jurnal, artikel, *internet* dan literatur lain.

b) Pengumpulan Data

Mengumpulkan *query* dari 6 *user* (laki-laki dan perempuan) dimana setiap *user* menyenandungkan 10 *query* yang berbeda. Sehingga didapatkan jumlah keseluruhan *query* sebanyak 60 *query*.

c) Simulasi Perangkat Lunak

Merancang simulasi dan melakukan pengujian sistem QbSH untuk musik dangdut dengan menggunakan *pitch* sebagai fitur melodi.

1.5 Sistematika Penulisan

Penulisan laporan skripsi ini akan dibagi dalam lima bagian yaitu:

Bab I Pendahuluan

Bab ini menjelaskan latar belakang, tujuan penulisan, batasan masalah, metode penulisan dan sistematika pembahasan pada skripsi ini.

Bab II Musik dan Sistem Query by Singing/Humming

Bab ini menjelaskan tentang beberapa istilah musik yang berkaitan dengan sistem QbSH, teknik pengambilan frekuensi dasar *Auto-Correlation Function* (ACF) dan metode *Hidden Markov Model* (HMM) untuk sistem pencocokkan representasi melodi.

Bab III Perancangan Simulasi Sistem *Query by Singing/Humming*

Bab ini menjelaskan mengenai perancangan simulasi sistem QbSH untuk musik dangdut dengan menggunakan *Auto-Correlation Function* (ACF) sebagai teknik pencarian frekuensi dasar *query* dan metode *Hidden Markov Model* (HMM) untuk mencocokkan representasi melodi.

Bab IV Hasil Uji Coba dan Analisis Simulasi Sistem QbSH

Bab ini berisi hasil uji coba perancangan sistem QbSH yang dibuat serta analisisnya. Hasil analisis merupakan dasar untuk pembentukan kesimpulan pada penelitian ini.

Bab V Penutup

Bab ini berisi kesimpulan dan saran dari keseluruhan isi skripsi ini.

BAB 2

MUSIK DAN SISTEM QUERY BY SINGING/HUMMING

2.1 Dasar Musik

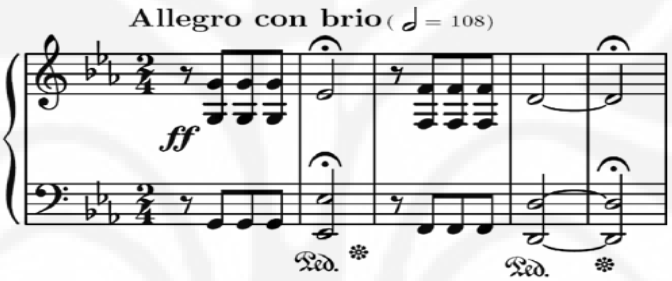
Musik merupakan suatu seni yang terdiri dari urutan suara dalam suatu waktu yang mempunyai keharmonisan dan keteraturan dalam irama, tempo dan ritme. Keharmonisan suara yang dihasilkan suatu alat musik berbeda satu sama lain. Musik ibarat bahasa, tersusun atas kalimat-kalimat yang merupakan rangkaian kata-kata. Tiap kata tersusun dari huruf-huruf yang diambil dari abjad yang sudah dikenal. Musik juga memiliki abjad, disebut sebagai tangga nada (*scale*). Dalam seni musik dikenal istilah tangga nada yang merupakan kumpulan nada-nada yang harmonis. Dikenal dua jenis tangga nada yaitu tangga nada kromatik dan tangga nada mayor [4].

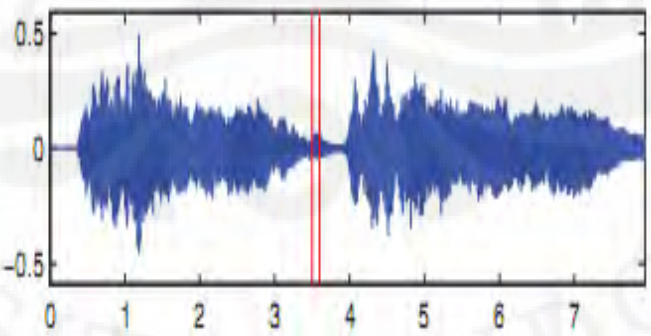
Tangga nada kromatik merupakan kumpulan semua nada dalam suatu musik. Setiap nada dalam tangga nada kromatik memiliki frekuensi tersendiri. Terdapat dua belas nada dalam satu oktaf. Namun penamaannya hanya menggunakan huruf A sampai G untuk tujuh nada. Sedangkan lima nada yang lain menggunakan tanda kres (#) atau tanda mol (b) setelah notasi nada. Tangga nada kromatik dapat dibuat dengan kombinasi atau menghilangkan beberapa nada. Tabel 2.1 menunjukkan frekuensi dua belas nada dalam satu oktaf [4].

Tabel 2.1 Tangga Nada Kromatik Dalam Musik [4]

Tangga Nada Kromatik	
A	440,00 Hz
A#/Bb	466,16 Hz
B	493,88 Hz
C	523,25 Hz
C#/Db	554,37 Hz
D	587,33 Hz
D#/Eb	622,25 Hz
E	659,25 Hz
F	698,46 Hz

F#/Gb	739,99 hz
G	783,99 Hz
G#/Ab	830,61 Hz
A	880,00 Hz





$$semitone = 69 + 12.^{2}\log(\frac{frequency(Hz)}{440})$$

2.1.2 Melodi

Melodi merupakan rangkaian nada-nada yang dimainkan sehingga membentuk makna tersendiri bagi seseorang. Melodi merupakan bagian dari lagu yang paling mempengaruhi perasaan manusia. Dalam suatu lagu melodi merupakan bagian yang paling utama. Melodi dalam suatu lagu biasanya selalu berulang dalam beberapa waktu. Orang lebih mudah mengingat melodi daripada lirik suatu lagu. Melodi merupakan *perceptual feature* dari suatu lagu. Persepsi melodi seseorang terhadap lagu berbeda-beda.

2.2 Sistem Query By Singing/Humming

2.2.1 Music Information Retrieval

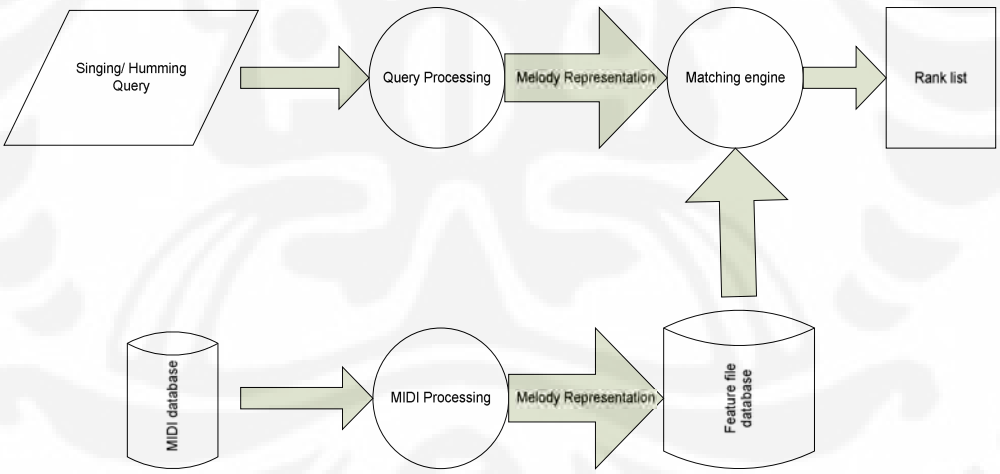
Music Information Retrieval (MIR) merupakan sesuatu yang berkaitan dengan representasi, penyimpanan, pengorganisasian, dan perolehan informasi dari musik itu sendiri. MIR bertujuan untuk memberikan kemudahan dalam mengakses dan menikmati suatu musik. MIR memiliki asosiasi di tingkat internasional yaitu MIREX (*Music Information Retrieval Exchange*). MIREX merupakan wadah bagi peneliti yang konsen di bidang musik yang ingin menunjukkan hasil karyanya. Setiap tahun MIREX mengadakan kompetisi di bidang MIR yang bertujuan memacu inovasi peneliti sehingga semakin berkembang dan bermanfaat.

Perkembangan teknologi di bidang MIR sudah semakin canggih. Terdapat berbagai bidang yang menjadi fokus pada penelitian MIR mulai dari *mood detection* dimana seseorang dapat mengklasifikasi suatu lagu berdasarkan *mood* yang terkandung dalam lagu tersebut. Sehingga apabila seseorang sedang dalam kondisi sedih maka dapat dengan mudah memutar lagu yang tepat. Disamping itu terdapat bidang *chord detection* yang dapat digunakan untuk membantu menganalisis suatu jenis musik sehingga dapat diambil karakteristik musik tersebut.

Bahkan terdapat bidang penelitian MIR yang sedang berkembang dan sangat menarik perhatian peneliti yaitu sistem pencarian lagu berdasarkan melodinya atau dikenal dengan sistem *query by singing/humming* (QbSH). Dahulu, agar bisa mendengarkan lagu orang harus mengetahui meta-data dari lagu tersebut seperti judul lagu, penyanyi, jenis musik (jazz, pop, rock, klasik dll) atau liriknya. Namun dengan adanya sistem QbSH seseorang dapat dengan mudah mencari lagu yang diinginkan hanya dengan menyenandungkan (na-na-na atau la-la-la) potongan melodi dari lagu tersebut. Dengan mencocokkan representasi melodi maka akan dicari lagu yang paling sesuai dengan melodi yang disenandungkannya.

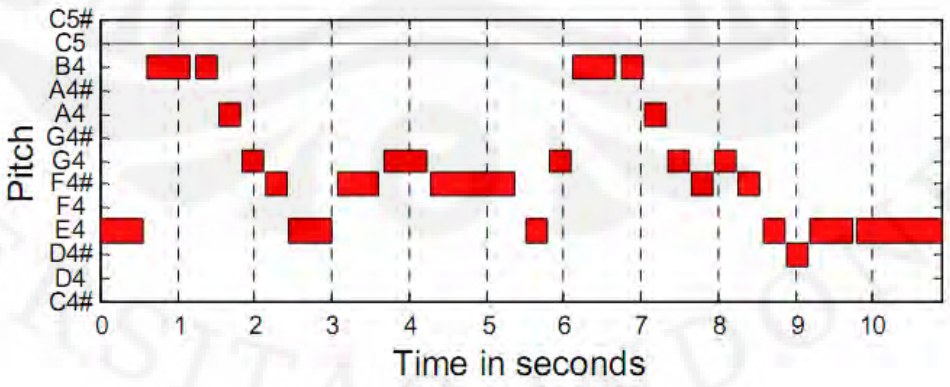
2.2.2 Prinsip Dasar Sistem Query By Singing/Humming

Sistem *Query by Singing/Humming* (QbSH) bekerja dengan cara mencocokkan representasi melodi *query* (masukan berupa senandungan) dengan representasi melodi lagu pada *database.* Terdapat beberapa tahapan penting dalam pembuatan sistem QbSH seperti ditunjukkan pada Gambar 2.3 berikut [3].



Gambar 2.3 Diagram Blok Sistem QbSH [3]

Tahapan pertama yang harus dilakukan adalah membuat *database* lagu . Selanjutnya lagu yang terdapat dalam *database* diproses (ekstraksi fitur) untuk diambil representasi melodinya dan hasil ekstraksi fiturnya disimpan dalam satu *feature file database.* Tahap kedua adalah memproses *query* baik secara *real-time*


C5#
C5
B4
A4#
A4
G4#
G4
F4#
F4
E4
D4#
D4
C4#
Pitch
0
1
2
3
4
5
6
7
8
9
10
Time in seconds

MIDI dapat berupa *polyphonic* maupun *monophonic* tergantung pada banyaknya *channel*. *Polyphonic* MIDI memiliki 16 *channel* yang terdiri dari *channel* bass, drum, gitar, piano, violin, vokal dll. Sedangkan *monophonic* MIDI hanya memiliki *channel* tunggal yaitu dapat berupa salah satu dari 16 *channel* tersebut misalnya vokal atau piano. Parameter MIDI tersebut diekstraksi dan akan diambil fitur representasi melodinya. Selanjutnya parameter itu akan disimpan dalam satu *feature file database* yang merupakan suatu matriks baris yang berisi kumpulan lagu dengan parameter yang terkandung dalam lagu tersebut. Sehingga pengambilan fitur representasi melodi pada *database* dapat dengan mudah dilakukan. Fitur representasi melodi ini yang akan dicocokkan dengan representasi melodi *query* menggunakan metode *melody matching.*

2.2.2.2 Pemrosesan *Query*

Format audio dari *query* yang sering digunakan adalah ".wav". Karena merupakan sinyal akustik yang berasal dari suara manusia. Selanjutnya sinyal ".wav" tersebut akan diekstraksi beberapa jenis fitur yang dapat digunakan sebagai representasi melodi. Ekstraksi fitur pada *query* berbeda dengan ekstraksi fitur pada MIDI. Hal ini dikarenakan sinyal akustik *query* masih berbentuk sinyal analog (lebih kompleks). Oleh karena itu, sebelum ekstraksi fitur maka *query* akan mengalami tahap *pre-processing* yaitu digitalisasi sinyal agar lebih memudahkan ekstraksi fitur. Setelah itu dilakukan pemrosesan *query* untuk mendapatkan representasi melodi yang terdiri dari dua tahap yaitu:

a. *Pitch Tracking*

Pitch tracking adalah suatu metode untuk mendapatkan frekuensi dasar (*fundamental frequency*) f_0 dari suatu sinyal suara yang akan menjadi fitur representasi melodi. Beberapa metode yang digunakan pada *pitch tracking* dibagi berdasarkan: *Time Domain Method, Frequency Domain Method, dan Auditory Model-Based Method. Time Domain Method* merupakan metode yang paling sering digunakan untuk mengestimasi f_0 pada sinyal suara. Beberapa fungsi yang dapat digunakan pada metode ini antara lain, ACF (*Autocorrelation Function*), AMDF (*Average Magnitude Difference Function*), dan SITF (*Simpel Inverse*

Filter Tracking). Pada *Time Domain Method* sinyal *query* akan langsung diproses tanpa harus ditransformasikan dari domain waktu ke domain lain.

Pada *Frequency Domain Method* sinyal *query* harus ditransformasikan terlebih dahulu dari domain waktu kedalam domain frekuensi dengan menggunakan transformasi fourier. *Frequency Domain Method* merupakan metode dengan menggunakan *two-way mismatch procedure*. Sedangkan untuk metode *Auditory Model-Based Method,* konsep dasarnya adalah memodelkan sinyal masukan menjadi dalam bentuk respon frekuensi dari telinga manusia. Kelemahan dari metode ini adalah tidak cocok dipakai untuk sinyal yang memiliki *pitch* yang cukup tinggi. Konsep ini lebih sering dipakai pada aplikasi *speech recognition* [8].

b. *Note Segmentation*

Note merupakan urutan *pitch* dengan nilai yang sama dalam suatu waktu. Suatu *note* mengandung *pitch* dan durasi lamanya *note* tersebut. Jumlah banyaknya *pitch* dalam suatu *note* berbeda-beda. Tergantung durasi lamanya *note* tersebut. Semakin lama durasi *note* tersebut maka semakin banyak jumlah *pitch* dalam *note* tersebut. *Note segmentation* adalah suatu metode untuk mensegmentasi *note* dari suatu *query*. Tujuannya adalah untuk mendapatkan fitur (*pitch* ataupun durasi) dari suatu *note*. *Note segmentation* berkaitan erat dengan *onset* dan *offset detection*. *Onset* adalah waktu dari mulainya suatu *note*, sedangkan *offset* adalah waktu dari berakhirnya suatu *note*. *Note segmentation* juga berfungsi sebagai penanda apakah dalam suatu segmen signal tertentu berada dalam keadaan *silence*, terkena *noise*, atau terdapat suara. *Note segmentation* dapat dilakukan dengan dua cara, yaitu berdasarkan *Amplitude* dan berdasarkan *Pitch* [8].

Segmentasi berdasarkan *amplitude* adalah membagi segmen dengan melihat *amplitude* sinyal dalam domain waktu. Proses segmentasi ini simple dan sangat mudah diimplementasikan. Namun segmentasi ini tidak cocok jika digunakan untuk kondisi real-time. Selain itu, jika ingin menggunakan segmentasi jenis ini maka setiap *user* yang ingin mencari lagu harus menyisipkan suatu tanda (“da”

atau “ta”) diantara *note* yang disenandungkannya. Hal ini tentunya akan merepotkan *user* yang akan melakukan *query* [8].

Segmentasi berdasarkan *pitch* adalah membagi segmen dengan melihat perubahan dari pitch suatu sinyal. Untuk menentukan perubahan pitch ini diperlukan suatu algoritma tertentu. Salah satu algoritma untuk menentukan perubahan pitch ini adalah *sliding window method*. Segmentasi dengan menggunakan pitch sangat cocok jika digunakan untuk kondisi *real-time* dan tidak mempedulikan cara bersenandungnya [8].

2.2.2.3 Pencocokkan Melodi

Potongan melodi dari *query* yang disenandungkan akan dicocokkan dengan lagu yang ada di *database*. Terdapat berbagai macam metode yang dapat digunakan untuk mencocokkan representasi melodi diantaranya adalah *Linear Scaling, Dynamic Time Wrapping* atau *Hidden Markov Model. Linear scaling* merupakan salah satu teknik yang mudah untuk membandingkan *pitch* input dengan lagu *database*. Konsep dasarnya adalah dengan menekan atau meregangkan panjang *pitch* input dengan skala tertentu agar memiliki panjang *pitch* yang sama dengan lagu *database* [3].

Dynamic Time Wrapping (DTW) merupakan suatu teknik suatu teknik untuk mengurangi nilai ketaksamaan antara dua *template*/*sequence* yang disebabkan oleh pergeseran waktu [8]. Atau dapat dikatakan DTW adalah teknik untuk mengarahkan dua *senquence* yang bersifat *time dependent* sehingga dapat dihitung jarak terpendek diantara keduanya. Terdapat banyak jenis tipe DTW yang ditentukan berdasarkan *step size* yang digunakan. Penentuan *step size* akan berpengaruh terhadap hasil dan kecepatan proses komputasi dari perhitungan sast pencocokkan. DTW memiliki tingkat keakuratan yang tinggi, namun membutuhkan proses komputasi yang lama. Sehingga DTW kurang cocok jika sistem QbSH dengan menggunakan *database* dalam jumlah besar.

Hidden Markov Model (HMM) merupakan suatu teknik membentuk model statistik berdasarkan prinsip probabilitas. HMM banyak digunakan pada sistem pengenalan suara. Dalam HMM urutan suatu kejadian dapat diramalkan dengan

menggunakan suatu model hasil *training* (pembelajaran) yang telah dilakukan. HMM merupakan bentuk perluasan dari *Markov Chain* (rantai markov) dimana pada HMM terdapat parameter-parameter yang disembunyikan (*hidden*). Konsep utama HMM adalah mencari nilai probabilitas maksimum dari urutan suatu kejadian sehingga bisa dikenali.

2.3 *Auto-Correlation Function* (ACF)

Auto-Correlation Function (ACF) merupakan suatu teknik yang digunakan untuk mencari frekuensi dasar *query* berdasarkan *time domain method.* ACF memiliki beberapa kelebihan diantaranya [9]:

1. *Simple*

   Prinsip kerja ACF sangat sederhana yaitu dengan mengalikan sinyal asli dengan sinyal yang sudah mengalami pergeseran. Sehingga didapatkan periode dasar dari sinyal tersebut.

2. *Fast*

   Proses pengambilan *pitch* ini tidak membutuhkan waktu yang lama. Karena termasuk dalam *time domain method* dimana sinyal langsung dapat dianalisis tanpa perlu ditransformasikan dari domain waktu ke dalam domain yang lain.

3. *Low-Mid Frequency*

   Metode ini cocok digunakan untuk pengambilan frekuensi suara manusia.

Metode ACF merupakan bentuk turunan dari metode *cross correlation.* Prinsip kerja dari metode *cross correlation* adalah mengalikan sinyal satu dengan sinyal lain. Sedangkan pada metode *autocorrelation function* (ACF) sinyal satu merupakan sinyal asli dan sinyal lain merupakan sinyal itu yang sudah mengalami pergeseran waktu. Kemudian sinyal ini akan diukur korelasinya untuk mendapatkan periode dasar dari sinyal asli. Sinyal input disegmentasi ke dalam beberapa *frame*.

Ukuran *frame* yang digunakan tergantung dari rentang frekuensi dasar yang ingin diperoleh. Pada sistem QbSH ini akan diambil frekuensi dari 62,5 –

2000 Hz. Ukuran *frame* dapat dinyatakan dalam jumlah *sample point* dan waktu. Syarat dari suatu ukuran *frame* agar dapat dilakukan proses *pitch tracking* adalah minimal harus memuat dua kali besar dari periode dasar (*fundamental periode*). Periode dasar adalah sama dengan frekuensi *sampling* dibagi dengan frekuensi dasar. Batas atas ukuran *frame* ditentukan oleh batas bawah dari frekuensi dasar. Berikut ini adalah perhitungan ukuran *frame* yang digunakan [6] :

$$frame_{max} = 2.\frac{f_s}{fo_{min}} \tag{2.2}$$

$$frame_{max} = 2.\frac{f_s}{fo_{min}} = 2.\frac{8000\ Hz}{62{,}5\ Hz} = 256\ sample\ point \tag{2.3}$$

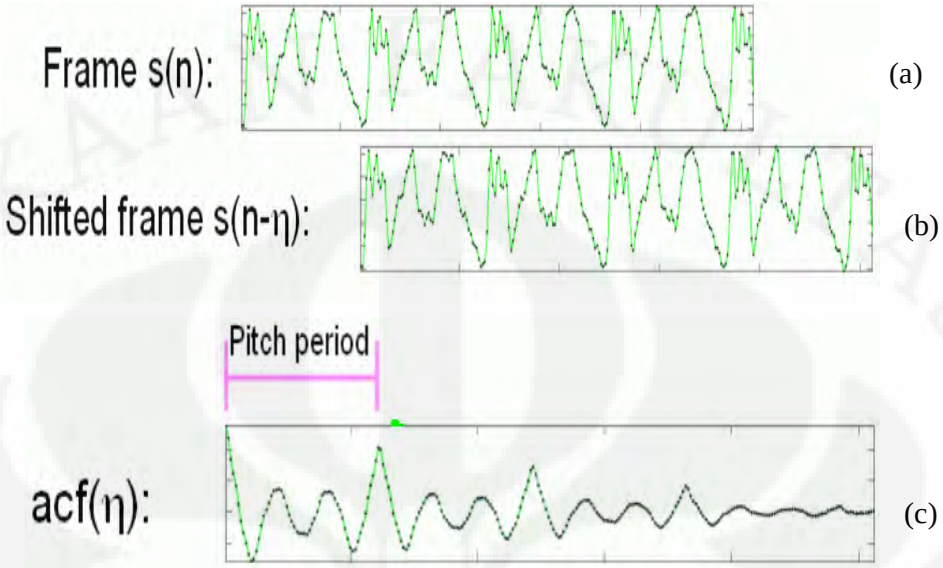
Dengan menggunakan frekuensi minimal sebesar 62,5 Hz maka *frame* yang dapat digunakan adalah minimal mengandung 256 *sample point*.

Setelah sinyal terbagi menjadi beberapa *frame*, besar dari periode dasar setiap *frame* tersebut akan ditentukan dengan cara mengukur korelasi antara sinyal awal dengan sinyal itu yang telah mengalami pergeseran. Dengan mengetahui periode dasar sinyal tersebut maka dapat ditentukan frekuensi dasar sinyal itu yang merupakan kebalikan dari periodenya. Seperti ditunjukkan pada Gambar 2.5 pencarian periode dasar dengan menggunakan ACF [6]. Proses mendapatkan periode dasar dengan menggunakan metode *autocorrelation function* adalah sebagai berikut [6]:

1. Sinyal s(n) tersebut digeser sebanyak η sehingga didapatkan sinyal s(n- η)
2. Menghitung ACF (A*utocorrelation Function*) untuk setiap nilai η yang dimulai dari satu sampai fungsi ACF menghasilkan nilai yang maksimal seperti ditunjukkan pada persamaan (2.4) [6].

$$ACF(\eta) = \sum_{n=1}^{t+frame-1} S(n).S(n-\eta) \tag{2.4}$$

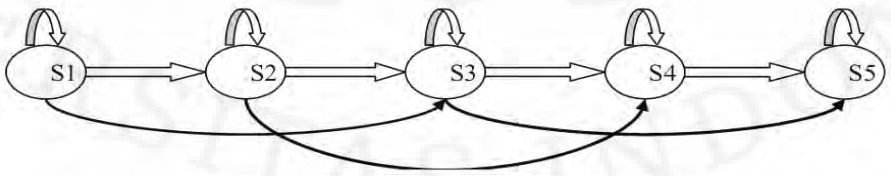
3. Selisih antara nilai τ yang menghasilkan *local maximum* dari fungsi ACF merupakan besar dari periode dasar dari sinyal tersebut.
4. Setelah mendapatkan periode dasar maka akan didapatkan besar frekuensi dasar yang merupakan pitch pada *frame* tersebut.



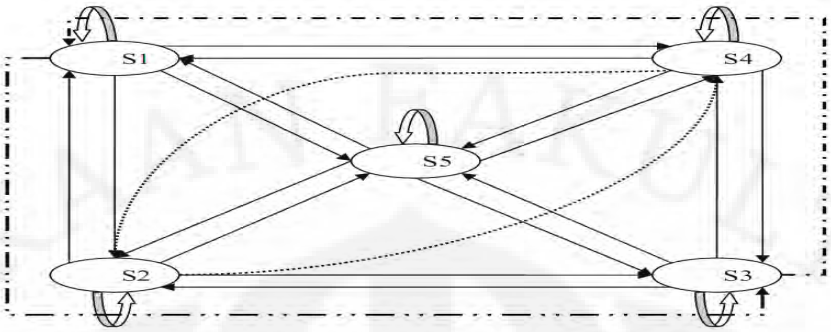
Gambar 2.5 Pencarian Periode Dasar dengan Menggunakan ACF (a) sinyal satu *frame* (b) Sinyal s(n) digeser sebanyak η (c)sinyal hasil ACF [9]

2.4 *Hidden Markov Model* (HMM)

Hidden Markov Model (HMM) adalah suatu model statistik yang dapat digunakan untuk menguraikan urutan waktu diskrit berdasarkan prinsip probabilitas. Model statistik tersebut didapatkan dari hasil *training* (pembelajaran) yang telah dilakukan dan digunakan untuk meramalkan urutan suatu kejadian. HMM merupakan bentuk perluasan dari proses Markov. Proses Markov didefinisikan sebagai proses stokastik dimana distribusi probabilitas transisi dari *state* satu ke *state* yang lain hanya bergantung pada kondisi *state* tersebut . Proses Markov dapat dimodelkan dalam suatu rantai Markov. Terdapat dua bentuk model rantai Markov yaitu model *ergodic* dan model *left-right.* Gambar 2.6 memperlihatkan bentuk model rantai markov 5 *state* [10].



(a)



$$\sum_{j=1}^{N} aij$$

$$A=a_{ij}=\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix} \quad (2.8)$$

4. Probabilitas distribusi B yang merupakan probabilitas distribusi simbol yang diamati dalam suatu *statej*. Probabilitas distribusi simbol B dalam suatu *state j* dilambangkan B={$b_j(k)$}, dimana

$$b_j(k)=P[v_k \text{ saat } t|qt=Sj], \quad 1\le j \le N, \quad 1\le k \le M \quad (2.9)$$

Probabilitas distribusi B dapat dinyatakan dalam matriks kolom Nx1. Dimana N menunjukkan jumlah state yang ada. Jika terdapat n buah *state,* maka akan terbentuk matriks B sebagai berikut :

$$B=\begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{pmatrix} \quad (2.10)$$

5. Probabilitas distribusi awal $\pi=\{\pi_i\}$, dimana

$$\pi_i=P[q_1=S_1], \quad 1\le i \le N \quad (2.11)$$

Probabilitas distribusi awal π juga dinyatakan dalam suatu matriks kolom Nx1. Seperti halnya probabilitas distribusi B, jika terdapat n buah *state* maka akan terbentuk matriks π sebagai berikut :

$$\pi=\begin{pmatrix} c_1 \\ c_2 \\ . \\ . \\ c_n \end{pmatrix} \quad (2.12)$$

Terdapat dua jenis tipe HMM yaitu *Discrete HMM* (DHMM) dan *Continuous HMM* (CHMM) [10]. Pada DHMM elemen symbol yang diamati

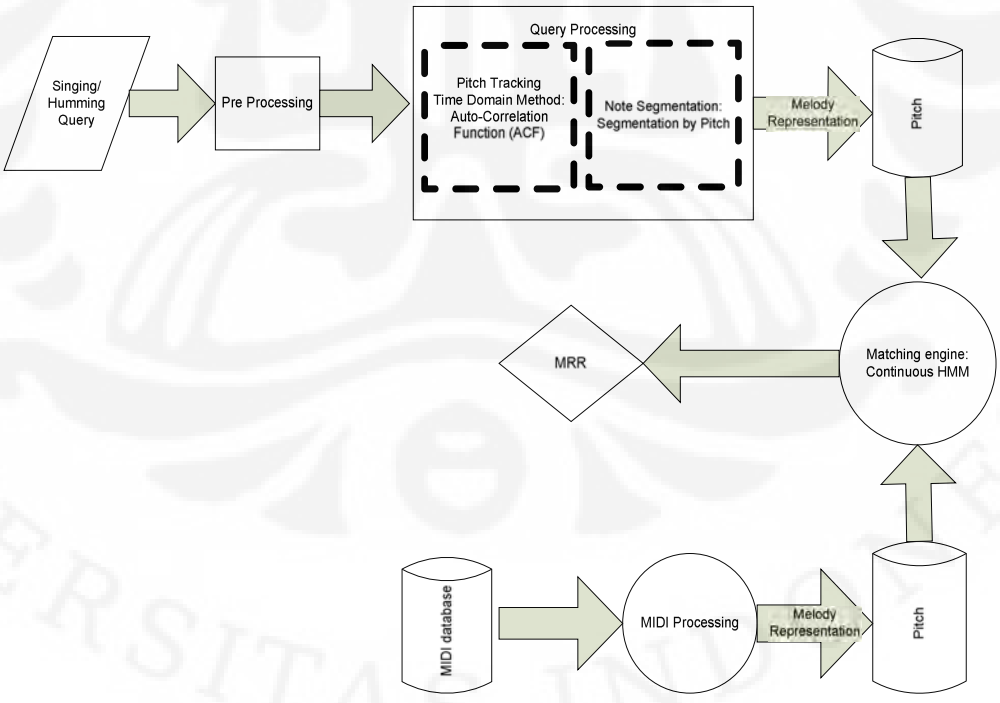
(*observer state*) dikuantisasi dalam suatu vektor kuantisasi *codebook*. Disamping itu sebelum dilakukan kuantisasi vector diperlukan labelisasi dimana tiap suara diberi suatu label dan hasilnya dikumpulkan dalam suatu *database* label [11]. Sedangkan pada CHMM tidak membutuhkan kuantisasi vektor dan labelisasi. Baik DHMM maupun CHMM membutuhkan dua tahapan yaitu tahap *training* (pembelajaran) dan *test* (pengujian). Tahap *training* dilakukan untuk mencari suatu model λ yang akan digunakan untuk meramalkan suatu probabilitas urutan kejadian. Sedangkan tahap *test* digunakan untuk melakukan pengujian pengenalan sistem.

BAB 3

PERANCANGAN SIMULASI SISTEM QUERY BY SINGING/HUMMING

3.1 Perancangan Simulasi Sistem Query By Singing/Humming

Perancangan simulasi sistem *query by singing/humming* untuk musik dangdut ini menggunakan piranti perangkat lunak Matlab 7.1.0.246(R14). Terdapat tiga proses utama dalam simulasi sistem QbSH ini yaitu pembuatan *database*, ekstraksi fitur *query* dan pencocokkan representasi melodi. Proses pembuatan *database* lagu dangdut dari *polyphonic* menjadi *monophonic* MIDI dilakukan secara manual menggunakan piranti perangkat lunak Guitar Pro 5. Ekstraksi fitur *query* menggunakan *file-file* simulasi dari Roger Jang [12]. Untuk perekaman *query* (*real-time* maupun *non real-time*) dilakukan dengan menggunakan *microphone.* Sedangkan proses pencocokkan melodi dilakukan dengan menggunakan *source code* GpdsHMM Toolbox dari *Universidad de Las Palmas de Gran Canaria Campus de Tafira* yang telah dimodifikasi sebelumnya.



Gambar 3.1 Perancangan Simulasi Sistem QbSH

Perancangan simulasi sistem QbSH untuk musik dangdut ditunjukkan pada Gambar 3.1 diatas. Pada perancangan simulasi sistem QbSH diatas tahap awal yang harus dilakukan adalah membuat suatu *database* lagu *monophonic* MIDI yang mengandung *channel vocal* atau melodi. Dan membuatnya menjadi satu *feature file database* untuk diekstraksi representasi melodinya berupa *pitch.* Setelah itu dilakukan proses pembelajaran (*training*) terhadap *database* lagu tersebut menggunakan *Hidden Markov Model* (HMM) untuk mencari model lambda (λ) yang optimum dari representasi melodi tersebut.

Selanjutnya *user* melakukan *query* berupa senandungan dari potongan melodi lagu yang ada di *database* untuk menguji tingkat keakuratan pengenalan sistem QBSH yang diukur dari besarnya nilai MRR (*Mean Reciprocal Rank*). Sinyal suara (*query*) ini direkam menggunakan *microphone* selama 10 detik dengan tipe *channel* mono. Proses ini dapat dilakukan baik secara *real-time* ataupun *non real-time*. Setelah itu *query* akan mengalami tahap *pre-processing* dimana sinyal (*query*) akan mengalami digitalisasi untuk mempermudah pengambilan *pitch.* Sinyal akan dicuplik dengan frekuensi sampling 8.000 Hz dengan 8 bit tiap sampel poin. Selanjutnya sinyal akan dibagi menjadi beberapa frame.

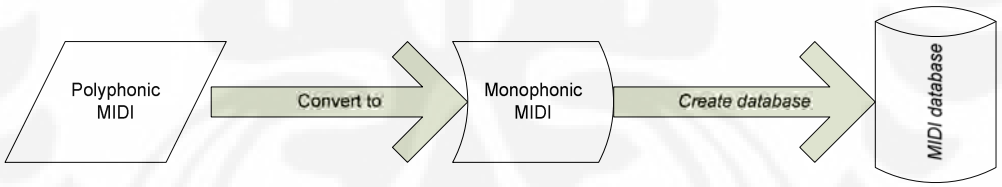
Setelah mengalami digitalisasi maka akan dilakukan pemrosesan *query*. Tujuannya adalah untuk mendapatkan representasi melodi dari *query*. Proses ini terdiri dari dua tahap utama yaitu *pitch tracking* dan *note segmentation. Pitch tracking* dilakukan untuk mencari frekuensi dasar dari *query* yang dinyanyikan atau disenandungkan. Teknik *pitch tracking* yang digunakan adalah berdasarkan *time domain* method yaitu *Auto-Correlation Function* (ACF). Teknik ACF digunakan untuk mendapatkan *pitch vector* dari *query*. *Pitch vector* merupakan rangkaian kumpulan *pitch.* Setelah didapatkan *pitch vector* dari *query* maka dilakukan proses *note segmentation.* Proses *note segmentation* bertujuan untuk mendapatkan urutan *note* (melodi) dari suatu sinyal audio dengan cara menggabungkan *pitch vector* yang memiliki kemiripan. Sebelum dilakukan *note segmentation* kondisi *silent* terlebih dahulu dihilangkan dengan menggunakan *rest handle* [12]. Setelah itu baru dilakukan *note segmentation* dengan menggunakan

segmentation by pitch. Suatu *note* memiliki urutan nilai *pitch vector* yang sama. Sehingga apabila sinyal tersebut mengalami perubahan *pitch vector* maka akan disegmentasi. Selanjutnya *note* hasil segmentasi tersebut hanya diambil *pitch*-nya saja.

Tahap berikutnya adalah sistem QbSH akan melakukan proses pencocokkan antara representasi melodi (*pitch*) lagu pada *database* dengan *query*. Sistem akan mencari kemiripan potongan representasi melodi *query* dengan lagu dan menampilkan urutannya. Lagu yang berada di urutan pertama dan seterusnya merupakan lagu yang memiliki kemiripan representasi melodi dengan *query* paling tinggi. Tahap yang terakhir adalah menghitung MRR dimana merupakan nilai yang menyatakan tingkat keakuratan sistem QbSH.

3.2 Proses Pembuatan Database

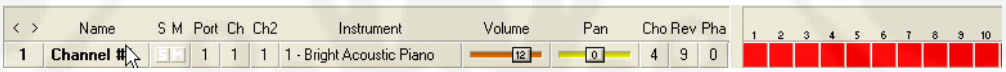
Proses pembuatan *database* MIDI dilakukan dalam dua tahap diantaranya adalah pengubahan *polyphonic* menjadi *monophonic* dan pembuatan *database* MIDI. Gambar 3.2 memperlihatkan proses pembuatan *database* lagu.



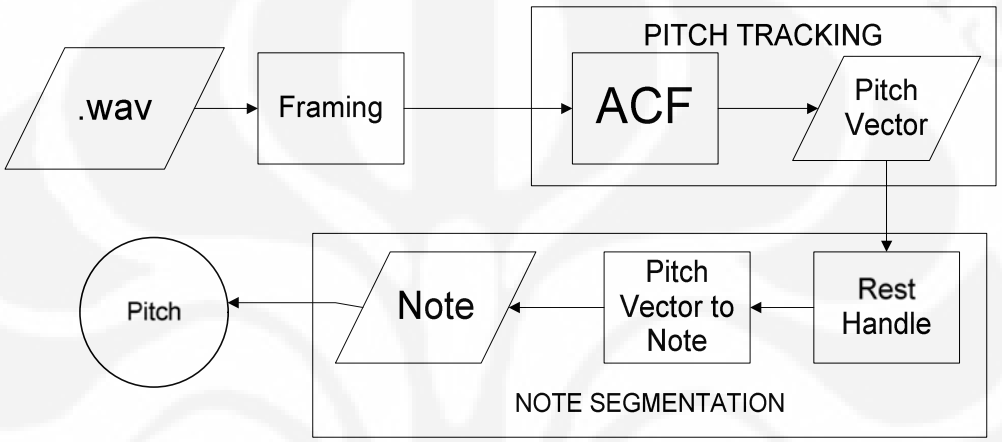
Gambar 3.2 Proses Pembuatan *Database*

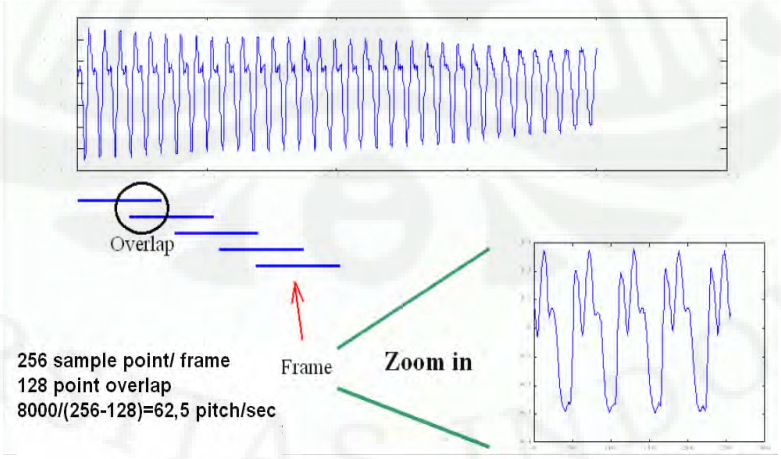
Sebelum *database* MIDI dibuat maka tahap awal yang harus dilakukan mengubah format MIDI dari *polyphonic* menjadi *monophonic*. Pada umumnya lagu MIDI yang ada sekarang merupakan *polyphonic*. Sehingga harus diubah terlebih dahulu menjadi *monophonic*. Dan lagu MIDI tersebut harus mengandung *channel* melodi atau *vocal*. Tujuannya adalah untuk mempermudah ekstraksi parameter dan pembuatan *database*. Hal ini disebabkan karena hanya mengandung satu *channel* sehingga lebih mudah untuk diproses. MIDI dangdut yang digunakan pada skripsi

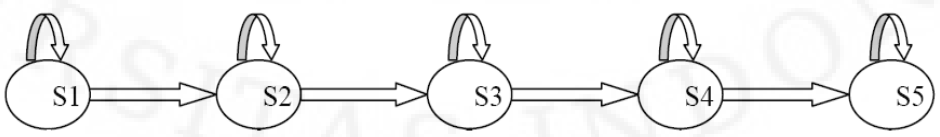
	Name	S	M	Port	Ch	Ch2	Instrument	Volume	Pan	Cho	Rev	Pha
1	Channel #1			1	1	1	1 - Bright Acoustic Piano	12	0	4	9	0
2	Channel #2			1	2	2	35 - Fretless Bass	12	0	0	0	0
3	Channel #3			1	3	3	27 - Electric Guitar (clean)	12	0	2	9	0
4	Channel #4			1	4	4	48 - String Ensemble 1	16	0	0	12	0
5	Channel #5			1	5	5	82 - Lead 3 (calliope lead)	12	0	0	12	0
6	Channel #7			1	6	6	65 - Alto Sax	12	0	2	15	0
7	Channel #8			1	7	7	11 - Vibraphone	12	0	0	12	0
8	Channel #9			1	8	8	126 - Applause	12	0	0	12	0
9	Channel #10			1	9	9	25 - Acoustic Guitar (steel)	12	0	0	0	0



	Name	S	M	Port	Ch	Ch2	Instrument	Volume	Pan	Cho	Rev	Pha
1	Channel #1			1	1	1	1 - Bright Acoustic Piano	12	0	4	9	0


PITCH TRACKING
.wav
Framing
ACF
Pitch Vector
Pitch
Note
Pitch Vector to Note
Rest Handle
NOTE SEGMENTATION


Overlap
256 sample point/ frame
128 point overlap
8000/(256-128)=62,5 pitch/sec
Frame
Zoom in


S1
S2
S3
S4
S5

$$A=a_{ij}=\begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & a_{55} \end{pmatrix}$$

Gambar 3.8 Matriks Transisi A

Pada sistem QbSH ini sebuah *note* direpresentasikan sebagai sebuah *state.* Dan setiap *state* dimodelkan kedalam suatu *probability density function* (PDF) distribusi *Gaussian* yang mengandung suatu *mean* dan *variance.* Persamaan (3.1) memperlihatkan fungsi PDF distribusi *Gaussian.*

$$g(x,\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \quad (3.1)$$

Dimana *x* adalah nilai *pitch* yang diamati (*observer*), μ dan σ^2 merupakan *mean* (rata-rata) dan *variance* dari suatu PDF. Sehingga untuk nilai *x* yang diberikan dari fungsi $g(x,\mu_j,\sigma^2_j)$ untuk *state j* didapatkan probabilitas *state* yang dinotasikan dengan $p_s(x,j)$. Dalam aplikasi sistem QbSH digunakan "*sum log probability"* sebagai ganti dari "*the product of probability"* seluruh *state* dalam satu lagu. Dan probabilitas transisi dari *state i* ke *j* dinotasikan $p_t(x,j)$ [3].

$$0 \le p_t(i,j) \le 1, \forall i,j$$

$$p_t(i,j) = 0, if\ j < i\ or\ j > i+1$$

$$p_t(n,n) = 1$$

$$p_t(i,i) + p_t(i,i+1) = 1, \forall i \quad (3.2)$$

Dimana n menunjukkan banyaknya *state* dan $p_t(n,n) = 1$ mengindikasikan pada *state* terakhir hanya terjadi transisi di *state* itu sendiri.

3.4.1 Proses Pembelajaran

Proses pembelajaran (*training*) dilakukan untuk mendapatkan model λ=(A,B,π) yang akan digunakan untuk meramalkan probabilitas urutan kejadian.

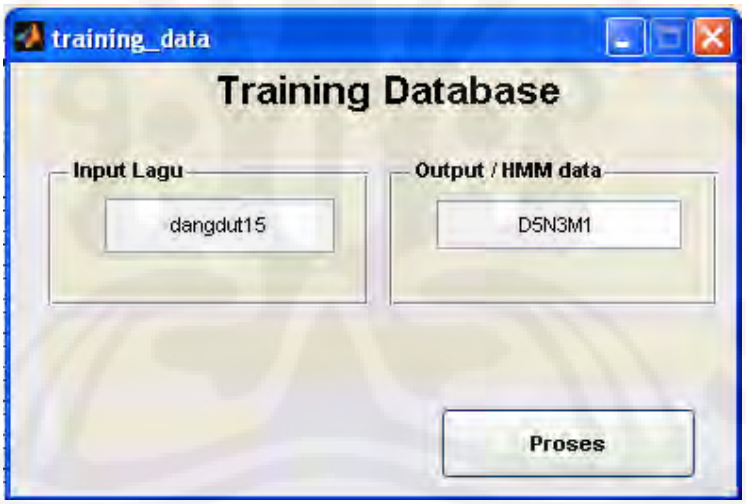
$\bar{\lambda}$

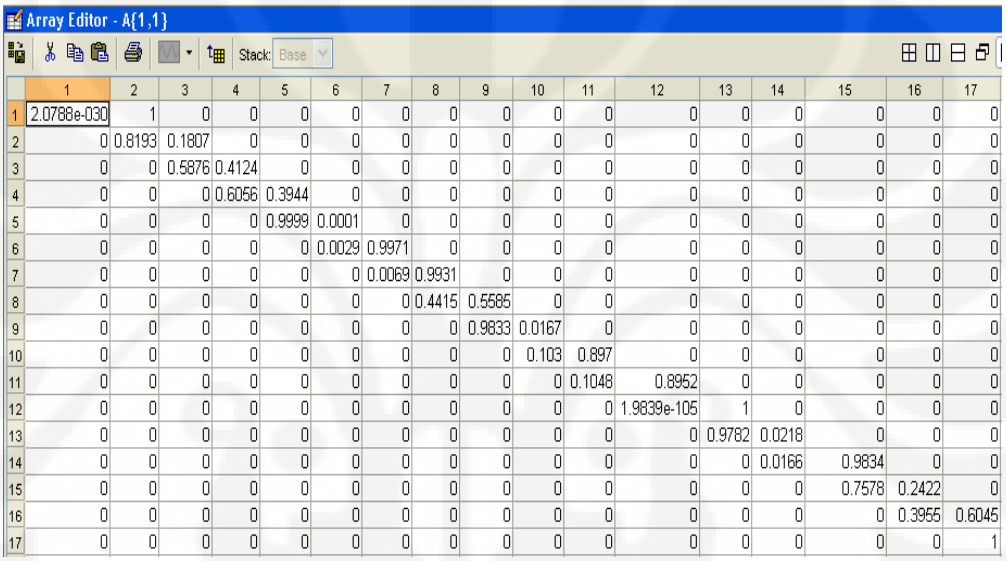
$\bar{\lambda}$ $\bar{A}$ $\bar{B}$, $\bar{\pi}$

$\bar{\lambda}$

$P(\mathbf{O}|\bar{\lambda}) - P$ $\mathbf{O}$ λ

P $\mathbf{O}$ $\bar{\bar{\lambda}}$





	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2.0788e-030	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0.8193	0.1807	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0.5876	0.4124	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0.6056	0.3944	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0.9999	0.0001	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0.0029	0.9971	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0.0069	0.9931	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0.4415	0.5585	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0.9833	0.0167	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0.103	0.897	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0.1048	0.8952	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1.9839e-105	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0.9782	0.0218	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0166	0.9834	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7578	0.2422	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3955	0.6045
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1


QBH_Sys
Qbh System
Database HMM
D5N3M1
Database Musik
dangdut15
===> 4.78 sec
Live Recording
Proses
Hasil Pencarian
01. kopi-dangdut
02. Colak-Colek
03. Adu-buyung
04. Darah-Muda
05. Syahdu
06. Kuch-Kuch-Hotahei
07. Penasaran
08. ang-toyib
09. mbah-dukun
10. sekuntum-mawar-merah

3.5 Proses Penghitungan MRR

Hasil pencocokkan melodi yang ditampilkan dalam urutan lagu diukur dalam suatu nilai yang disebut MRR [1]. MRR (*Mean Reciprocal Rank*) suatu nilai yang menyatakan ukuran keakuratan sistem QbSH. Besarnya nilai MRR dapat dirumuskan dengan persamaan berikut [1]:

$$MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{r_i} \tag{3.3}$$

Dimana, r_i adalah peringkat dari lagu yang tepat untuk *query* ke-i untuk jumlah *query* yang dilakukan sebesar n. Disamping MRR parameter yang menunjukkan keberhasilan sistem QbSH adalah waktu yang dibutuhkan selama proses pencocokkan. Hal ini erat kaitannya dengan penggunaan *database* dalam skala besar.

BAB 4

HASIL UJI COBA DAN ANALISIS SIMULASI SISTEM QbSH

4.1 Hasil Uji Coba Simulasi

Simulasi pada sistem QbSH ini adalah bagaimana mengenali lagu dari *query* yang disenandungkan *user* hanya dalam 10 detik. Lagu yang dikenali terbatas pada 15 lagu. Pengujian sistem QbSH dilakukan dengan mengambil sampel *query* dari 6 orang yang terdiri dari 3 orang laki-laki dan 3 orang perempuan. Setiap orang menyenandungkan 10 lagu dari 15 lagu yang ada pada *database*. Sehingga terdapat 60 *query* yang akan diuji-cobakan pada sistem QbSH.

Pengujian dilakukan dengan cara mencocokkan representasi melodi (*pitch*) suara orang dengan lagu pada *database*. Terdapat perbedaan nilai *pitch* antara suara orang (*query*) yang berkisar antara 50-70 (dalam *semitone*). Sedangkan nilai *pitch* lagu memiliki nilai 69-85 (dalam *semitone*). Karena perbedaan besarnya nilai *pitch* ini, maka perlu dilakukan normalisasi. Sehingga besarnya nilai *pitch* setelah normalisasi baik *query* maupun lagu berada pada *range* yang tidak jauh berbeda. Oleh karena itu, pada sistem QbSH ini akan dilakukan pengujian terhadap jenis normalisasi dan durasi pengambilan *pitch* dari lagu dan perekaman *query*. Normalisasi yang digunakan adalah berdasarkan nilai maksimumnya, rata-ratanya, dan energinya. Sedangkan durasi pengambilan *pitch* dari suatu lagu dilakukan selama 5, 8 dan 10 detik.

Tabel 4.1 Contoh Hasil Uji Coba Perbandingan Jenis Normalisasi

Gender	Type	Song														
LK 1	Norm	AB	DM	GD	J	JB	KD	KK	MD	SM	SMS	PS	BT	CC	MJ	SD
	1	7	4	2	9	6	13	4	8	1	13	2	12	4	15	15
	2	2	12	6	4	6	3	7	1	11	4	4	9	6	14	15
	3	4	1	10	6	6	1	1	2	10	15	3	5	14	13	14

Berikut adalah keterangan hasil uji coba perbandingan jenis normalisai simulasi sistem QbSH . Pada Tabel 4.1 terdapat 3 bagian kolom utama yang dapat dijelaskan sebagai berikut :

1. *Gender*

   Kolom ini Menunjukkan jenis kelamin dari orang yang menyenandungkan *query*. Terdapat 3 orang laki-laki (LK1, LK2 dan LK3) dan 3 Orang perempuan (PR1, PR2 dan PR3).

2. *Type*

   Kolom ini menunjukkan jenis normalisasi yang digunakan yaitu Normalisasi 1 (energi), Normalisasi 2 (maksimum) dan Normalisasi 3 (rata-rata).

   - Normalisasi 1 (energi)

     Setiap *pitch* dibagi dengan nilai energi ($\hat{x}$) dari kumpulan *pitch* tersebut.

     $$\hat{x} = \frac{\sqrt[2]{\sum f(x)^2}}{\sum f} \quad (4.1)$$

   - Normalisasi 2 (maksimum)

     Setiap *pitch* dibagi dengan nilai maksimum ($\dot{x}$) dari kumpulan *pitch* tersebut.

     $$\dot{x} = Xmaks \quad (4.2)$$

   - Normalisasi 3 (rata-rata)

     Setiap *pitch* dibagi dengan nilai rata-rata ($\bar{x}$) dari kumpulan *pitch* tersebut.

     $$\bar{x} = \frac{\sum f(x)}{\sum f} \quad (4.3)$$

3. *Song*

   Kolom ini menunjukkan judul lagu yang ada pada *database*. Nilai 1-15 menunjukkan urutan lagu yang dikenali dari *query* yang disenandungkan. Berikut adalah daftar singkatan lagu yang ada pada *database*.

   AB= Adu Buyung
   KK= Kuch-Kuch Hotahei
   DM= Darah Muda
   MD= Mbah Dukun

GD= Goyang Dombret
J = Jablai
JB = Jatuh Bangun
KD= Kopi Dangdut
CC= Colak-Colek
MJ = Mabuk Janda
SM = Sekuntum Mawar Merah
SMS= SMS
PS = Penasaran
BT = Bang Toyib
SD= Syahdu

Durasi pengambilan *pitch* merupakan waktu lamanya pengambilan *pitch* dari suatu lagu dalam *database*. Setiap lagu memiliki lama durasi yang berbeda-beda. Oleh karena itu, dilakukan pengujian terhadap waktu pengambilan *pitch* dari lagu. Sehingga semua lagu memiliki lama durasi pengambilan *pitch* yang sama.

4.1.1 Hasil Uji Coba Simulasi Pengenalan Lagu Saat Durasi 5 Detik

Setiap *user* menyenandungkan 10 *query* yang berbeda dari 15 lagu yang ada pada *database*. Setelah itu akan dilakukan pencocokkan *pitch* dengan menggunakan variasi normalisasi dan durasi pengambilan *pitch*. Dan sistem QbSH akan menampilkan hasil urutan judul lagu sesuai dengan besarnya *log probability*. Pada Tabel 4.2 dapat dilihat perbandingan jenis normalisasi saat durasi pengambilan *pitch* lagu dilakukan selama 5 detik dengan waktu perekaman *query* dilakukan selama 10 detik.

Tabel 4.2 Perbandingan Jenis Normalisasi Saat Durasi Pengambilan *Pitch* Lagu 5 Detik

Durasi 5 Detik											
LK 1	Norm	AB	DM	GD	J	JB	KD	KK	MD	SM	SMS
	1	7	5	3	9	6	13	4	3	1	13
	2	5	15	3	3	1	2	7	1	5	8
	3	2	1	15	4	1	1	8	2	8	15
LK 2	Norm	AB	CC	DM	J	JB	KD	KK	MJ	PS	SM
	1	1	1	6	5	6	1	1	10	6	6
	2	5	1	14	2	2	1	6	14	11	5
	3	1	1	1	6	1	1	7	7	12	8

LK 3	Norm	AB	BT	DM	GD	J	KD	MD	PS	SM	SMS
	1	4	10	3	5	5	2	1	9	9	13
	2	2	9	15	10	3	1	1	15	5	14
	3	2	14	1	8	3	4	1	10	9	7
PR1	Norm	AB	DM	GD	J	JB	KK	MD	PS	SM	SMS
	1	5	5	8	3	9	9	3	1	4	9
	2	1	13	3	4	2	1	1	9	2	14
	3	1	1	14	3	2	8	1	9	8	11
PR2	Norm	BT	CC	DM	GD	J	JB	KD	PS	SM	SMS
	1	12	3	14	8	1	11	3	4	3	12
	2	5	14	12	3	3	1	1	14	3	7
	3	14	1	4	4	2	3	1	10	8	11
PR3	Norm	CC	DM	GD	J	JB	KD	MD	PS	SM	SMS
	1	2	14	8	3	6	1	2	1	12	13
	2	10	12	10	2	5	1	1	3	3	14
	3	2	3	15	3	6	3	4	9	3	9

4.1.2 Hasil Uji Coba Simulasi Pengenalan Lagu Saat Durasi 8 Detik

Pada Tabel 4.3 dapat dilihat perbandingan jenis normalisasi pada saat durasi pengambilan *pitch* 8 detik.

Tabel 4.3 Perbandingan Jenis Normalisasi Saat Durasi Pengambilan *Pitch* Lagu 8 Detik

Durasi 8 Detik											
LK 1	Norm	AB	DM	GD	J	JB	KD	KK	MD	SM	SMS
	1	7	4	3	12	6	13	3	3	1	10
	2	2	10	7	4	4	3	1	8	7	9
	3	3	5	7	11	1	1	2	1	11	9
LK 2	Norm	AB	CC	DM	J	JB	KD	KK	MJ	PS	SM
	1	1	1	7	6	6	3	1	8	5	6

	2	2	6	10	2	1	2	1	5	14	7
	3	7	2	2	11	2	1	1	2	15	12
LK 3	Norm	AB	BT	DM	GD	J	KD	MD	PS	SM	SMS
	1	3	9	4	4	6	5	1	9	9	10
	2	2	5	9	12	4	2	1	15	6	10
	3	5	5	6	9	11	5	1	13	13	7
PR1	Norm	AB	DM	GD	J	JB	KK	MD	PS	SM	SMS
	1	4	4	6	3	10	10	3	1	3	6
	2	1	3	4	2	6	7	6	8	7	10
	3	1	9	3	10	5	1	1	13	4	5
PR2	Norm	BT	CC	DM	GD	J	JB	KD	PS	SM	SMS
	1	12	3	13	8	3	10	2	1	3	11
	2	3	6	8	5	2	5	3	14	6	8
	3	5	1	6	3	11	3	1	14	10	9
PR3	Norm	CC	DM	GD	J	JB	KD	MD	PS	SM	SMS
	1	3	14	5	5	7	2	3	1	13	12
	2	6	8	12	2	8	3	8	4	4	11
	3	3	5	12	10	10	4	4	10	5	7

4.1.3 Hasil Uji Coba Simulasi Pengenalan Lagu Untuk Durasi 10 Detik

Pada Tabel 4.4 dapat dilihat perbandingan jenis normalisasi pada saat durasi pengambilan *pitch* 10 detik.

Tabel 4.4 Perbandingan Jenis Normalisasi Saat Durasi Pengambilan *Pitch* Lagu 10 Detik

Durasi 10 Detik											
LK 1	Norm	AB	DM	GD	J	JB	KD	KK	MD	SM	SMS
	1	7	4	2	9	6	13	4	8	1	13
	2	2	12	6	4	6	3	7	1	11	4
	3	4	1	10	6	6	1	1	2	10	15
LK 2	Norm	AB	CC	DM	J	JB	KD	KK	MJ	PS	SM

	1	1	1	5	4	6	1	1	8	8	5
	2	1	2	11	6	4	1	7	15	14	12
	3	6	2	1	7	3	1	2	12	14	10
LK 3	Norm	AB	BT	DM	GD	J	KD	MD	PS	SM	SMS
	1	2	11	3	5	4	1	5	11	9	13
	2	2	5	11	13	3	4	1	15	13	8
	3	4	1	1	9	3	5	1	14	11	15
PR1	Norm	AB	DM	GD	J	JB	KK	MD	PS	SM	SMS
	1	2	4	8	3	8	9	8	2	3	7
	2	1	9	9	5	5	1	1	11	6	10
	3	1	1	9	5	8	1	1	12	10	15
PR2	Norm	BT	CC	DM	GD	J	JB	KD	PS	SM	SMS
	1	11	3	13	8	1	10	2	4	3	12
	2	6	5	9	7	3	9	1	14	9	6
	3	2	1	3	5	5	5	1	12	9	15
PR3	Norm	CC	DM	GD	J	JB	KD	MD	PS	SM	SMS
	1	2	13	8	5	7	1	5	3	13	12
	2	4	8	11	3	10	2	2	8	7	6
	3	3	2	14	4	9	5	3	11	6	15

4.2 Analisis Hasil Simulasi Sistem QbSH

Pada sistem QbSH *query* yang disenandungkan *user* dibandingkan dengan lagu pada *database*. Pada sistem pengenalan ini *user* hanya diperkenankan menyenandungkan *query* pada bagian awal lagu saja. Karena sistem QbSH yang dibuat di-*setting* pada bagian awal lagu. Hal ini berkaitan dengan waktu proses pencocokkan representasi melodi. Jika *query* di-*setting* secara sembarang maka sistem membutuhkan waktu yang lama untuk mencari representasi melodi yang cocok pada *database* lagu. Keberhasilan tingkat pengenalan sistem diukur dari besarnya nilai MRR yang dapat dihitung dengan persamaan (3.3) Semakin tinggi nilai MRR maka tingkat pengenalan sistem semakin akurat. Oleh karena itu, untuk meningkatkan nilai MRR pada sistem QbSH ini dilakukan analisis terhadap uji

coba variasi normalisasi, durasi pengambilan *pitch* lagu, durasi perekaman *query* dan pengaruh jenis kelamin (*gender*) terhadap *query*.

4.2.1 Analisis Pengaruh Normalisasi Terhadap *Query*

Normalisasi dilakukan karena terdapat perbedaan antara nilai *pitch* senandungan *query* dengan lagu pada *database*. Perbedaan ini menyebabkan nilai probabilitas saat pembelajaran dengan pengujian berbeda. Pada saat pembelajaran model $\bar{\lambda}$ yang didapat setelah iterasi menggunakan *observer* (*pitch*) dari lagu. Sehingga setiap lagu memiliki model $\bar{\lambda}$ tersendiri. Pada saat pengujian yang menjadi *observer* adalah *pitch* dari *query*. Nilai *pitch* dari *query* (*x*) ini akan dimasukkan kedalam fungsi PDF $g(x,\mu_j,\sigma^2_j)$. Setelah itu dihitung *log probability* dari setiap lagu. Untuk lagu pertama memiliki *log probability* $P(\mathbf{O}|\bar{\lambda}_1)$, lagu kedua *log* $P(\mathbf{O}|\bar{\lambda}_2)$ sampai dengan lagu kelima belas *log* $P(\mathbf{O}|\bar{\lambda}_{15})$. Besarnya *log probability* dari semua lagu akan diurutkan dari yang terbesar. Lagu yang memiliki *log probability* terbesar merupakan lagu yang paling mirip representasi melodinya dengan senandungan *query*. Dan lagu ini akan ditempatkan di urutan pertama pada *output* hasil pencarian.

Tabel 4.5 Contoh Perbandingan *Pitch* Sebelum dan Sesudah Normalisasi Lagu Adu Buyung yang Disenandungkan Oleh LK2

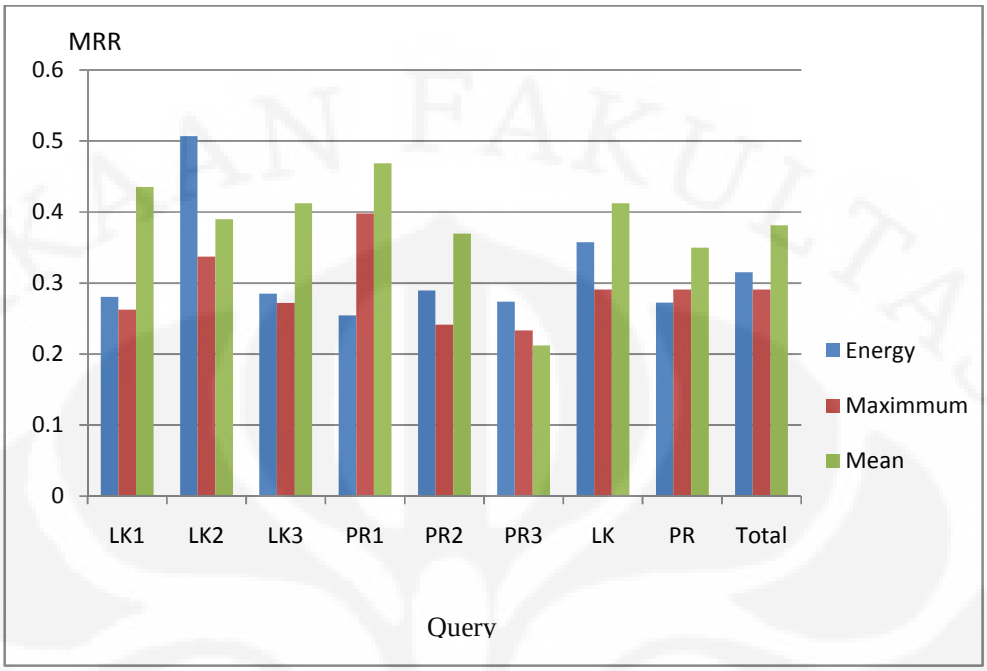
Sebelum Normalisasi	*Pitch Query*	52	51	56	57	56	59	55	57
	Pitch Lagu	74	76	75	78	74	71	74	72
Setelah Normalisasi 3	*Pitch Query*	0,94	0,98	1,02	1,02	0,99	1,03	1,06	1,01
	Pitch Lagu	0,98	1,01	1,00	1,04	0,98	1,01	0,94	0,98

Pada Tabel 4.5 terlihat bahwa besarnya *pitch* antara *query* (Adu Buyung LK2) dengan lagu (Adu Buyung) berbeda jauh. Akan tetapi setelah mengalami normalisasi 3 (energi) kedua nilai *pitch* tersebut hampir mirip. Sehingga *log probability* dari lagu Adu Buyung tersebut memiliki nilai yang besar.

Setelah itu dihitung MRR tiap-tiap orang dari data yang diperoleh pada Tabel 4.4 dengan memvariasikan ketiga jenis normalisasi. Seperti terlihat pada Gambar 4.1 bahwa dari total (seluruh) *query* yang disenandungkan *user* jenis normalisasi 3 (rata-rata) memiliki MRR terbesar (38,12%) dibandingkan jenis normalisasi 1 (energi) dan normalisasi 2 (maksimum) dengan durasi pengambilan *pitch* lagu durasi perekaman *pitch* 10 detik. Sedangkan *query* laki-laki (LK) memiliki MRR lebih tinggi (41,24%) daripada *query* perempuan (PR). Hal ini dikarenakan pada sampel *query* ini laki-laki lebih mengingat melodi lagu. Disamping itu senandungannya memiliki tempo yang pas. Sehingga setelah dilakukan ketiga jenis normalisasi *query* laki-laki (LK) memiliki nilai *pitch* yang mirip dengan lagu. Akan tetapi, pada *query* yang disenandungkan LK2 normalisasi 1 (energi) memiliki MRR lebih tinggi (50,67%) dibandingkan normalisasi 3 (rata-rata) yang memiliki MRR 41,23%. Hal ini terjadi karena setelah normalisasi 1 (energi) senandungan LK2 hanya memiliki 3 buah *query* yang dikenali berada di urutan 5 keatas (lagu). Sedangkan dengan Menggunakan normalisasi 3 (rata-rata) senandungan LK2 memiliki 5 buah *query* yang dikenal berada di urutan 5 keatas (lagu) seperti ditunjukkan pada Tabel 4.4 diatas. Hal ini tentu akan mempengaruhi nilai MRR.

Tabel 4.6 Perbandingan Jenis Normalisasi Tiap *Query*

Norm	MRR								
	LK1	LK2	LK3	PR1	PR2	PR3	LK	PR	Total
Energy	0.2807	0.5067	0.2853	0.2546	0.2893	0.2738	0.3576	0.2726	0.3151
Max	0.2627	0.3372	0.272	0.398	0.2414	0.2334	0.2906	0.2909	0.2908
Mean	0.435	0.3898	0.4123	0.4686	0.3694	0.2123	0.4124	0.3501	0.3812



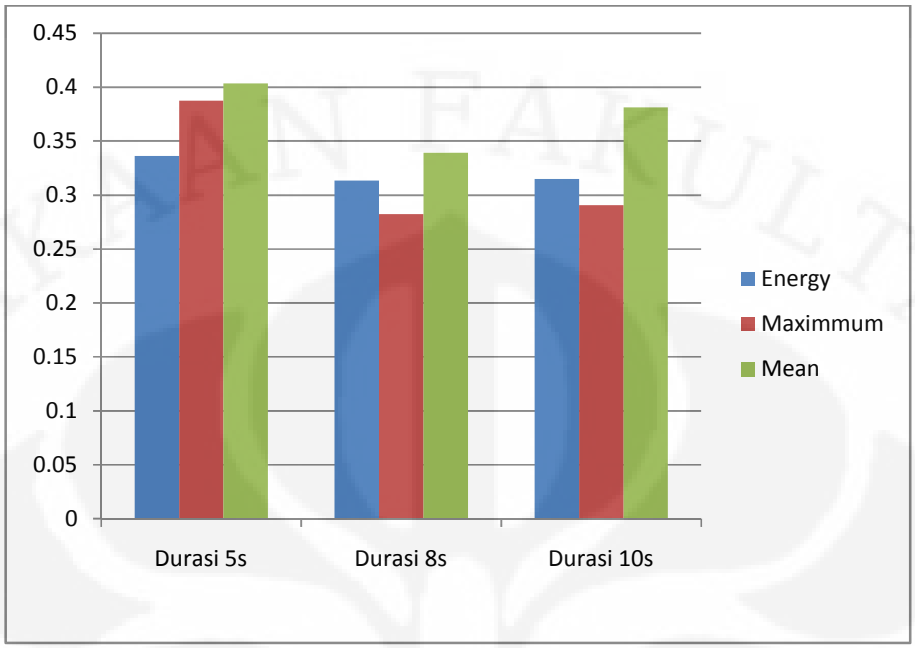
Gambar 4.1 Grafik Perbandingan Jenis Normalisasi Tiap *Query*

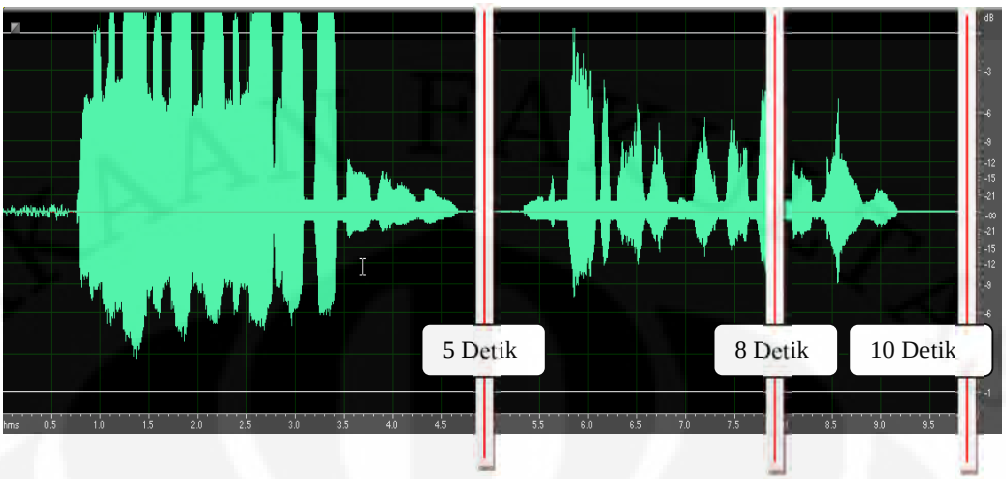
4.2.2 Analisis Pengaruh Durasi Pengambilan *Pitch* Lagu

Banyaknya *pitch* yang terkandung dalam suatu lagu berbeda-beda bergantung pada lamanya durasi lagu tersebut. Semakin lama durasi lagu tersebut maka semakin banyak jumlah *pitch* yang terkandung didalamnya. Pengujian dilakukan dengan menyamakan durasi pengambilan *pitch* selama 5 detik (5s), 8 detik (8s) dan 10 detik (10s). Disamping itu pengujian juga dilakukan dengan memvariasikan jenis normalisasinya. Hasil pengujian diukur dalam besarnya nilai MRR seperti ditunjukkan pada Tabel 4.7 berikut.

Tabel 4.7 Perbandingan Durasi Pengambilan *Pitch* Lagu

Normalisasi	MRR		
	Durasi 5s	Durasi 8s	Durasi 10s
Energy	0.3363	0.3135	0.3151
Maximmum	0.3875	0.2823	0.2908
Mean	0.4035	0.3394	0.3812


0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Durasi 5s
Durasi 8s
Durasi 10s
Energy
Maximmum
Mean



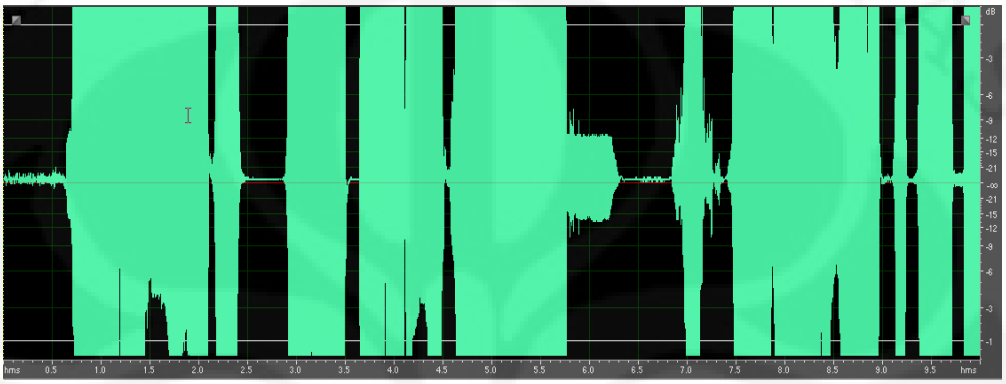
Gambar 4.3 Contoh *Query* PR2 (Kopi Dangdut)

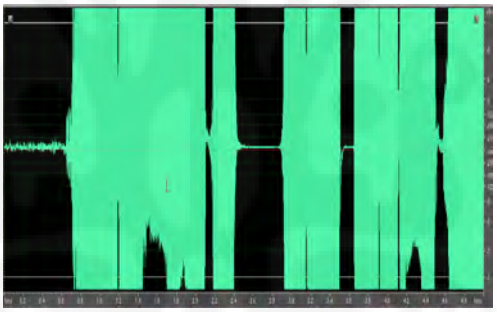
Seperti ditunjukkan pada Tabel 4.4 bahwa *query* kopi dangdut PR2 berada di urutan pertama. Oleh karena itu, durasi pengambilan *pitch* 5 detik dari lagu lebih merepresentasikan *query* yang disenandungkan *user*.

Saat durasi pengambilan *pitch* lagu 8 detik terdapat perubahan nada. Sehingga menyebabkan pembagi ( ) *pitch* normalisasi tidak merepresentasikan populasi data *pitch* lagu. Hal ini mengakibatkan nilai MRR menjadi turun. Sedangkan pada saat pengambilan *pitch* lagu 10 detik terjadi perubahan nada yang lebih tinggi. Sehingga menyebabkan pembagi ( ) *pitch* normalisasi merepresentasikan populasi data *pitch* lagu. Dan mengakibatkan MRR menjadi tinggi kembali (38,12%) dibandingkan saat durasi pengambilan 8 detik (MRR 33,94%). Seperti ditunjukkan pada Gambar 4.2 bahwa dengan menggunakan jenis normalisasi 3 (rata-rata) dan durasi pengambilan *pitch* lagu 5 detik dapat menghasilkan nilai MRR yang optimum sebesar 40,35 %.

4.2.3 Analisis Pengaruh Durasi Pengambilan *Pitch Query*

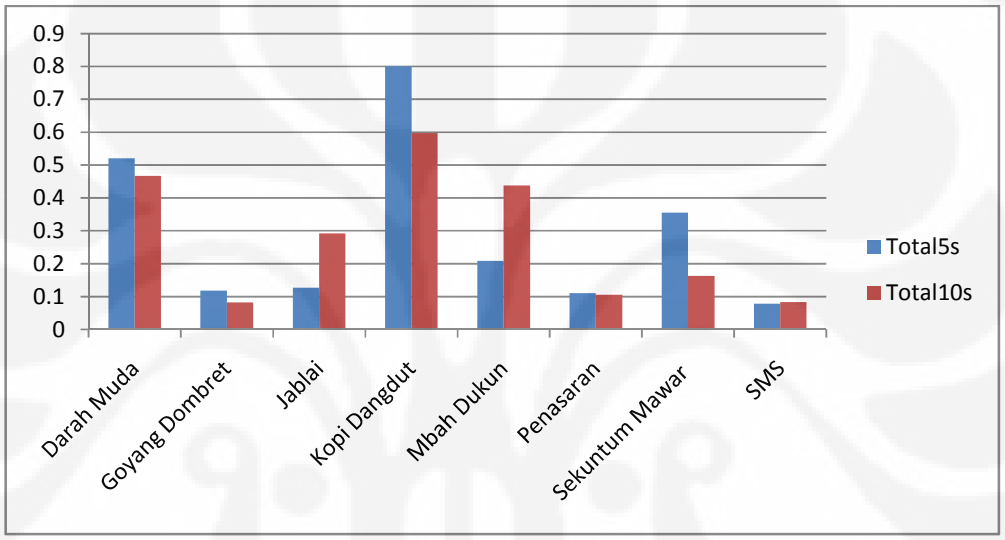
Prinsip sistem QbSH adalah mencari lagu menggunakan senandungan melodi dalam beberapa detik saja. Pengujian dilakukan terhadap durasi pengambilan *pitch* dari senandungan *query*. Berapa lama waktu efektif yang dibutuhkan untuk melakukan senandungan agar hasilnya menempati urutan pertama. Pengujian dilakukan dengan menggunakan lagu yang disenandungkan oleh 4 orang (2 orang laki-laki dan 2 orang perempuan). Jadi satu lagu





Gender	MRR							
	DM	GD	J	KD	MD	PS	SM	SMS
LK_{10}	0.1833	0.0889	0.25	0.5714	0.375	0.1	0.1181	0.0871
PR_{10}	0.75	0.075	0.3333	0.625	0.5	0.1111	0.2083	0.0788
$Total_{10}$	0.4666	0.0819	0.2916	0.5982	0.4375	0.1055	0.1632	0.0829

Gender	MRR							
	DM	GD	J	KD	MD	PS	SM	SMS
LK_5	0.625	0.0788	0.125	1	0.25	0.101	0.127	0.0788
PR_5	0.4167	0.1583	0.127	0.6	0.1667	0.1181	0.5833	0.0774
$Total_5$	0.52085	0.11855	0.126	0.8	0.20835	0.10955	0.35515	0.0781



pada Gambar 4.6 diatas dapat disimpulkan bahwa durasi lama senandungan yang efektif adalah selama 10 detik.

Pada lagu-lagu yang populer misalnya kopi dangdut dan sekuntum mawar merah cukup dengan melakukan *query* selama 5 detik sudah merepresentasikan melodi lagu tersebut. Ketika lagu itu sangat populer maka orang akan selalu ingat melodi lagu tersebut. Sehingga melodi yang disenandungkannya memiliki tempo yang pas. Akan tetapi sebaliknya, ketika lagu tersebut kurang popular atau terlalu sulit melodinya mengakibatkan orang terlalu lama berpikir untuk menyenandungkan melodi yang tepat. Sehingga timbul banyaknya kondisi *silent.* Dan seperti yang telah dijelaskan sebelumnya bahwa kondisi ini mempengaruhi banyaknya *pitch* yang terkandung dalam *query* tersebut.

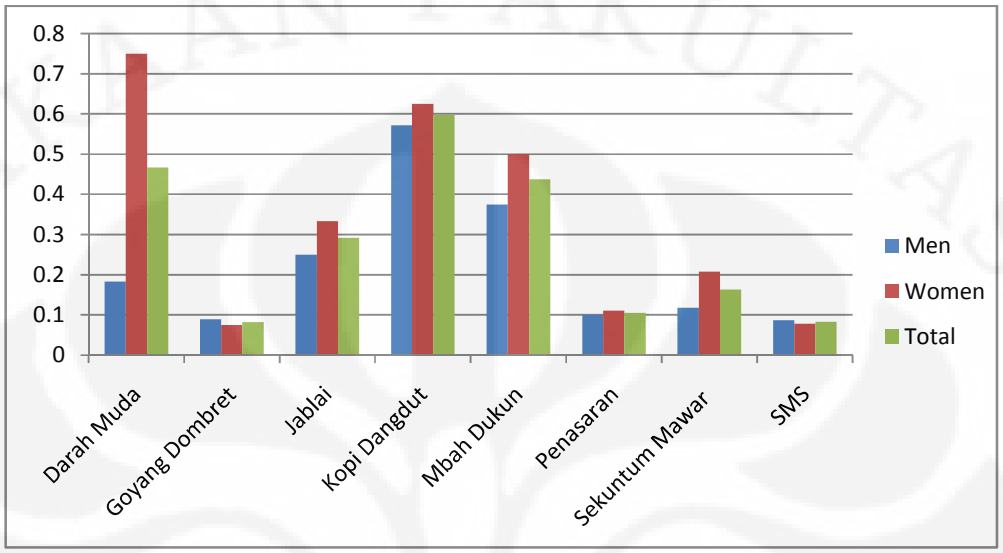
4.2.4 Analisis Pengaruh *Gender*

Seperti yang telah dijelaskan sebelumnya bahwa jenis normalisasi dan durasi pengambilan *pitch* lagu yang paling optimum (MRR 40,35%) adalah menggunakan normalisasi 3 (rata-rata) dan durasi 5 detik. Pengujian dilakukan terhadap jenis kelamin penyenandung *query*. Seperti ditunjukkan pada Gambar 4.7 terlihat bahwa *query* yang disenandungkan perempuan (PR) memiliki MRR yang selalu lebih tinggi dibandingkan *query* laki-laki (LK). Hal ini disebabkan *query* yang disenandungkan perempuan memiliki *pitch* yang relatif lebih tinggi dibandingkan laki-laki. Disamping itu senandungan memiliki irama yang lebih teratur seperti ditunjukkan pada tabel 4.10 berikut.

Tabel 4.10 Contoh Perbandingan *Pitch* Laki-Laki dan Perempuan

Lagu	Darah Muda			Jablai			Sekuntum Mawar		
Pitch	MIDI	LK	PR	MIDI	LK	PR	MIDI	LK	PR
	71	54	60	69	53	61	82	53	60
	69	57	60	68	55	63	84	56	60
	72	56	60	69	58	61	84	57	60
	71	58	59	71	55	64	85	57	63
	69	58	61	71	55	64	84	56	58

	72	57	57	69	57	61	84	56	61



BAB 5
KESIMPULAN

Berdasarkan hasil uji coba dan analisis terhadap piranti perangkat lunak simulasi sistem QbSH, maka dapat diambil kesimpulan antara lain sebagai berikut :

1. Besarnya nilai MRR pada sistem QbSH ditentukan oleh parameter representasi melodinya.
2. Jenis normalisasi menggunakan rata-rata menghasilkan nilai MRR maksimal dibandingkan normalisasi menggunakan energi atau maksimumnya.
3. Durasi pengambilan *pitch* lagu selama 5 detik menghasilkan nilai MRR optimum dibandingkan 8 detik ataupun 10 detik.
4. Dengan menggunakan variasi normalisasi rata-rata dan durasi pengambilan *pitch* lagu selama 5 detik didapatkan MRR sebesar 0,40.
5. Durasi pengambilan *pitch* senandungan yang optimum dilakukan selama 10 detik.
6. Jenis kelamin mempengaruhi nilai MRR. Pada simulasi sistem QbSH ini sampel perempuan memiliki nilai MRR lebih tinggi dibandingkan sampel laki-laki.

DAFTAR REFERENSI

[1] Dannenberg, R.B. (2007). *Comparative Evaluation of Search Techniques for Query-by-Humming Using MUSART*. *Journal of the American Society for Information Science and Technology*, vol. 58, no. 2, pp.687–701 Diakses November 17, 2009 dari http://ismir2003.ismir.net/papers/Dannenberg.PDF

[2] Lie Lu, Hong You and Hong-Jiang Zhang. (2001). *A New Approach To Query By Humming In Music Retrieval*. Microsoft Research, China.

[3] Jang, Roger., & Hsu, Chao-ling (2005). *Continuous HMM and Its Enhancement for Singing/Humming Query Retrieval. Proceeding* Ismir2005. Diakses November 17, 2009 dari http:// ismir2005.ismir/netproceedings1064-chmm.pdf

[4] Sijabat, David. (2009). Simulasi Pengenalan *Chord* Terisolasi Berbasiskan *Speaker Dependent* Dengan Menggunakan *Hidden Markov Model.* Skripsi, Universitas Indonesia, Depok.

[5] Muller, Meinard. (2007). *Information Retrieval for Music and Motion.* Berlin: Springer.

[6] Jyh-Shing Roger Jang. (2003). *Audio signal processing and recognition.* Online course diakses November 17, 2009 dari http://neural.cs.nthu.edu.tw/jang/courses/isa5571/

[7] Schutte, Ken.*MATLAB and MIDI,* available from the link at author's homepage at" http://www.kenschutte.com/midi".

[8] Klapuri, Anssi. (2006). *Signal processing method for music transcription*. New York: Springer.

[9] Talkin, David. (1995). *A Robust Algorithm for Pitch Tracking.* Entropic Research Laboratory, USA.

[10] Ferrer, Miguel. A. *gpdsHMM: A hidden Markov model toolbox in the Matlab environment*. Available at http://www.gpds.ulpgc.es/download/index.htm

[11] Rabiner, Lawrence. (1985). *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Proceeding of the IEEE, VOL. 77, NO. 2. February 1989. Diakses November 25, 2009 dari http://www.cs.ubc.ca/~murphyk/Bayes/rabiner.pdf

[12] Jyh-Shing Roger Jang, "Melody Recognition Toolbox", available from the link at the author's homepage at "http://mirlab.org/jang".

[13] Jyh-Shing Roger Jang, *Speech and Audio Processing Toolbox*, available from the link at the author's homepage at "http://mirlab.org/jang".

[14] Jyh-Shing Roger Jang, "Utility Toolbox", available from the link at the author's homepage at "http://mirlab.org/jang".

LAMPIRAN

Script simulasi yang dimodifikasi dari toolbox

Training Database

```
function varargout = training_data(varargin)
% TRAINING_DATA M-file for training_data.fig
%      TRAINING_DATA, by itself, creates a new TRAINING_DATA or raises the
existing
%      singleton*.
%
%      H = TRAINING_DATA returns the handle to a new TRAINING_DATA or the
handle to
%      the existing singleton*.
%
%      TRAINING_DATA('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in TRAINING_DATA.M with the given input
arguments.
%
%      TRAINING_DATA('Property','Value',...) creates a new TRAINING_DATA or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before training_data_OpeningFunction gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to training_data_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 training_data

% Last Modified by GUIDE v2.5 02-May-2010 14:49:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @training_data_OpeningFcn, ...
                   'gui_OutputFcn',  @training_data_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 training_data is made visible.
function training_data_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 training_data (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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

 % --- Outputs from this function are returned to the command line.
function varargout = training_data_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 Data_Callback(hObject, eventdata, handles)
% hObject    handle to Data (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 Data as text
%        str2double(get(hObject,'String')) returns contents of Data as a
double


% --- Executes during object creation, after setting all properties.
function Data_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Data (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 fileHmm_Callback(hObject, eventdata, handles)
% hObject    handle to fileHmm (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 fileHmm as text
%        str2double(get(hObject,'String')) returns contents of fileHmm as a
double


% --- Executes during object creation, after setting all properties.
function fileHmm_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fileHmm (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 pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function pushbutton1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



% --- Executes on button press in proses.
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to proses (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Qbh system (training PART)
% modified from Grupo De Procesado Digital De Senales(GPDS) Universidad de
Las Palmas de Gran Canaria
% HMM toolbox CHMM
% using Roger Jang SAP toolbox
% by Ewaldo Zihan

tic
% load the song data
```

```
data=get(handles.Data,'string');
load(data);
% number of classes
nc=size(songDb,2); %menghitung jumlah lagu angka 2 menunjukkan mengambil dari
kolom
%nc=1;
% number of repetitions
nr=10;
% number of groups
ng=2;

% Matriks for the parameters
vlcp=cell(nc,ng);
for ic=1:nc
   vlcp{ic,1}=cell(nr,1);
   vlcp{ic,2}=cell(nr,1);
   for ig=1:ng
      vlcp{ic,ig}=cell(nr,1);
   end
end
% generation nr repetitions
for ic=1:nc
    for ir=1:nr
    pv=songDb(ic).pv;
    % pv=pv(1:100);
    % change pv to note
    pv_tanpa_nol=restHandle(pv,0);
    timeUnit=256/8000;
    pitchTh=0.8;
    micNoteDuration=0.1;
    plotOpt=0;
    note = pv2note(pv_tanpa_nol, timeUnit, pitchTh, micNoteDuration,
                   plotOpt);
    durasi=note(2:2:end);
    durasi5s=detik(durasi);
    pitch=note(1:2:end);
    pitch_dur=pitch(1:durasi5s);
    pitch_nor=normalize3(pitch_dur);
    vlcp{ic,1}{ir}=[pitch_nor(1:end)'];

    %ambil pitch lagi
    pv=songDb(ic).pv;
    % pv=pv(1:100);
    % change pv to note
    pv_tanpa_nol=restHandle(pv,0);
    timeUnit=256/8000;
    pitchTh=0.8;
    micNoteDuration=0.1;
    plotOpt=0;
    note = pv2note(pv_tanpa_nol, timeUnit, pitchTh, micNoteDuration,
                    plotOpt);
    durasi=note(2:2:end);
    durasi5s=detik(durasi);
    pitch=note(1:2:end);
    pitch_dur=pitch(1:durasi5s);
    pitch_nor=normalize3(pitch_dur);
```

```
    vlcp{ic,2}{ir}=[pitch_nor(1:end)'];

    end
end

% get name for the HMM data
file_hmm=get(handles.fileHmm,'string');
vtrain=vlcp;
vl=vtrain;

save vtrain vl
clear vtrain vl vlcp;

chmm_def(file_hmm);
train(file_hmm,'vtrain');
selesai='Proses Selesai';
set(handles.selesai,'string', selesai);
% print the procces time
durasi=sprintf(' ===> %.2f sec\n', toc);
set(handles.durasi,'string', durasi);
```

DURASI

```
function durasi5s=detik(durasi)
lama=6;
[nb,nc]=size(durasi);
durasi5s=1;
jumlah=0;
while jumlah<lama && durasi5s<=nc,
    jumlah=jumlah+durasi(1,durasi5s);
    durasi5s=durasi5s+1;
end
durasi5s=durasi5s-1;
```

NORMALIZE 1 (Energi)

```
function normalize1=normalize1(pitch_dur)

norm(pitch_dur);
energy=norm(pitch_dur)/length(pitch_dur);
normalize1=pitch_dur/energy;
```

NORMALIZE 2 (Maksimum)

```
function normalize2=normalize2(pitch_dur)

pitchmax=max(pitch_dur);
normalize2=pitch_dur/pitchmax;
```

NORMALIZE 3 (Rata-rata)

```
function normalize3=normalize3(pitch_dur)

pitchmean=mean(pitch_dur);
normalize3=pitch_dur/pitchmean;
```

CHMM_DEF

```
%--------------------------------------------------------------------------
%
%                           %%%%%%%%%%
%                           %CHMM_DEF %
%                           %%%%%%%%%%
%
%
% This function is described only as example to set up the HMM.
%
% function chmm_def(fhmm)
%
% This function defines the parameters of the discrete HMM. We can split the
parameters in 4 groups:
%
% Entry:    fhmm is the HMM´s file name.
%
%   vDB defines the database:
% vDB=[' nc ng agrup Np'];
% where nc is the number of class, ng is the number of group,
% agrup defines how to gather the parameters together
% NP is a vector with the number of each group of parameter.
% We have the following relations:
% [nc,ng]=size(vl);
% agrup{ig}=[1 .... size(vl{1,ig}{1},2)+1];
% Np(ig)=length(agrup{ig})-1;
%
%   vHMM defines the HMM :
%   vHMM =[' BAKIS salto maxiter umbral maxitermi Ngauss Ne A B Med Var Pi'];
% if Bakis = 1 it implements a Bakis HMM (also called left-right), else an
ergodic HMM is defined
% Salto is the maximum path authorized in the Bakis HMM from a state to
another.
% Maxiter is the maximum iteration number in the Bakis HMM (condition to stop
the algorithm)
% Umbral is the threshold condition to stop the HMM (the error is calculated
with the maximum likelihood criterion)
% Maxitermi is the iteration number to find the initial model.
% Ne states number
% Ngauss is the number of Gaussians in the mixture of Gaussians used to
represent the distribution of the observation by state.
% A, B and Pi are the matrices of the HMM (it will be calculated thanks to
the CHMM) but must de defined here.
% Med and Var are the matrices of the mean and variances of the Gaussians.
%
%  vTEST is the matrix parameter for the test :
% vTEST=[' salhmm'];
```

```
% Salhmm is a matrix used to store the probabilities values
% We have the following relations:
% salhmm=cell(nc,ng);
%
%
%
% ------------------------------------------------------------------------


function chmm_def(fhmm)

% name of the file of the CHMM
%fhmm='hmm';

iniciar=1;

% VARIABLES of the Database
vDB=[' nc ng agrup Np'];
% classes number and groups number with
% the relation [nc,ng]=size(vl);

% number of classes
load vtrain vl
% number of classes
nc=size(vl,1); % 1 menunjukkan ambil sebanyak n baris
ng=size(vl,2);  % 2menunjukkan ambil sebanyak m kolom

% Definition and numbers of the inputs.
% each realisation is a group of vectors and every vector contains
% a sequence of components that we gather in oreder to create the paramenters
inputs.
% We define the gathering of the components and the number of groups.
% we have the relation agrup{ig}=[1 .... size(vl{1,ig}{1},2)+1];
grup=cell(ng,1);
Np=zeros(ng,1);
for ig=1:ng
   agrup{ig}=[1 2];
end


%agrup{2}=[1 2 3];
for ig=1:ng
   Np(ig)=length(agrup{ig})-1;
end


% VARIABLES of the HMM
vHMM=[' BAKIS salto maxiter umbral maxitermi Ngauss Ne A B Med Var Pi men'];
% Number of gaussians by groups and parameter.
% All the states have the same number of gaussians
Ngauss=cell(ng,1);
for ig=1:ng
   Ngauss{ig}=1.*ones(Np(ig),1);% We can change it and fix a number different
for each prameter of each group For instance: Ngauss{1}=[1 ; 3];
end;
```

```
men=1;
% Variables for the HMM
% States number.
%Ne=6.*ones(nc,ng);% we could change the number of state for each class and
groupt. Ne=[3 4; 5 6; 7 8; 9 10]
% number of state depend of number of notes or durations
Ne=zeros(nc,2);
for ic=1:nc,
    s=size(vl{ic,1}{1,1});
    s=s(:,1);
    Ne(ic,1)=s;
    a=size(vl{ic,2}{1,1});
    a=a(:,1);
    Ne(ic,2)=a;
end
% if Bakis = 1 it implements a Bakis HMM (also called left-right), else an
ergodic HMM is defined.
BAKIS=1;
% Salto is the maximum path authorized in the Bakis HMM from a state to
another.
salto=1;
% Maximum number of iterations in the Bakis HMM (condition to stop the
algorithm).
maxiter=10;
% the threshold condition to stop the HMM (the error is calculated with the
maximum likelihood criterion)
umbral=0.05;
% the iteration number to find the initial model.
maxitermi=10;
% Matrices of the HMM
A=cell(nc,ng);
B=cell(nc,ng);
Med=cell(nc,ng);
Var=cell(nc,ng);
Pi=cell(nc,ng);
for ic=1:nc,
   for ig=1:ng
      A{ic,ig}=zeros(Ne(ic,ig),Ne(ic,ig));
      B{ic,ig}=cell(Np(ig),1);
      Med{ic,ig}=cell(Np(ig),1);
      Var{ic,ig}=cell(Np(ig),1);
      for ip=1:Np(ig)
         B{ic,ig}{ip}=cell(Ne(ic,ig),1);
         Med{ic,ig}{ip}=cell(Ne(ic,ig),1);
         Var{ic,ig}{ip}=cell(Ne(ic,ig),1);
         for ie=1:Ne(ic,ig),
            B{ic,ig}{ip}{ie}=zeros(Ngauss{ig}(ip),1);
            Med{ic,ig}{ip}{ie}=zeros(Ngauss{ig}(ip),agrup{ig}(ip+1)-
agrup{ig}(ip));
            Var{ic,ig}{ip}{ie}=zeros(Ngauss{ig}(ip),agrup{ig}(ip+1)-
agrup{ig}(ip));
         end
      end
      Pi{ic,ig}=zeros(Ne(ic,ig),1);
   end
end
```

```
% VARIABLES for the TEST
vTEST=[' salhmm'];

% outputs vectors for each sequence of the test/model
salhmm=cell(nc,ng);

% we save the HMM
guardar=['save ',fhmm,vDB,vHMM,vTEST,' iniciar vDB vHMM vTEST guardar'];
eval([guardar]);
return
```

QBH_SYS

```
function varargout = QBH_Sys(varargin)
% QBH_SYS M-file for QBH_Sys.fig
%      QBH_SYS, by itself, creates a new QBH_SYS or raises the existing
%      singleton*.
%
%      H = QBH_SYS returns the handle to a new QBH_SYS or the handle to
%      the existing singleton*.
%
%      QBH_SYS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in QBH_SYS.M with the given input arguments.
%
%      QBH_SYS('Property','Value',...) creates a new QBH_SYS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before QBH_Sys_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to QBH_Sys_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 QBH_Sys

% Last Modified by GUIDE v2.5 02-May-2010 15:00:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @QBH_Sys_OpeningFcn, ...
                   'gui_OutputFcn',  @QBH_Sys_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 QBH_Sys is made visible.
function QBH_Sys_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 QBH_Sys (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = QBH_Sys_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 edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (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 proses.
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to proses (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Qbh system (TEST PART)
% modified from Grupo De Procesado Digital De Senales(GPDS) HMM toolbox
Universidad de Las Palmas de Gran Canaria
% using Roger Jang SAP toolbox
% by Ewaldo Zihan

tic
% load the HMM file
file_hmm=get(handles.fileHmm,'string');

load vtrain vl

% number of classes
nc=size(vl,1);

% number of groups
ng=2;
vtest=cell(nc,ng);

%parameter live recording
duration=10;
fs=8000;
nbits=8;

% ==== == Input format

liveRecording=get(handles.liveRec,'Value');;
if liveRecording
        fprintf('Prepared to recording... %d-second ', duration);
        pause (1);fprintf('\n 3');
        pause (1);fprintf('\n 2');
        pause (1);fprintf('\n 1');pause (1);
        fprintf('\nstart %d-second recording... ', duration);
        y=wavrecord(fs*duration, fs, 'uint8');
        y=double(y);
        y=y-mean(y);
        waveFile=[tempname, '_qbsh.wav'];
        wavwrite(y/128, fs, nbits, waveFile);
        fprintf('Finish recording (%s).\n', waveFile);
  else
    [query,path]=uigetfile('.WAV','query');
```

```
    waveFile=query;
end
    [y, fs, nbits]=wavread(waveFile);
    PP=ptParamSet(fs, nbits);
    plotOpt=0;
    [pitch, clarity]=wave2pitchByAcf(y, PP, plotOpt);
    pvTest=pitch;
 % change pv to note
    pvTest_tanpa_nol=restHandle(pvTest,0);
    timeUnit=256/8000;
    pitchTh=0.8;
    minNoteDuration=0.1;
    plotOpt=0;
    note = pv2note(pvTest_tanpa_nol, timeUnit, pitchTh, minNoteDuration,
plotOpt);
 % get pitch from note
    pitchTest=note(1:2:end)
    pitchTest=normalize3(pitchTest);


 % convert pitch to contour
    %contourTest=pitch2contourm(pitchTest);
    %[n,m]=size(pitchTest);
    vtest{1,1}{1}=[pitchTest'];
  %get duration from note
    %durasiTest=note(2:2:end);
    %[n,m]=size(durasiTest);

     pvTest_tanpa_nol=restHandle(pvTest,0);
    timeUnit=256/8000;
    pitchTh=0.8;
    minNoteDuration=0.1;
    plotOpt=0;
    note = pv2note(pvTest_tanpa_nol, timeUnit, pitchTh, minNoteDuration,
plotOpt);
 % get pitch from note
    pitchTest=note(1:2:end);
    pitchTest=normalize3(pitchTest);
    vtest{1,2}{1}=[pitchTest'];
    %vtest{1,2}{1}=[durasiTest'];
    vl=vtest;

   save vtest vl
   test(file_hmm,'vtest');

   logPO=probabilitas(file_hmm);
   data=get(handles.Data,'string');
   load(data);

   fprintf('\n');
   %number of output displayed
   outputSongNum=15;
   [junk, id4sort]=sort(logPO,'descend');
   peringkat='';
    for i=1:outputSongNum
        index=id4sort(i);
```

```
        songName=songDb(index).songName;
        %items=split(songName, '_');
        %if length(items)>1, songName=items{1}; end
        peringkat=sprintf('%s \n %02d. %s',peringkat, i, songName);
        rank=fprintf('%02d. (%.1f)  %s \n', i, logPO(index), songName);
        set(handles.rank,'string', peringkat);
    end
    fprintf('\n');
% print the procces time
durasi=sprintf(' ===> %.2f sec\n', toc);
set(handles.durasi,'string', durasi);


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


% --- Executes on button press in liveRec.
function liveRec_Callback(hObject, eventdata, handles)
% hObject    handle to liveRec (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 liveRec




function Data_Callback(hObject, eventdata, handles)
% hObject    handle to Data (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 Data as text
%        str2double(get(hObject,'String')) returns contents of Data as a
double


% --- Executes during object creation, after setting all properties.
function Data_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Data (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
```