UNIVERSITAS INDONESIA

PERANCANGAN SISTEM QUERY BY SINGING/HUMMING (QbSH) UNTUK MUSIK DANGDUT DENGAN PITCH DAN DURASI SEBAGAI FEATURE

SKRIPSI

Diajukan sebagai salah satu syarat memperoleh gelar sarjana

EWALDO ZIHAN

06 06 07 3871

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

TEKNIK ELEKTRO

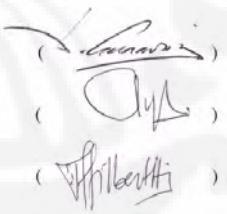
DEPOK

JUNI 2010

HALAMAN PERNYATAAN ORISINALITAS

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

Nama : Ewaldo Zihan
NPM : 0606073871
Tanda Tangan :
Tanggal : 15 Juni 2010



UCAPAN TERIMA KASIH

Puji dan syukur saya panjatkan kepada Allah SWT karena atas rahmat dan hidayah-Nya saya dapat menyelesaikan skripsi ini. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu saya ingin mengucapkan terima kasih kepada :

1. Allah SWT yang telah memberikan kekuatan kepada Saya untuk menyelesaikan buku skripsi ini;
2. Bapak Prof. Dr. Ir. Dadang Gunawan , M.Eng. selaku pembimbing skripsi saya;
3. Bapak Indrabayu Amirullah, ST, MT, M.Bus, Sys. telah banyak memberikan waktu serta ilmunya selama proses pembuatan skripsi ini ;
4. Teddy Febrianto dan Abdul Aziz Muslim yang membantu proses pembelajaran serta pembuatan seminar ini;
5. Orang tua dan keluarga saya yang telah memberikan dukungan moral dan material;
6 Indra Sragen, Indra 2007, Anti, Bani dan Cindy yang menyumbangkan suaranya dengan ikhlas.
7. Seluruh keluarga besar Civitas Akademika Fakultas Teknik Universitas Indonesia khususnya karyawan Departemen Teknik Elektro yang telah banyak memberikan bantuan dalam urusan administrasi skripsi.

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

Depok, 15 Juni 2010

Ewaldo Zihan

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Ewaldo Zihan
NPM : 0606073871
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 Nonoksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul :

PERANCANGAN SISTEM QUERY BY SINGING/HUMMING (QbSH) UNTUK MUSIK DANGDUT DENGAN PITCH DAN DURASI SEBAGAI FEATURE

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta sebagai pemegang Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 2 Juni 2010
Yang menyatakan

Ewaldo Zihan

ABSTRAK

Nama : Ewaldo Zihan

Program Studi : Teknik Elektro

Judul : Perancangan Sistem Query by Singing/Humming (QbSH) untuk musik dangdut dengan pitch dan durasi sebagai feature

Keterbatasan metode pencarian yang bersifat *text-based* pada sistem pencari lagu konvensional telah mendorong lahirnya sistem pencari lagu baru yang berbasis konten lagu. Salah satunya adalah sistem *Query by Singing/Humming* (QbSH). Sistem QbSH adalah sistem yang menggunakan melodi dari lagu sebagai dasar pencarian. Mayoritas penelitian terhadap sistem QbSH hanya menggunakan *pitch* sebagai *feature* padahal durasi juga merupakan karakteristik dari melodi oleh karena itu skripsi ini memaparkan perancangan sistem QbSH untuk musik dangdut yang menggunakan pitch dan durasi sebagai *feature*. *Pitch* direpresentasikan dalam 3, 5, 7, 9, 11 level *contour* dan *pitch interval* untuk mengatasi transposisi frekuensi sedangkan durasi akan direpresentasikan dalam logDurasi, RDD (*Relative Duration Difference*) logRDD kemudian akan dilihat kombinasi mana yang paling baik dalam merepresentasikan musik dangdut yang memiliki karakteristik perubahan pitch yang signifikan. Untuk pencocokan melodi digunakan *Continous hidden markov model* (CHMM). CHMM dipilih karena performa HMM yang baik dalam *speech recognition,* selain itu dengan CHMM *user* tidak perlu bersenandung dengan “da” atau ” la”.

Kata kunci : QbSH, dangdut, representasi *pitch*, representasi durasi, CHMM

ABSTRACT

Name : Ewaldo Zihan

Majority : Electrical Engineering

Title : Query by Singing / Humming (QbSH) system design for dangdut music with pitch and duration as features

The limitation of text-based searching method in the conventional track searching system encourages the invention of the new searching system that used content-based searching method one which is the query by singing/humming (QbSH) system. QbSH system is a system that uses the melody of the song as the basis for the search. The majority of research on QbSH system only uses pitch as a feature but pitch isn't the only feature that characterized the melody therefore this final present a QbSH system design for dangdut music using pitch and duration as a feature. Pitch is represented in three, five, seven, nine, 11-level contour and pitch interval to overcome the transposition frequency while the duration will be represented in logdurasi, RDD (Duration Relative Difference) logRDD then will see where the best combination to represent dangdut music that has characteristics significant changes in pitch. To match the melody used in continuous hidden Markov model (CHMM). CHMM HMM was selected because of good performance in speech recognition, in addition to the CHMM user does not need to hum with "da" or "la".

Key words : QbSH, dangdut, pitch representation, duration representaion, CHMM

DAFTAR ISI

Halaman

HALAMAN JUDUL ……i

HALAMAN PERNYATAAN ORISINALITAS ……...ii

LEMBAR PERSETUJUAN……iii

UCAPAN TERIMA KASIH……iv

LEMBAR PERSETUJUAN PUBLIKASI KARYA ILMIAH ……v

ABSTRAK ……..vi

ABTRACT ……..vii

DAFTAR ISI……viii

DAFTAR GAMBAR ……xi

DAFTAR TABEL ……..……xii

DAFTAR SINGKATAN xiii

BAB 1 PENDAHULUAN ……1

1.1 Latar Belakang ……1

1.2 Perumusan Masalah ……2

1.3 Tujuan.....…….……2

1.4 Pembatasan Masalah .....……...3

1.5 Sistematika Penulisan……...3

BAB 2 MELODI, REPRESENTASI MUSIK, DAN QbSH.....…….……5

2.1 Melodi…….....5

2.1.1 Karakteristik nada.....…….....5

2.1.2 Persepsi Melodi.....……6

2.1.3 *Melodic Similarity*……...……7

2.2 Representasi Musik...……...7

2.2.1 Audio Format/Waveform format .....…….……8

2.2.2 MIDI…...….....……..…….……...…...10

2.3 *Query by Humming* (QbH)...…….…….10

2.3.1 *Database*.....……...11

2.3.2 Pemerosesan *Query*..……12

2.3.2.1 *Pitch Tracking*……12

2.3.2.1 *Note Segmentation*……..12

2.3.3 Proses Pencocokan Melodi…….....…...13

2.3.4 Menampilkan Hasil.……..13

BAB 3 PERANCANGAN SIMULASI SISTEM QbSH

DENGAN METODE PITCH TRACKING AMDF.....…….....……..14

3.1 Perancangan Sistem QbSH ……....……14

3.2 *Database*.....…….15

3.3 Pemerosesan *Query*……..…….....15

3.3.1 *Average Magnitude Difference Function* (AMDF).....16

3.3.2 *Note Segmentation*……..……...17

3.4 Perepresentasian Melodi.....18

3.5 *Continuous Hidden Markov Models* (CHMM).....19

3.6 Parameter Performa 22

3.7 Simulasi.....23

BAB IV HASIL SIMULASI DAN ANALISIS…..…….....24

4.1 Hasil Simulasi..... ……...……24

4.1.1 Hasil Uji 3 level *Contour*…….26

4.1.2 Hasil Uji 5 level *Contour*.…….26

4.1.3 Hasil Uji 7 level *Contour*……..27

4.1.4 Hasil Uji 9 level *Contour*……..27

4.1.5 Hasil Uji 11 level *Contour*……...28

4.1.6 Hasil Uji *Pitch Interval*.……..28

4.1.7 Hasil Uji Variasi *Framing* dan *Overlap* …….29

4.1.8 Hasil Uji Waktu *Training* Terhadap Representasi Melodi……...29

4.2 Analisa……...…….30

4.2.1 Analisa Pengaruh Representasi Melodi Terhadap MRR……..30

4.2.2 Analisa Pengaruh Framing dan Overlap Terhadap MRR…….34

4.2.3 Analisa Performa……..34

BAB V KESIMPULAN…….…….…….....36

DAFTAR REFERENSI..……..……..37

DAFTAR GAMBAR

Halaman

Gambar 2.1 *Score* dari *Beethoven's Fifth* ……..8

Gambar 2.2 *Waveform* dengan frekuensi 4Hz.…...…….……..8

Gambar 2.3 Detik ke 7.3 sampai 7.8 dalam *Beethoven's fifth Bernstein*.....9

Gambar 2.4 Blok diagram QbSH …….……..……...11

Gambar 3.1 Diagram sistem QbSH.....14

Gambar 3.2 Diagram Alir Pengolahan *Query*……15

Gambar 3.3 Proses AMDF.....17

Gambar 3.4 Gambar Markov Chain.....20

Gambar 3.5 Gambar *left-right* CHMM.....21

Gambar 3.6 Gambar GUI training.….....22

Gambar 3.7 Gambar GUI test…..….....22

Gambar 4.1 Grafik MRR terhadap variasi *framing* dan *overlap*……...29

Gambar 4.2 Grafik pengaruh representasi melodi tehadap waktu training……29

Gambar 4.3 Grafik pengaruh representasi melodi tehadap MRR…….....30

Gambar 4.4 Grafik variasi *contour* dalam logDurasi…..……..……..30

Gambar 4.5 Grafik variasi *contour* dalam RDD.....31

Gambar 4.6 Grafik variasi *contour* dalam logRDD……...32

Gambar 4.7 Grafik representasi durasi dalam 3 level dan 5 level *contour*.….....32

Gambar 4.8 Grafik representasi durasi dalam 11 level *contour* dan PI….....33

Gambar 4.9 Grafik MRR terhadap pergeseran pv.…….…..... 35

DAFTAR TABEL

Halaman

Tabel 4.1 *Database*.....24

Tabel 4.2 MRR 3 level *contour*.....26

Tabel 4.3 MRR 5 level *contour*.....26

Tabel 4.4 MRR 7 level *contour*.....27

Tabel 4.5 MRR 9 level *contour*.....27

Tabel 4.6 MRR 11 level *contour*.....28

Tabel 4.7 MRR PI.....28

DAFTAR SINGKATAN

AMDF	*Average Magnitude Difference Function*
CHMM	*Continuous Hidden Markov Model*
MIDI	*Musical Digital Interface*
MRR	*Mean Reciprocal Rank*
PI	*Pitch Interval*
pv	*pitch vector*
RDD	*Relative Duration Difference*
QbSH	*Query by Singing/Humming*

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Keterbatasan sistem pencari lagu konvensional terletak pada metode pencariannya yang berupa *category-based browsing* atau *text-base searching* [1]. Sehingga untuk mencari sebuah lagu kita harus mengetahui judul lagunya atau informasi-informasi lain seperti: nama penyanyinya, nama albumnya dll.. Padahal adakalanya kita lupa judul atau informasi tambahan lain dari lagu yang ingin kita cari. Solusinya adalah sistem pencari yang berkerja berdasarkan isi lagu seperti: lirik, melodi, atau ritme dari lagu tersebut, salah satunya adalah sistem *Query by Singing/Humming* (QbSH).

Salah satu contoh sistem QbSH yang telah diaplikasikan adalah Midomi. Midomi adalah aplikasi web yang memungkinkan *user* mencari lagu dengan menyenandungkan atau menyanyikan lirik dari lagu yang ingin di-*download*. Selain itu ada juga aplikasi Mobile Midomi. Mobile Midomi mirip seperti Midomi namun penggunaannya pada ponsel, Ada pula sistem QbSH untuk aplikasi pemilihan *ringtone* ponsel. Sistem QbSH ini juga mungkin digunakan dalam sistem karaoke dimana *user* tidak perlu repot menekan tombol untuk mencari lagu yang diinginkan semua cukup dilakukan dengan menyenandungkan sedikit melodi dari lagu yang dicari. Bayangkan pula penggunaan QbSH pada MP3 player, ketika kita sedang mengendarai kendaraan atau sedang berlari pagi dan merasa terlalu repot untuk mengganti lagu apalagi untuk mencari lagu yang kita ingin dengar, kita cukup bersenandung dan lagu yang diinginkan langsung terputar.

Mayoritas penelitian sistem QbSH hanya menggunakan *feature pitch* saja dalam merepresentasikan melodi seperti dalam [1], [2] dan [3] padahal durasi juga merupakan karakteristik dari melodi [4]. Karena itu penulis mencoba menggunakan kedua feature ini. *Pitch* dapat direpresentasikan dengan berbagai cara seperti dengan menggunakan *absolute pitch*, *absolute pitch* yang menggunakan durasi, direpresentasikan dengan intervalnya atau direpresentasikan dengan konturnya [1]. Metode yang digunakan untuk pengambilan *feature* adalah

Average Magnitude Difference Function (AMDF) karena metode ini dikenal sederhana dan prosesnya cepat [5]. Cara perepresentasian melodi yang akan digunakan adalah bentuk kontur karena bentuk ini lebih sederhana dan lebih toleran terhadap kesalahan pergeseran nada pada penyenandungan [4]. Metode pencocokan melodi yang digunakan adalah *Continous Hidden Markov Model* (CHMM). CHMM dipilih karena performa HMM yang dikenal andal dalam *speech recognition*, selain itu dengan CHMM *user* tidak perlu bersenandung dengan "da da da" atau "la la la" [2].

Sistem ini juga akan dicobakan dalam musik khas Indonesia yaitu dangdut. Dengan penelitian tentang sistem QbSH untuk musik dangdut masih jarang. Diharapakan akan bisa menjadi fondasi untuk menghasikan aplikasi seperti Midomi untuk musik Indonesia, selain itu diharapkan pula -bila hasilnya yang memuaskan- akan membantu pelestarian musik dangdut yang merupakan ciri khas bangsa Indonesia.

1.2 Perumusan Masalah

Masalah pokok yang akan dibahas pada skripsi ini adalah bagaimana cara terbaik untuk merepresentasikan melodi dangdut kedalam *pitch* dan durasi. Musik dangdut memiliki dinamika *pitch* dan durasi yang bervariasi sehingga sulit untuk dinyanyikan dengan baik karena itu agar hasil pencarian optimal *pitch* dan durasi perlu direpresentasikan kedalam bentuk yang memiliki toleransi tinggi terhadap pergeseran nada dan durasi namun tetap menjaga karakteristik lagu.

1.3 Tujuan

Tujuan dari skripsi ini adalah untuk merancang sistem QbSH untuk musik dangdut dengan menggunakan *pitch* dan durasi sebagai *feature* untuk dikenali. Selain itu skripsi ini juga bertujuan membandingkan unjuk kerja presentasi *pitch* yang berupa: *pitch interval*, 3, 5, 7, 9 dan 11 level kontur yang masing-masing dikombinasikan dalam durasi yang direpresentasikan dalam bentuk logDurasi, *ratio duration difference*(RDD) dan logRDD yang dikuantisasikan.

1.4 Pembatasan Masalah

Sistem QbSH yang direncanakan terbatas pada sistem yang memiliki spesifikasi sebagai berikut;

1. *Database* yang digunakan adalah 15 MIDI Dangdut yang diubah menjadi bentuk monoponik. MIDI digunakan karena *file*-nya yang kecil sehingga mempercepat proses pencarian selain itu pengambilan *feature* pada MIDI lebih mudah.

2. *Query* dari *user* dilakukan selama 10 detik disimpan dalam format wav dengan frekuensi *sampling* 8000 Hz.

3. Metode pengambilan feature yang digunakan adalah AMDF untuk *pitch tracking* dan *note segmentation by amplitude* dan *by pitch* pada *note segmentation.*

4. Untuk *matching engine* metode yang digunakan adalah CHMM.

1.5 Sistematika Penulisan

Untuk mempermudah penulisan dan agar pembahasan yang disajikan lebih sistematis, maka laporan ini dibagi ke dalam lima bab. Isi masing – masing bab diuraikan secara singkat dibawah ini :

BAB 1 PENDAHULUAN

Pada bab ini dijelaskan tentang alasan mengambil topik ini sebagai skripsi, apa saja yang akan dibahas dalam skripsi ini beserta tujuannya, dan sistematika penulisan skripsi.

BAB 2 MELODI, REPRESENTASI MUSIK dan QbSH

Pada bab ini dijelaskan tentang melodi, karakteristik nada, persepsi melodi, perepresentasian melodi, *melodic similarity* beserta alasan melodi digunakan sebagai dasar pencarian dalam sistem ini. Akan dipaparkan pula mengenai format-format musik yang digunakan dalam QbSH yaitu *audio format* (wav, mp3 dll.) dan MIDI juga penjelasan mengenai sistem QbSH.

BAB 3 PERANCANGAN SISTEM QbSH

Bab ini menjelaskan tentang perancangan sistem QbSH seperti *database* apa yang digunakan, bagaimana mengambil *feature* melodi dari *database* dan senadungan, bagaimana merepresentasikan *feature* tersebut, bagaimana proses pencocokan melodinya dan metode apa yang digunakan untuk proses ini.

BAB 4 HASIL SIMULASI DAN ANALISIS

Bab 4 berisi hasil simulasi yang dilakukan dan analisis dari hasil simulasi tersebut.

BAB 5 KESIMPULAN

Bab 5 berisi tentang kesimpulan dari hasil yang diperoleh pada bab sebelumnya.

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

Timbre adalah warna suara, *timbre* merupakan pembeda antara dua jenis sumber bunyi. *Timbre*-lah yang membuat kita bisa membedakan suara gitar atau suara piano walaupun memainkan nada yang sama.

Nada dalam musik hanya ada 12 saja, dilambangkan dengan huruf A sampai G, yaitu C – C# - D – D# - E – F – F# - G – G# - A – A# - B – (Kembali ke) C. Antara nada ke nada berikutnya didefinisikan berjarak ½ (setengah), Sebagai contoh, nada C menuju C# berjarak setengah, C menuju D berjarak satu, C menuju D# berjarak satu setengah dan seterusnya [3].

2.1.2 Persepsi melodi

Melodi adalah bagian dari musik yang paling mempengaruhi perasaan manusia. Bila kita mendengar melodi dari suatu lagu kita bisa langsung mengetahui apakah itu lagu sedih atau lagu yang bersemangat. Selain itu melodi merupakan bagian yang paling berkesan dalam suatu lagu sehingga jadi bagian yang pertama dihafal dari suatu lagu.

Tune adalah melodi yang paling *simple*, mudah dinyanyikan dan paling *catchy* [1]. Melodi biasanya terdiri dari berbagai *tune* dengan tambahan nada-nada lain. *Motif* adalah bagian khusus dalam melodi yang sering diulang-ulang dalam komposisi musik. Biasanya motif ini tidak sepanjang *tune* bahkan mungkin hanya sepanjang dua nada.

Tekstur adalah gabungan dari beberapa suara dan melodi dalam suatu lagu. Monoponik adalah contoh dari tekstur yang paling *simple*. Pada monoponik hanya ada satu melodi saja. Bila satu melodi diiringi oleh suara lain *chord* contohnya, sebutanya homoponik. Ketika ada dua atau lebih melodi dimainkan secara bersama-sama maka tekstur ini disebut poliponik.

Persepsi seseorang terhadap melodi dari suatu lagu itu berbeda-berbeda. Karena itu melodi disebut *perceptual feature* dari musik. Yang membuat melodi mudah diingat dan menarik adalah kombinasi dari *melodic contour* dan ritme. *Melodic contour* adalah perubahan dari nada dalam melodi. Pada 3 *level contour* pergerakan dari nada disimbolkan dengan + (untuk menujukan naik) –

(menujukan nada turun) 0 (tidak ada perubahan nada). Ritme adalah pola khusus dari musik yang berkaitan dengan waktu.

Ada beberapa aspek yang berkaitan dengan sifat persepsi dari melodi ini yang harus diperhatikan dalam sistem QbSH [1]:

1. Menentukan bagian mana dari melodi yang paling diingat orang.
2. Dalam musik melodi telah tercampur dengan bermacam-macam nada lain bahkan kadang ada lebih dari dua melodi pada suatu lagu, sangat penting untuk menentukan melodi mana yang lebih dominan dan memiliki kemungkinan untuk diingat orangnya labih besar.
3. Perlu ditekankan bahwa system QbSH mencari kemiripan saja bukan mencari melodi yang sama persis.

2.1.3 *Melodic similarity*

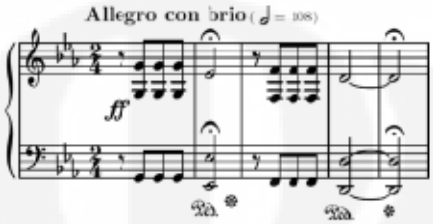
Yang menjadi inti dalam sistem QbSH adalah mencari kesamaan antara melodi dari *database* dengan melodi dari senandungan *user*. Alasan digunakannya melodi adalah:

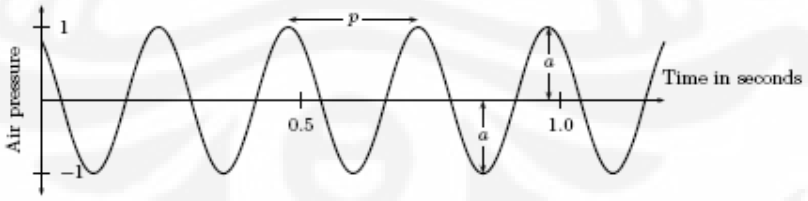
1. Kita secara tidak sadar kadang menyenandungkan melodi atau bersiul sambil mendengar lagu. Jadi melodi ini terekam secara tidak langsung dalam alam bawah sadar manusia.
2. Melodi merupakan ciri khusus suatu lagu.

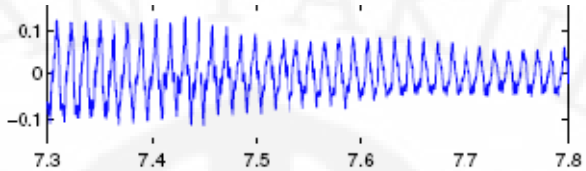
Faktor-faktor yang menjadi pertimbangan dalam menetukan kemiripan melodi ini adalah kontur, *pitch* dan transposisi nada [1].

2.2 Representasi musik

Perkembangan teknologi menyebabkan munculnya berbagai macam bentuk media penyimpanan musik. Dulu musik disimpan dalam bentuk tertulis (*score*) sehingga bila ingin menikmati musik yang kita simpan harus terlebih dahulu menerjemahkan s*core* tersebut/dimainkan. Sekarang musik tidak hanya disimpan dalam bentuk *score*, ada format midi, wav, mp3 dll. Yang memungkinkan musik yang disimpan bisa juga dinikmati dengan mudah. Secara


Allegro con brio
ff


1
−1
Air pressure
p
a
a
0.5
1.0
Time in seconds


0.1
0
−0.1
7.3
7.4
7.5
7.6
7.7
7.8

2.2.2 MIDI

MIDI merupakan pendigitalan dari format *score*. MIDI merupakan kepanjangan dari *Musical Instrument Digital Interface*. MIDI adalah suatu standar yang memungkinkan berbagai instrumen musik *digital* dari berbagai merek bisa dimainkan bersama.

MIDI memungkinkan musisi mengontrol instumen *digital* secara *real time*. Contohnya pada piano *digital*, ketika kita menekan tuts piano *digital*, piano langsung mengeluarkan bunyi sesuai dengan nada dari tuts yang ditekan intesitas bunyi juga bisa diatur dengan mengatur kecepatan menekan tuts. Ketika tuts dilepas bunyi juga berhenti. Atau bisa juga bila malas menekan tus kita bisa memberikan *input* berupa MIDI *messages* yang berisi informasi *note-on*, kecepatan dan *note-off* [7].

Sama seperti s*core*, MIDI hanya merupakan pesan yang menunjukan bagaimana musik dimainkan dan bagaimana cara memainkannya. Dalam MIDI tiap-tiap *timbre* memiliki *channel* masing-masing sehingga sehingga analisis dalam MIDI lebih mudah.

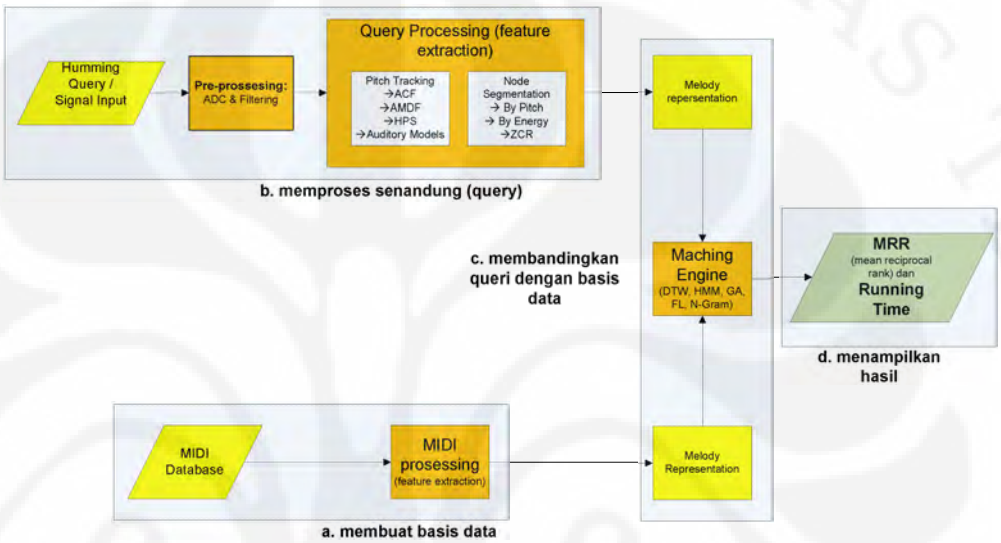
2.3 *Query by Singing/Humming* (QbSH)

Query by content adalah pencarian lagu berdasarkan isi/konten dari lagu, Berbeda dengan sistem pencari lagu kovensional mencari berdasarkan data yang disisipkan saja (judul, nama penyanyi, dll) [8] dengan kata lain bila kita tidak mengetahui judul, nama penyanyi, dll. Kita harus mencari seluruh *database* untuk mencari lagu yang diinginkan, hal ini sangat merepotkan. Padahal yang pertama diingat dari lagu adalah isi dari lagu tersebut, seperti; lirik, melodi dan ritme. Jadi yang seharusnya menjadi dasar pencarian adalah konten dari lagunya. Sistem yang melakukan pencarian lagu berdasarkan konten/isi dari lagu disebut s*ystem query by content. System query by content* yang akan dibahas dalam seminar ini adalah *query by singing/humming* (QbSH).

Inti dari sistem QbSH adalah pencocokan antara melodi yang disenandungkan oleh *user* dengan melodi dari *database*, kemudian sistem akan

me-*ranking* lagu dari yang termirip diharapkan lagu yang cari ada dalam peringkat pertama.

Maksud dari diberikan *ranking* bukannya hanya satu kandidat adalah untuk mengantisisipasi bila ada beberapa versi lagu atau ada lagu yang memiliki kemiripan melodi. Gambar 2.4 menunjukan *block* diagram QbSH [9].



Gambar 2.4 Blok diagram QbSH [9]

2.3.1 *Database*

Untuk membuat *database* dibutuhkan lagu. Pada umumnya untuk *database* dari sistem QbSH digunakan format MIDI monoponik. Penggunaan format ini dimaksudkan untuk memudahkan dalam pengambilan *feature* dari sebuah lagu.

Dalam membuat *database* ini sebelumnya perlu diperhatikan pemilihan *feature* yang akan digunakan sebagai data dalam *database*. Seperti yang telah dijelaskan sebelumnya *feature* dapat berupa *pitch*, interval, *contour*, tempo, birama, dan lain-lain. Selanjutnya *feature* yang telah diekstrak dari lagu akan menjadi representasi melodi dan akan dibandingkan dengan melodi dari *query*.

2.3.2 Pemerosesan *Query*

Query dari *user* berformat WAV. Selanjutnya dari format WAV ini akan diambil *feature* yang sama seperti *feature* yang diambil pada *database*, sehingga akan diperoleh representasi melodi dari *query*. Namun seperti yang telah dijelaskan sebelumnya, pada *database* format yang digunakan adalah MIDI sedangkan pada *query* format yang digunakan adalah WAV, maka proses yang dilakukan untuk pengambilan/ekstraksi *feature* akan berbeda. Ekstraksi *feature* dari format WAV memiliki proses yang cenderung lebih kompleks. Pengambilan *feature* ini yang akan menjadi sistem utama dari penulis.

Proses ekstraksi *feature* biasanya dimulai dengan tahap *pitch tracking* (pengambilan *pitch*) dan *note segmentation* (pemisahan antar nada).

2.3.2.1 *Pitch Tracking*

Pitch tracking adalah metode untuk mendapatkan frekuensi dasar (*fundamental frequency/*F0) dari suatu sinyal suara, yang nantinya menjadi *feature* untuk representasi melodi. *Pitch Traking* yang digunakan dalam sistem ini adalah AMDF.

2.3.2.1 *Note Segmentation*

Note segmentation adalah suatu metode untuk mendapatkan durasi dari suatu nada atau dalam dunia sistem disebut juga sebagai nilai nada. *Note segmentation* berkaitan erat dengan *onset* dan *offset detection*. *Onset* adalah penanda dari mulainya suatu nada, sedangkan *offset* adalah penanda dari berakhirnya suatu nada. *Note segmentation* juga berfungsi sebagai penanda dalam *signal* berada dalam keadaan diam, terkena *noise*, atau terdapat suara.

Note segmentation dapat dilakukan dengan dua cara, yaitu, Segmentasi berdasarkan ampiltudo dan berdasarkan *pitch.*Segmentasi berdasarkan amplitudo adalah pembagian segmen dengan melihat amplitudo sinyal dalam domain waktu. Proses segmentasi ini *simple* dan sangat mudah di implementasikan. Namun segmentasi ini tidak tepat jika digunakan dalam kondisi *real-time* [5].

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 mempunyai keuntungan jika digunakan dalam sistem yang *real-time* dan tidak mempedulikan cara bersenandungnya.

2.3.3 Proses Pencocokan Melodi

Bagian ini merupakan inti dari sistem QbSH, karena pada bagian ini dilakukan pencarian lagu berdasarkan masukan *query* yang diberikan. Bagian ini juga merupakan bagian dari sistem yang paling banyak mengkonsumsi waktu. Berapa metode yang dapat digunakan untuk membandingkan *query* dengan *database* antara lain [7]:

a) Dynamic Time Warping (DTW)
b) Hidden Markov Models (HMM)
c) Genetic Algorithm (GA)
d) N-Gram Model
e) Fuzzy Logic

2.3.4 Menampilkan Hasil

Hasil penelitian dari sistem QbSH masih jauh dari sempurna, pencarian yang selalu menghasilkan lagu yang dicari selalu dalam peringkat pertama sampai saat ini belum ada. Oleh karena itu, hasil yang ditampilkan biasanya masih dalam bentuk daftar sepuluh hasil terbaik yang mendekati *query*. Nilai yang menunjukan hasil kinerja dari sistem QbSH adalah *mean reciprocal rank* (MRR) [7], yang dirumusan sebagai berikut:

$$MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{r_i} \qquad (2.2)$$

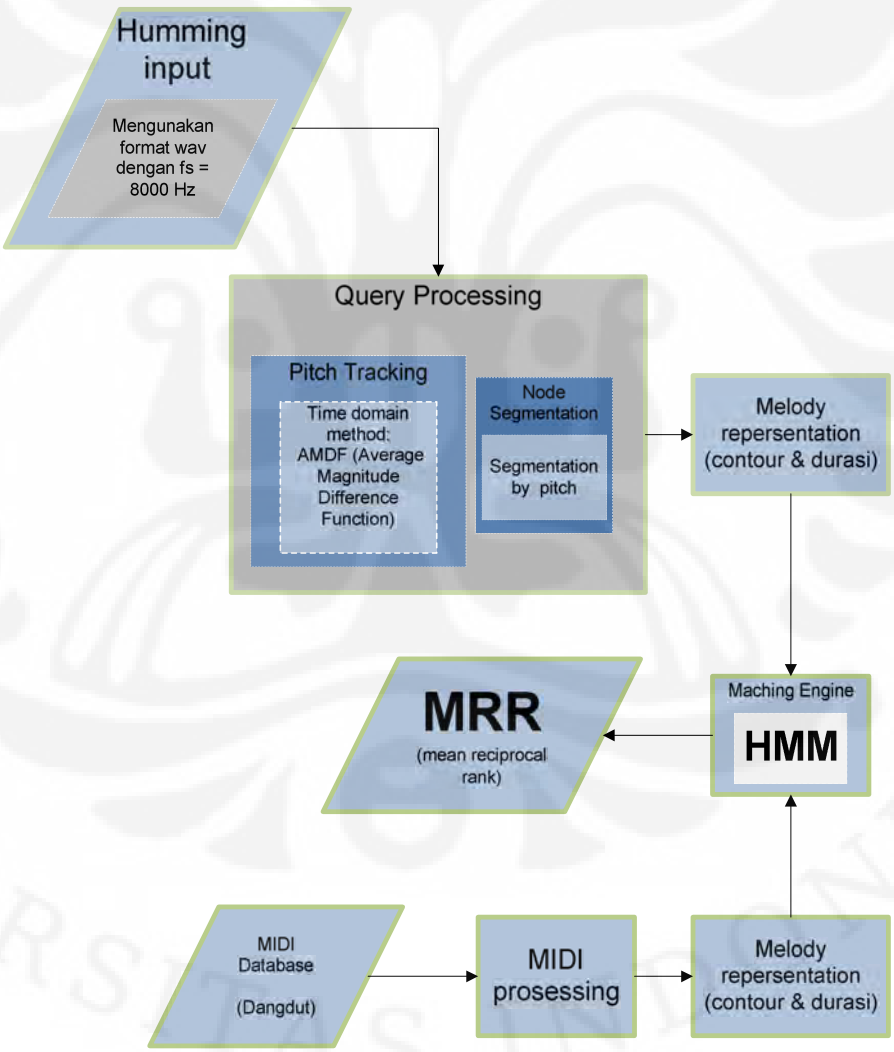
Dimana, r_i adalah peringkat dari lagu yang tepat untuk *query* ke-i untuk jumlah *query* yang dilakukan sebesar n.

BAB 3

PERANCANGAN SIMULASI SISTEM QbSH

3.1 Perancangan Sistem QbSH

Pada sistem QbSH ini *input* berupa *query user* simpan dalam format wav kemudian *input* ini akan di ekstrak *feature-feature*-nya agar didapatkan representasi melodi yang efektif. Selanjutnya representasi melodi dari senandung *user* ini akan dibandingkan dengan *database*. Gambar 3.1 merupakan modifikasi dari [9] dan menjadi blok diagram dari sistem ini.



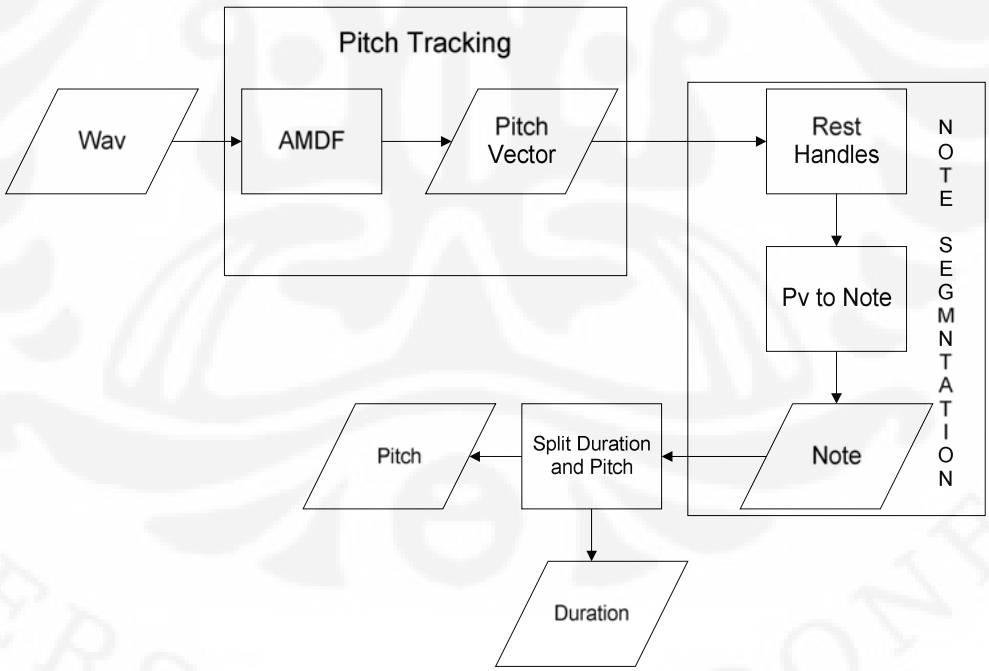
Gambar 3.1 Diagram sistem QbSH [9].

3.2 Database

Untuk membuat *database* sistem QbSH dibutuhkan MIDI monoponik. Namun, mencari MIDI dangdut monoponik itu sulit jadi kami menggunakan MIDI dangdut poliponik yang diubah menjadi monoponik dengan mengambil *channel* melodinya saja. Kemudian kumpulan MIDI monoponik ini dibentuk menjadi *database* MATLAB dengan bantuan *MIDI Toolbox* [11]. Dari *database* ini diambil *pitch vectornya* kemudian *pitch vector* ini diubah menjadi bentuk note. Lalu dari note ini diambil *pitch* dan durasinya.

3.3 Pemerosesan *query*

Query user disimpan dalam format wav, frekuensi s*ampling* sebesar 8000 Hz dengan tipe *channel mono*. Pengolahan *query* ini dilakukan dengan menggunakan bantuan *utility toolbox* [11], *speech and audio processing toolbox* [12], dan *melody recognition toolbox* [13]. Gambar 3.2 menunjukan diagram alir pemrosesan *query*.



Gambar 3.2 Diagram Alir Pengolahan *Query*

3.3.1 Average Magnitude Difference Function (AMDF)

Metode *pitch tracking* yang digunakan adalah *time domain method* AMDF [5]. Pada metode AMDF periode dasar didapat dari mengukur selisih dalam domain waktu dari suatu sinyal dengan sinyal tersebut ketika mengalami pergeseran waktu (*time-shifted version*). Dari periode dasar ini kemudian diperoleh frekuensi dasar.

Dalam metode AMDF sinyal dibagi menjadi beberapa *frame.* Ukuran *frame* yang digunakan tergantung dari rentang frekuensi dasar yang ingin diperoleh. Pada sistem QbH ini akan diambil frekuensi dasar 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 [5].

$$frame_{\max} = 2 \bullet \frac{f_s}{f_{0\min}} \tag{3.1}$$

$$frame_{\max} = 2 \bullet \frac{f_s}{f_{0\min}} = 2 \bullet \frac{8000}{62.5} = 256 \text{ sample point}$$

Dengan demikian ukuran *frame* yang dapat digunakan adalah minimal berukuran 256 *sample point* dan untuk mencegah diskontinuitas maka antar *frame* dibuat *overlap.*

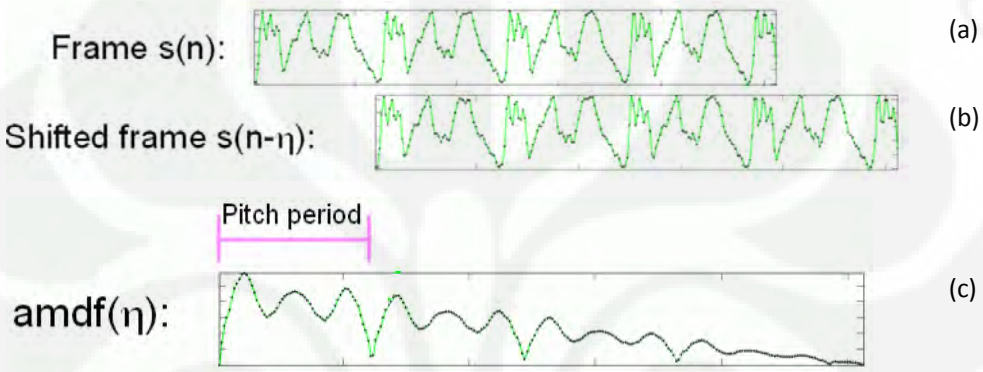
Setelah mendapatkan besar *frame* periode dasar dari setiap *frame* dicari dengan proses sebagai berikut:

1. Diketahui suatu sinyal dalam suatu *frame* s(n)
2. Sinyal s(n) tersebut digeser sebanyak η sehingga didapatkan sinyal s(n- η)
3. Menghitung AMDF (*average magnitude difference function*) untuk setiap nilai η yang dimulai dari satu sampai fungsi AMDF menghasilkan nilai yang minimal.

$$\text{AMDF}(\eta) = \sum_{n=t}^{t+frame-1} \left(s(n) - s(n+\eta) \right)^2 \tag{3.2}$$

4. Nilai η dimana menghasilkan besar fungsi AMDF yang minimal merupakan periode dasar.

Setelah mendapatkan periode dasar maka akan didapatkan besar frekuensi dasar yang merupakan *pitch* pada *frame* tersebut [7]. Gambar 3.3 menunjukan proses AMDF. Keluaran proses *pitch tracking* ini berupa *pitch vector*. *Pitch vector* merupakan rentetan *pitch* dalam satuan waktu.



Gambar 3.3 Proses AMDF (a) sinyal satu *frame* (b) Sinyal s(n) digeser sebanyak η (c)sinyal hasil AMDF [9].

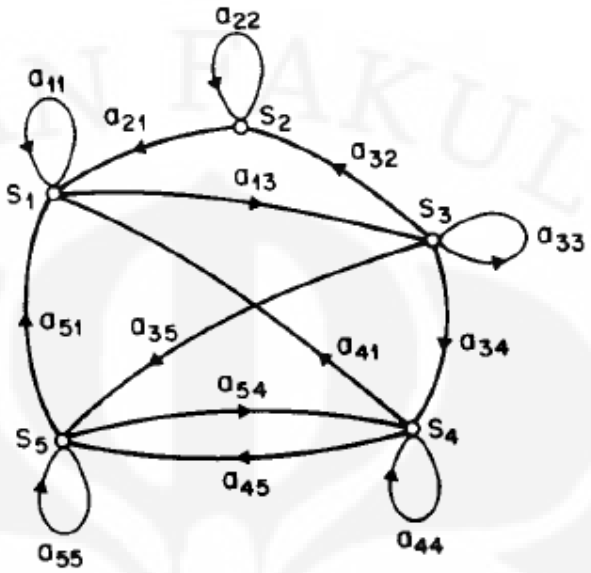
3.3.2 Note Segmentation

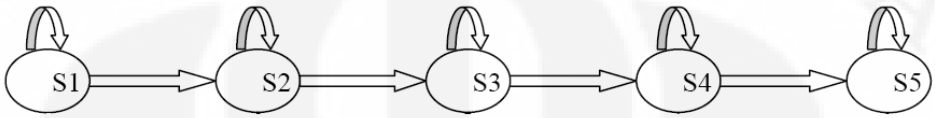
Setelah didapat *pitch vector* (pv) proses yang dilakukan setelahnya adalah *note segmentation*. Tujuan dari *note segmentation* ini adalah untuk mendapatkan *note* dari *query*. *Note* menunjukan *pitch* beserta lama *pitch* itu dimainkan (durasinya). Metode *note segmentation* yang digunakan adalah *segmentation by pitch*. Dimana *pitch* berurutan yang memiliki kemiripan nilai dalam *pitch vector* disatukan dan dihitung berapa durasi *pitch* tersebut.

Sebelum disegmentasi terlebih dahulu *silent* dihilangkan dari *query* dengan fungsi *rest handle* [13] untuk mendapatkan pv yang murni berisi *pitch*. Selanjutnya pv yang telah dihilangkan silentnya diubah menjadi note menggunakan fungsi *pv2note* dari *toolbox* [13].

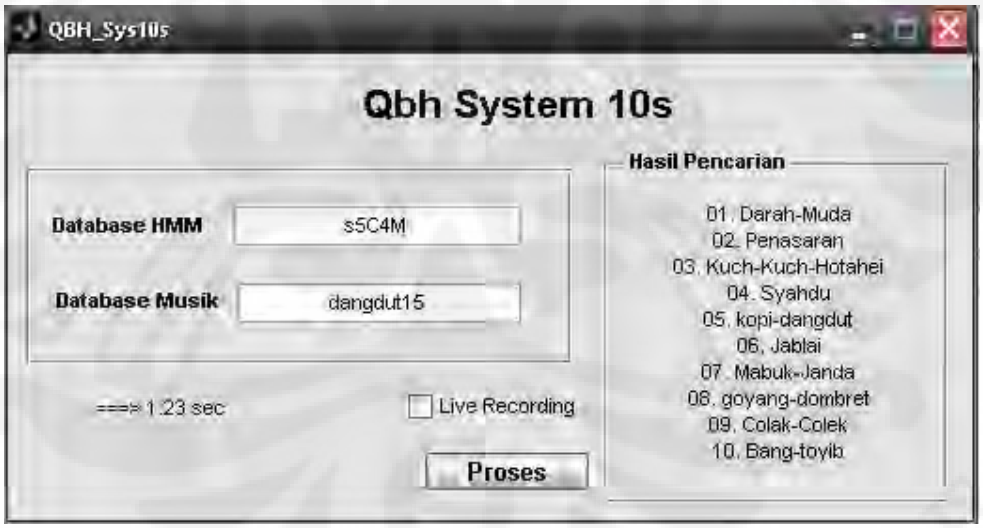
$$\mathrm{RDD}(k) = \frac{t_k}{t_{k-1}}$$

$$ratio\ (i) = round\left(10 \cdot \log_{10}\left(\frac{duration\ (i+1)}{duration\ (i)}\right)\right)$$


a22
a11
a21
S2
a32
a13
S1
S3
a33
a51
a35
a41
a34
a54
S5
S4
a45
a55
a44


S1
S2
S3
S4
S5


training_data
Training Database
Input Lagu
dangdut15
Output / HMM data
7CD7M
Proses Selesai
===> 629.13 sec
Proses


QBH_Sys10s
Qbh System 10s
Database HMM
s5C4M
Database Musik
dangdut15
===> 1.23 sec
Live Recording
Proses
Hasil Pencarian
01. Darah-Muda
02. Penasaran
03. Kuch-Kuch-Hotahei
04. Syahdu
05. kopi-dangdut
06. Jablai
07. Mabuk-Janda
08. goyang-dombret
09. Colak-Colek
10. Bang-toyib

MRR diukur berdasarkan urutan kemunculan lagu yang dicari dari senandung yang dinyanyikan. Semakin tinggi MRR maka semakin akurat sistem QbSH tersebut.

2. Running time

   Running time diukur dari pemrosesan *query* hingga tampilnya hasil dari *query*. Semakin kecil waktu *running time*, semakin efektif pencarian yang dilakukan oleh sistem QbSH.

3.7 Simulasi

Simulasi akan dilakukan dengan menggunakan *software* MATLAB versi 7.1 dengan menggunakan *signal processing toolbox, melody toolbox*, dan *HMM tool box*.

Proses simulasi tidak dilakukan secara *real time*, dimana tahap *pre-processing* dilakukan secara manual dengan menggunakan *software ADOBE AUDITION*.

Skenario dari simulasi ini adalah mencari kombinasi perepresentasian *pitch* dan durasi antara PI, 3 *level*, 5 *level*, 7 *level*, 9 *level* dan 11 *level* kontur dengan durasi, RDD, dan logRDD. Selain itu akan dicoba juga variasi framing 256, 320 dan 512, serta overlap 64 dan 128. Kemudian diamati mana kombinasi yang memberikan MRR terbesar.

BAB 4

HASIL SIMULASI DAN ANALISIS

4.1 Hasil Simulasi

Simulasi yang dilakukan pada penelitian ini adalah mengetes sistem QbSH menggunakan 60 *query* yang dinyanyikan oleh 3 laki-laki dan 3 wanita dalam masing-masing memilik 10 lagu secara acak dari *database*. Tabel 4.1 merupakan *list* lagu dalam *database*

Tabel 4.1 *Database*

No.	Lagu
1	Aduh Buyung
2	Bang Toyib
3	Colak-colek
4	Darah Muda
5	Jablai
6	Jatuh Bangun Aku
7	Kuch-Kuch Hotahei
8	Mabuk Janda
9	Penasaran
10	SMS
11	Syahdu
12	Goyang Dombret
13	Kopi Dangdut

14	Mbah Dukun
15	Sekuntum Mawar Merah

Untuk mendapatkan konfigurasi terbaik untuk *database* diatas akan divariasikan representasi *pitch* dan durasi dengan *framing* 512 dan *overlap* 64 setelah didapat representasi yang terbaik dilakukan variasi *framing* dan *overlap*nya untuk mengoptimalkan hasil. Keseluruhan pengujian meliputi:

1. Uji coba sampel dalam 3 *level contour* dan logDurasi.
2. Uji coba sampel dalam 5 *level contour* dan logDurasi.
3. Uji coba sampel dalam 7 *level contour* dan logDurasi.
4. Uji coba sampel dalam 11 *level contour* dan logDurasi.
5. Uji coba sampel dalam PI dan logDurasi.
6. Uji coba sampel dalam 3 *level contour* dan RDD.
7. Uji coba sampel dalam 5 *level contour* dan RDD.
8. Uji coba sampel dalam 7 *level contour* dan RDD.
9. Uji coba sampel dalam 11 *level contour* dan RDD.
10. Uji coba sampel dalam PI dan RDD.
11. Uji coba sampel dalam 3 *level contour* dan logRDD.
12. Uji coba sampel dalam 5 *level contour* dan logRDD.
13. Uji coba sampel dalam 7 *level contour* dan logRDD.
14. Uji coba sampel dalam 11 *level contour* dan logRDD.
15. Uji coba sampel dalam PI dan logRDD.
16. Uji coba sampel representasi melodi terbaik dengan variasi *framing* dan *overlap* sebagai berikut:
    a. *Frame* 256 *overlap* 64.
    b. *Frame* 256 ovelap 128.
    c. *Frame* 320 *overlap* 64.
    d. *Frame* 320 ovelap 128.
    e. *Frame* 512 *overlap* 64.
    f. *Frame* 512 ovelap 128.

4.1.1 Hasil Uji 3 *Level Contour*

Pada Tabel 4.2 dapat dilihat hasil uji coba sistem QbSH untuk 3 *level contour*.

Tabel 4.2 MRR 3 *level contour*

3 *level*	MRR						
	Nama						Total
	Indra W	Bani	Indra	Anti	Cindy	Anne	
logDurasi	0.158	0.3127	0.1561	0.1446	0.1797	0.1323	0.180567
RDD	0.2048	0.3455	0.1987	0.2198	0.3276	0.2377	0.255683
logRDD	0.2928	0.2599	0.1953	0.1557	0.3287	0.2124	0.2408

4.1.2 Hasil Uji 5 *Level Contour*

Pada Tabel 4.3 dapat dilihat hasil uji coba sistem QbSH untuk 5 *level contour*.

Tabel 4.3 MRR 5 *level contour*

5 *level*	MRR						
	Nama						Total
	Indra W	Bani	Indra	Anti	Cindy	Anne	
logDurasi	0.1372	0.3284	0.1853	0.2435	0.1615	0.1581	0.202333
RDD	0.2665	0.2607	0.2359	0.395	0.2346	0.201	0.265617
logRDD	0.2613	0.2222	0.3414	0.2955	0.2293	0.1541	0.250633

4.1.3 Hasil Uji 7 *Level Contour*

Pada Tabel 4.4 dapat dilihat hasil uji coba sistem QbSH untuk 7 *level contour*.

Tabel 4.4 MRR 7 *level contour*

7 *level*	MRR						
	Nama						Total
	Indra W	Bani	Indra	Anti	Cindy	Anne	
logDurasi	0.2248	0.266	0.1898	0.1362	0.2131	0.2289	0.2098
RDD	0.1893	0.1552	0.2374	0.1711	0.1891	0.1717	0.185633
logRDD	0.1753	0.3031	0.2594	0.2282	0.2608	0.1583	0.23085

4.1.4 Hasil Uji 9 *Level Contour*

Pada Tabel 4.5 dapat dilihat hasil uji coba sistem QbSH untuk 9 *level contour*.

Tabel 4.5 MRR 9 *level contour*

9 *level*	MRR						
	Nama						Total
	Indra W	Bani	Indra	Anti	Cindy	Anne	
logDurasi	0.2308	0.2361	0.165	0.2813	0.1561	0.2055	0.212467
RDD	0.2123	0.2348	0.1842	0.1735	0.2422	0.2018	0.208133
logRDD	0.2191	0.2251	0.2402	0.2405	0.2693	0.2124	0.234433

4.1.5 Hasil Uji 11 *Level Contour*

Pada Tabel 4.6 dapat dilihat hasil uji coba sistem QbSH untuk 11 *level contour*.

Tabel 4.6 MRR 11 *level contour*

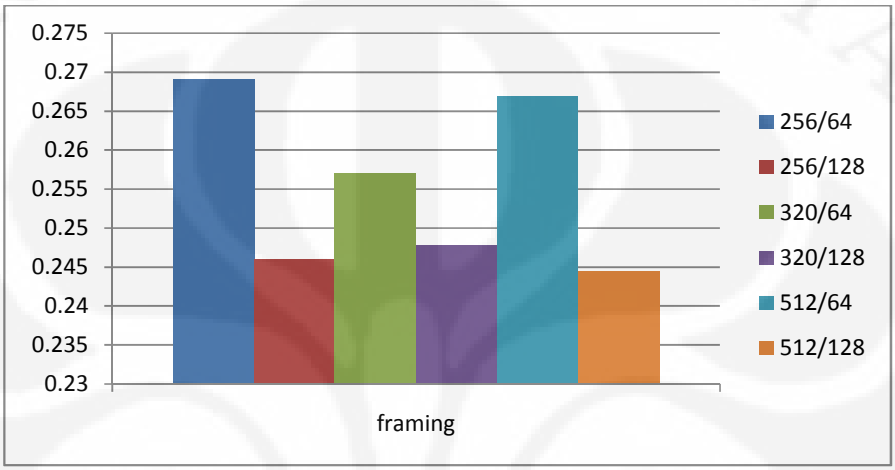
11 *level*	MRR						
	Nama						Total
	Indra W	Bani	Indra	Anti	Cindy	Anne	
logDurasi	0.2328	0.1726	0.2185	0.2432	0.2416	0.238	0.22445
RDD	0.224	0.1632	0.2144	0.3093	0.3059	0.2674	0.247367
logRDD	0.2193	0.2306	0.2614	0.3378	0.3446	0.2078	0.266917

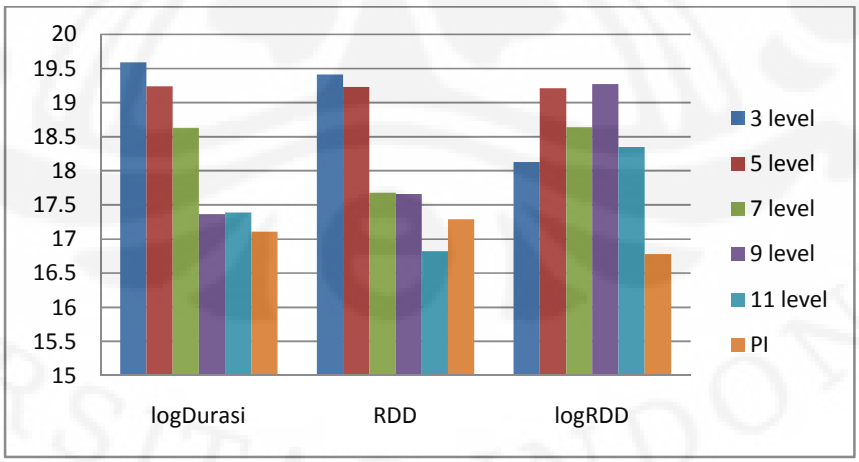
4.1.6 Hasil Uji *Pitch Interval* (PI)

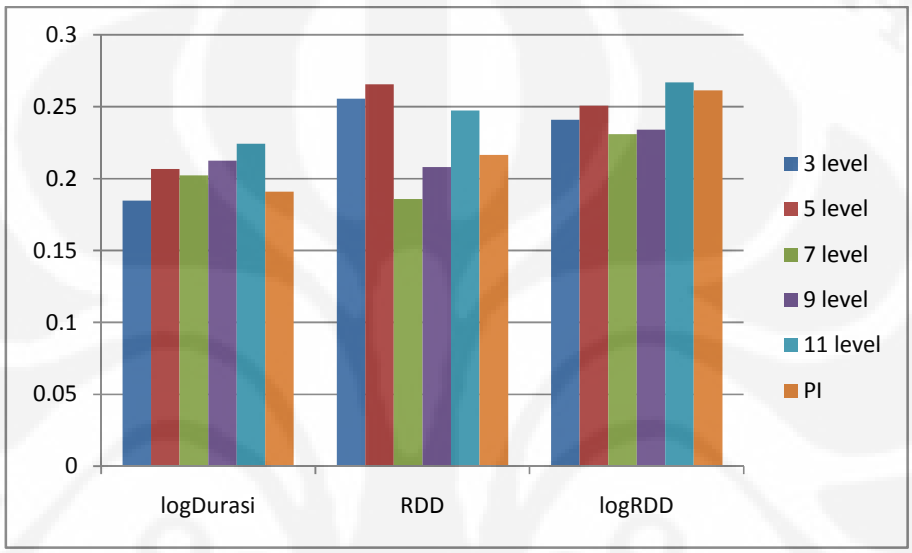
Pada Tabel 4.7 dapat dilihat hasil uji coba sistem QbSH untuk PI.

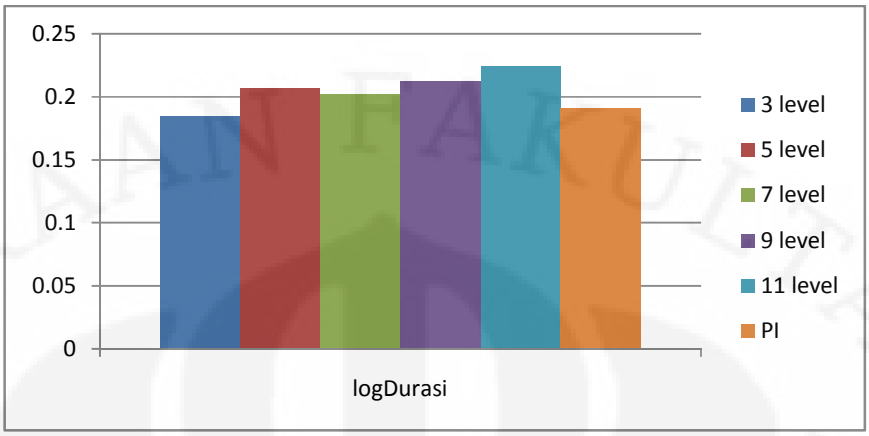
Tabel 4.7 MRR PI

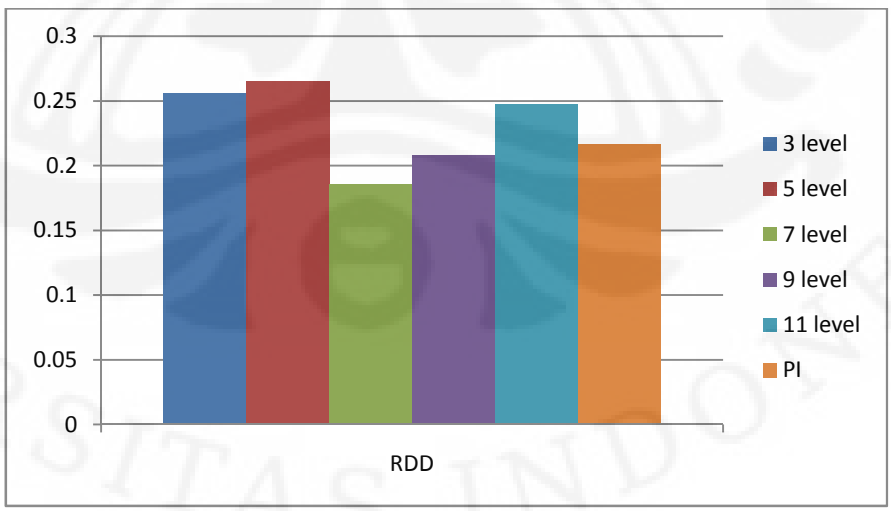
PI	MRR						
	Nama						Total
	Indra W	Bani	Indra	Anti	Cindy	Anne	
logDurasi	0.1591	0.2603	0.173	0.1838	0.1848	0.1846	0.190933
RDD	0.2146	0.213	0.2028	0.1707	0.236	0.2617	0.216467
logRDD	0.2328	0.2597	0.2342	0.2473	0.3062	0.2875	0.261283

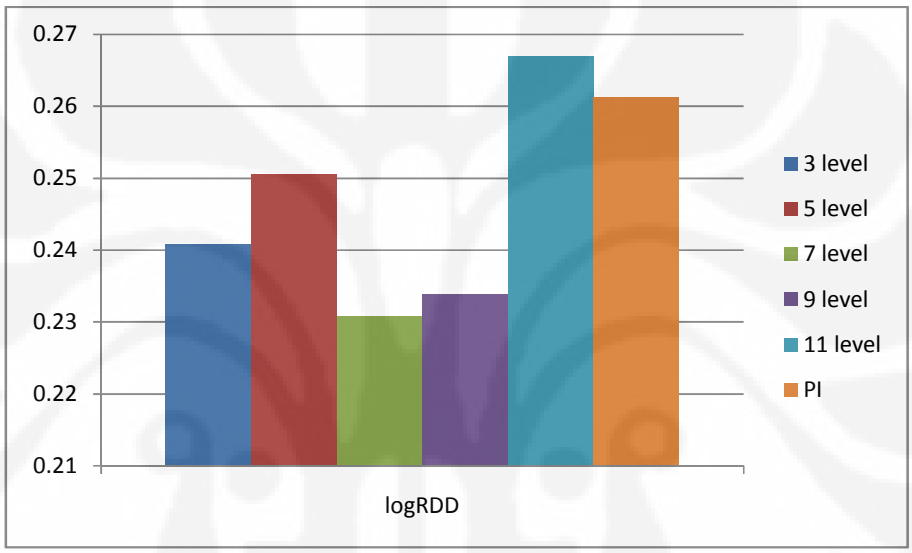

0.275
0.27
0.265
0.26
0.255
0.25
0.245
0.24
0.235
0.23
framing
256/64
256/128
320/64
320/128
512/64
512/128

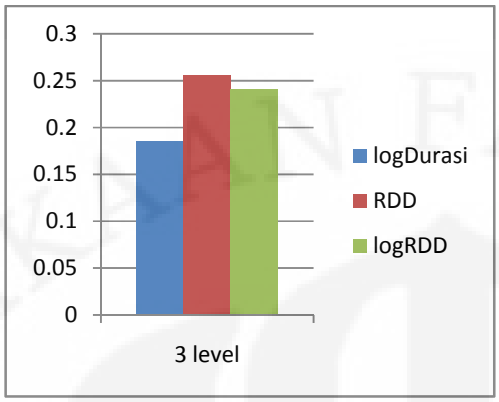

20
19.5
19
18.5
18
17.5
17
16.5
16
15.5
15
logDurasi
RDD
logRDD
3 level
5 level
7 level
9 level
11 level
PI

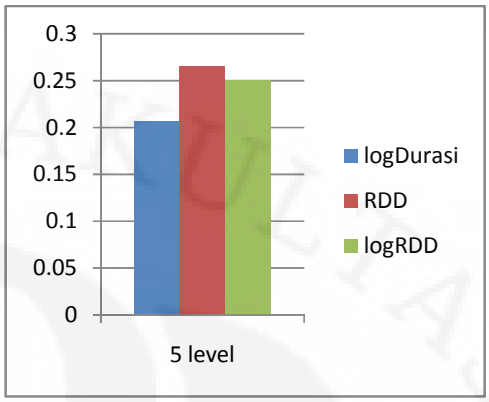

0.3
0.25
0.2
0.15
0.1
0.05
0
logDurasi
RDD
logRDD
3 level
5 level
7 level
9 level
11 level
PI

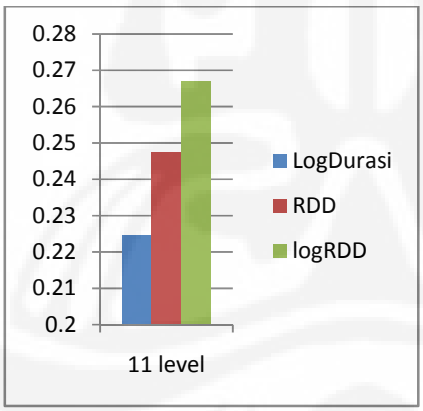

0.25
0.2
0.15
0.1
0.05
0
3 level
5 level
7 level
9 level
11 level
PI
logDurasi

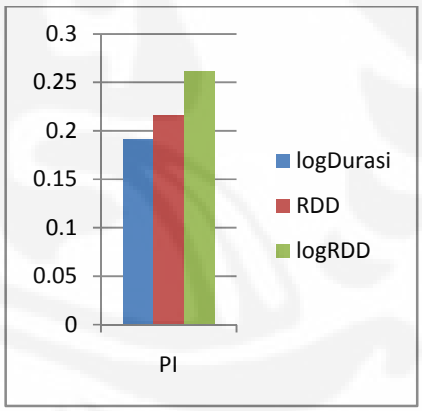

0.3
0.25
0.2
0.15
0.1
0.05
0
3 level
5 level
7 level
9 level
11 level
PI
RDD


0.27
0.26
0.25
0.24
0.23
0.22
0.21
logRDD
3 level
5 level
7 level
9 level
11 level
PI


0.3
0.25
0.2
0.15
0.1
0.05
0
logDurasi
RDD
logRDD
3 level


0.3
0.25
0.2
0.15
0.1
0.05
0
logDurasi
RDD
logRDD
5 level


0.28
0.27
0.26
0.25
0.24
0.23
0.22
0.21
0.2
LogDurasi
RDD
logRDD
11 level


0.3
0.25
0.2
0.15
0.1
0.05
0
logDurasi
RDD
logRDD
PI

training lebih lama disebabkan karena *contour* menganti lebih banyak variable dalam proses pe-*level*-annya.

Secara garis besar, terlihat semakin baik representasi dari *pitch* atau durasi maka semakin berkurang toleransinya terhadap *error* saat melakukan *query* dan terlihat pula representasi dari *pitch* yang detail dalam mengambarkan *database* bila dipasang dengan representasi dari durasi yang juga detail MRRnya tidak sebaik pasangan detail dan kurang detail.

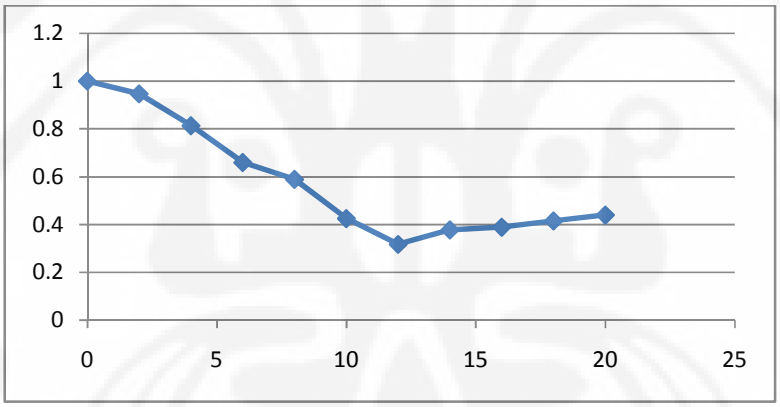
4.2.2 Analisa Pengaruh Framing dan *Overlap* Terhadap MRR

Dari grafik 4.1 didapat MRR terbaik yaitu 0.2691 dengan *Frame* 256 dan *overlap* 64 terlihat bahwa tidak ada pola khusus dalam pengaruh framing dan *overlap* terhadap MRR.

4.2.3 Analisa Performa

Dari hasil percobaan didapat 11 *level contour* dan logRDD menghasilkan MRR terbaik, namun hasilnya hanya 0.2691 atau lagu yang dicari berada sekitar peringkat 4 dalam setiap pencarian. Padahal performa HMM dalam *speech recognition* mencapai 95%. Hal ini terjadi karena 2 hal. Pertama karena sangat sulit untuk menyanyikan *query* seideal midi dan biasanya dalam penyanyian *query* terjadi hal berikut [18]:

a. Yang berkaitan dengan *pitch*:
    1. Transposisi: *Query* dinyanyikan dalam kunci yang berbeda dengan *database.*
    2. Modulasi: Pada beberapa bagian *query* ada yang kuncinya berubah atau dinyanyikan dengan nada yang berbeda dengan *database*.
b. Yang berkaitan dengan durasi:
    1. Pergeseran Tempo: *Query* dinyanyikan lebih lambat atau lebih cepat dari *database.*
    2. Perubahan Tempo: Pada beberapa bagian *query* ada dinyanyikan lebih lambat atau lebih cepat dari *database*.


1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25

BAB 5

KESIMPULAN

Berdasarkan hasil pengujian unjuk kerja pada sistem QbSH yang dilakukan dengan *database* dan *query* seperti dalam percobaan ini dapat diambil beberapa kesimpulan, yaitu:

1. Tingkat akurasi sistem QbSH dipengaruhi oleh jenis perepresentasian melodinya.
2. Ada *trade off* antara kedetailan karakteristik yang direpresentasian dengan toleransinya terhadap kesalahan penyanyian. Ketika representasi kurang menggambarkan *database* maka MRR akan jelek namun bila representasi terlalu menggambarkan *database* MRR pun tidak baik karena toleransi perepresentasian terhadap *error* berkurang.
3. logDurasi kurang baik dalam merepresentasikan durasi karena perepresentasian ini tidak toleran terhadap pergeseran durasi.
4. Perepresentasian *pitch* yang detail sebaiknya jangan dipasangkan dengan perepresentasian durasi yang detail, begitu juga sebaliknya.
5. Representasi terbaik untuk pecobaan ini adalah 11 *level contour* dan logRDD dengan MRR 0.267, diurutan kedua 5 *level contour* dan RDD dengan MRR 0.266.
6. Penyebab nilai MRR tidak begitu memuaskan karena:
    a. Perepresentasian tidak mampu mengatasi modulasi dan perubahan tempo pada *query*.
    b. Metode pencocokan *query* kurang toleran terhadap *error*.

DAFTAR REFERENSI

[1] Chai, Wei. (2001). *Melody Retrieval on The Web.* Thesis MIT. Diakses November 16, 2009 dari http://alumni.media.mit.edu/~chaiwei/papers/msthesis.pdf.

[2] 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

[3] Kusuma, W., & Thalib, F. (2006). *Representasi Nada Sinyal Suara Melodi Senandung sebagai fitur Identifikasi Lagu*. Seminar Ilmiah Nasional Komputer dan Sistem Inteligen (KOMMIT 2006). Diakses November 12, 2009 dari http://repository.gunadarma.ac.id:8000/Representasi_Chord_Wahyu_KusuKu_dkk_edit_835.pdf.

[4] Roy, Sumantra.D., & Rao, Preeti. (2005) *Melodic Contour-Based QBH System: Anaytica Modeling and Performance Evaluation*. Diakses Mei 25, 2010 dari http://www.cse.iitd.ac.in/sumantrapublicationsncc05_qbh.pdf

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

[6] Klapuri, A., & Davy, M. (2006). *Signal Proccesing Method for Music Transcription.* New York: Springer.

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

[8] Dickerson, K.B. (2009). *Musical Query by Content Using Self Organizing Maps*. Thesis Brigham Young University. Diakses November 11, 2009 dari http://contentdm.lib.byu.edu/ETD/image/etd2995.pdf

[9] I-Yang Lee. (1999). *Content-based music retrieval from acoustic input*. Thesis Department of Computer Science National Tsing Hua University. Diakses Desember 14, 2009 dari http://wayne.cs.nthu.edu.tw/~window/paper/thesis/cvgip99.doc.

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

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

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

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

[14] 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.

[15] Haus, Goffredo, & Pollastri, Emanuele. (2001) *An Audio Front End for Query-by-Humming Systems.* Ismir2001. Diakses November 25, 2009 dari http://ismir2001.ismir/netpdfhaus.pdf

[16] 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

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

[18] Meek, C. and W. P. Birmingham (2002). *Johnny Can't Sing: A Comprehensive Error Model for Sung Music Queries.* ISMIR 2002, Paris, France

LAMPIRAN

Script simulasi yang dimodifikasi dari toolbox

Training Data

```
function varargout = training_data10s(varargin)
% TRAINING_DATA10S M-file for training_data10s.fig
%      TRAINING_DATA10S, by itself, creates a new TRAINING_DATA10S or raises
the existing
%      singleton*.
%
%      H = TRAINING_DATA10S returns the handle to a new TRAINING_DATA10S or
the handle to
%      the existing singleton*.
%
%      TRAINING_DATA10S('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in TRAINING_DATA10S.M with the given input
arguments.
%
%      TRAINING_DATA10S('Property','Value',...) creates a new
TRAINING_DATA10S or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before training_data10s_OpeningFunction gets
called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to training_data10s_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_data10s

% 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_data10s_OpeningFcn, ...
                   'gui_OutputFcn',  @training_data10s_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_data10s is made visible.
function training_data10s_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_data10s (see VARARGIN)

% Choose default command line output for training_data10s
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes training_data10s wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = training_data10s_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

warning('off','MATLAB:dispatcher:InexactMatch');
tic
% load the song data
data=get(handles.Data,'string');
load(data);
% number of classes
nc=size(songDb,2); %menghitung jumlah lagu

% number of repetitions
nr=1;
% number of groups
ng=2;

% Matriz for the parameters
vl=cell(nc,ng);
for ic=1:nc,
   vlc{ic,1}=cell(nr,1);
   vlp{ic,1}=cell(nr,1);
   for ig=1:ng
      vl{ic,ig}=cell(nr,1);
   end
end
% generation nr repetitions
for in=1:nc
    for ir=1:nr,
    pv=songDb(in).pv;
% change pv to note
    timeUnit=256/8000;
    pitchTh=0.8;
    minNoteDuration=0.1;
    plotOpt=0;
    note = pv2note(pv, timeUnit, pitchTh, minNoteDuration, plotOpt);
 % get duration from note
    durasi=note(2:2:end);
 % counting how many coloum is the 10s part of the song
    lama=9;
    Ncdurasi=detik(lama,durasi);
 % get pitch from note
    pitch=note(1:2:end);
    pitch=pitch(1:Ncdurasi);
 % convert pitch to contour
    contour=pitch2contour11(pitch);
 % make contour as parameter 1
    vl{in,1}{ir}=[contour(1:end-1)'];
 % make duration as parameter 2
    Rdurasi=logRDD(durasi(1:Ncdurasi));
    vl{in,2}{ir}=[Rdurasi(1:end-1)'];
    end
end
```

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

 save vtrain vl;
 clear vtrain vl;

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);
```

QBSH System

```
function varargout = QBH_Sys10s(varargin)
% QBH_SYS10S M-file for QBH_Sys10s.fig
%      QBH_SYS10S, by itself, creates a new QBH_SYS10S or raises the existing
%      singleton*.
%
%      H = QBH_SYS10S returns the handle to a new QBH_SYS10S or the handle to
%      the existing singleton*.
%
%      QBH_SYS10S('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in QBH_SYS10S.M with the given input
arguments.
%
%      QBH_SYS10S('Property','Value',...) creates a new QBH_SYS10S or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before QBH_Sys10s_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to QBH_Sys10s_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_Sys10s

% 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_Sys10s_OpeningFcn, ...
                   'gui_OutputFcn',  @QBH_Sys10s_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_Sys10s is made visible.
function QBH_Sys10s_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_Sys10s (see VARARGIN)

% Choose default command line output for QBH_Sys10s
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes QBH_Sys10s wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = QBH_Sys10s_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

warning('off','MATLAB:dispatcher:InexactMatch');

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

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

% number of classes
nc=size(songDb,2);

% 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('Recording...');
        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
    tic
    [y, fs, nbits]=wavread(waveFile);
    PP=ptParamSet(fs, nbits);
```

```
    plotOpt=0;
    [pitch, clarity]=wave2pitchByAmdf(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);
 % convert pitch to contour
    contourTest=pitch2contourm(pitchTest);
    vtest{1,1}{1}=[contourTest(1:end-1)'];
  %get duration from note
    durasiTest=note(2:2:end);
    Rdurasi=dur(durasiTest);
    vtest{1,2}{1}=[Rdurasi(1:end-1)'];
    vl=vtest;

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

   logPO=probabilitas(file_hmm);

   %sorting the song based on logPO
   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
```

REPRESENTASI PITCH

3 level

```
function contour=pitch2contour3(pitch)
% mengubah pitch menjadi 3 level contour
pitch1=pitch;
pitch2=circshift(pitch,[0,1]);
contour=pitch1-pitch2;

% Proses kuantisasi
% negative
neg=find(contour<-0.45);
contour(neg)=-1;
% positive
pos=find(contour>0.45);
contour(pos)=1;
% nilai nol
nol=find(contour>=-0.45 & contour<=0.45);
contour(nol)=0;
contour(:,1)=[];
```

5 level

```
function contour=pitch2contour5(pitch)

% convert pitch to 5 level contour
pitch1=pitch;
pitch2=circshift(pitch,[0,1]);
contour=pitch1-pitch2;

% Proses kuantisasi
% negative
neg=find(contour>=-1.45 & contour<=-0.45);
contour(neg)=-1;
neg2=find(contour<-1.45);
contour(neg2)=-2;
% positive
pos=find(contour>=0.45 & contour<=1.45);
contour(pos)=1;
pos2=find(contour>1.45);
contour(pos2)=2;
nol=find(contour>=-0.45 & contour<=0.45);
contour(nol)=0;
contour(:,1)=[];
```

7 level

```
function contour=pitch2contour7(pitch)

% convert pitch to 7 level contour
pitch1=pitch;
pitch2=circshift(pitch,[0,1]);
contour=pitch1-pitch2;

% Proses kuantisasi
% negative
neg=find(contour>=-1.45 & contour<-0.45);
contour(neg)=-1;
neg2=find(contour>=-2.45 & contour<-1.45);
contour(neg2)=-2;
neg3=find(contour<-3.45);
contour(neg3)=-3;
% positive
pos=find(contour>0.45 & contour<=1.45);
contour(pos)=1;
pos2=find(contour>1.45 & contour<=2.45);
contour(pos2)=2;
pos3=find(contour>2.45);
contour(pos3)=3;
nol=find(contour>=-0.45 & contour<=0.45);
contour(nol)=0;
contour(:,1)=[];
```

9 level

```
function contour=pitch2contour9(pitch)

% convert pitch to 9 level contour
```

```
pitch1=pitch;
pitch2=circshift(pitch,[0,1]);
contour=pitch1-pitch2;

% Proses kuantisasi
% negative
neg=find(contour>=-1.45 & contour<-0.45);
contour(neg)=-1;
neg2=find(contour>=-2.45 & contour<-1.45);
contour(neg2)=-2;
neg3=find(contour>=-3.45 & contour<-2.45);
contour(neg3)=-3;
neg4=find(contour<-3.45);
contour(neg4)=-4;

% positive
pos=find(contour>0.45 & contour<=1.45);
contour(pos)=1;
pos2=find(contour>=1.45 & contour<=2.45);
contour(pos2)=2;
pos3=find(contour>2.45 & contour<=3.45);
contour(pos3)=3;
pos4=find(contour>3.45);
contour(pos4)=4;
nol=find(contour>=-0.45 & contour<=0.45);
contour(nol)=0;
contour(:,1)=[];
```

11 level

```
function contour=pitch2contour11(pitch)

% convert pitch to 11 level contour
pitch1=pitch;
pitch2=circshift(pitch,[0,1]);
contour=pitch1-pitch2;

% Proses kuantisasi
% negative
neg=find(contour>=-1.45 & contour<-0.45);
contour(neg)=-1;
neg2=find(contour>=-2.45 & contour<-1.45);
contour(neg2)=-2;
neg3=find(contour>=-3.45 & contour<-2.45);
contour(neg3)=-3;
neg4=find(contour>=-4.45 & contour<-3.45);
contour(neg4)=-4;
neg5=find(contour<-4.45);
contour(neg5)=-5;

% positive
pos=find(contour>0.45 & contour<=1.45);
contour(pos)=1;
pos2=find(contour>=1.45 & contour<=2.45);
contour(pos2)=2;
```

```
pos3=find(contour>2.45 & contour<=3.45);
contour(pos3)=3;
pos4=find(contour>3.45 & contour<=4.45);
contour(pos4)=4;
pos5=find(contour>4.45);
contour(pos5)=5;
nol=find(contour>=-0.45 & contour<=0.45);
contour(nol)=0;
contour(:,1)=[];
```

PI

```
function contour=pitch2contourm(pitch)
pitch1=pitch;
pitch2=circshift(pitch,[0,1]);
contour=pitch1-pitch2;
contour(:,1)=[];
```

REPRESENTASI DURASI

logDurasi

```
function Rduration=dur(durasi);
[nr,nc]=size(durasi);

for i=1:nc,
 Rduration(i)=-10*log10(durasi(i));
end
Rduration=round(Rduration);
return
```

RDD

```
function Rduration=RDD(durasi);
%this function convert duration to relative duration difference
[nr,nc]=size(durasi);
durasi1=durasi;
durasi2=circshift(durasi,[0,1]);
Rduration=zeros(nr,nc);
for i=1:nc,
    Rduration(i)=durasi1(i)/durasi2(i);
end
Rduration=round(Rduration);
Rduration(:,1)=[];
return
```

logRDD

```
function Rduration=logRDD(durasi);
%this function convert duration to 10 level log of relative duration
difference
[nr,nc]=size(durasi);
durasi1=durasi;
```

```
durasi2=circshift(durasi,[0,1]);
Rduration=zeros(nr,nc);
for i=1:nc,
    Rduration(i)=10*log10(durasi1(i)/durasi2(i));
end
Rduration=round(Rduration);
Rduration(:,1)=[];
return
```

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);
ng=size(vl,2);

% 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}=2.*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;
%Ngauss{1}=[3];
%Ngauss{2}=[3];
```

```
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;
    Ne(ic,2)=s;
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
```

CHMM_TRAIN

```
% ------------------------------------------------------------------------
--------------------------
% This program calls the diffrent functions to design the HMMs (one for each
class and for each group)
% designed in function CHMM_DEF.
%
% function chmm(fhmm,fptrain,fptest,fhmmout)
%
% Entry:  fhmm is the name of the HMMs set up in the function CHMM_DEF
%
%         fptrain is the name of the file containing the sequences of
parameters to train each HMM
%         with its own group of training set. In each repetition, we find the
a sequence of parameters
%         for a class and a group.
%
%         fptest is the name of the file containing the sequences of
parameters to test each HMM
%         with its own group of test set. In each repetition, we find the a
sequence of parameters
%         for a class and a group.
%
%         fsalhmm: Name of the file containing the outputs of each classifier
for each sample of the database of the test
%         in a cell array salhmm{class, group}(repetition).
%
%       Variables defined in the fhmm with the function CHMM_DEF
%
%     nc: number of classes.
%     ng: number of groups.
%     agrup{ng}: the way to gather the parameters together.
%     Np(ng): number of parameters by group.
%       The variables of the HMM:
%               cell array A=cell(nc,ng);
%               cell array B=cell(nc,ng);
%               cell array Med=cell(nc,ng);
%               cell array Var=cell(nc,ng);
%               cell array Pi=cell(nc,ng);
%       Ne(nc,ng): Number of states of the HMM for each class and group.
%       Maxitermi is the maximum iteration for the initial model
```

```
%       umbral: Umbral is the threshold condition to stop the HMM (the error
is calculated with the maximum likelihood criterion)
%       maxiter: Maxiter is the maximum iteration number in the Bakis HMM
(condition to stop the algorithm)
%       salto: Salto is the maximum path authorized in the Bakis HMM from a
state to another.
%       BAKIS: if Bakis = 1 implements a Bakis HMM (also called left-right),
else an ergodic HMM is defined
%
%           Salhmm is a matrix used to store the probabilities values
%
% NOTA: use the scripts chmm_def, and chmm_men.
% NOTA: use the functions: iniciacHMM, genchmm, alfabetac,  probsecC,
viterbic, baumc
% NOTA: if fptrain='' we only realize the test
% NOTA: if fptest='' we only make the training
%-------------------------------------------------------------------------
-------------------------

% the test function had been seperated from this function by ewaldo zihan

function train(fhmm,fptrain)

% Name of the files ot the training set
%fptrain='vtrain.mat';
% Name of the file of the test set
%fptest='vtest.mat';

% We read the file of set up for the HMMs defined in the DHMM_DEF

eval(['load ',fhmm]);
if eq(nargin,4),fhmm=fhmmout;end
guardar=['save ',fhmm,vDB,vHMM,vTEST,' iniciar vDB vHMM vTEST guardar'];

% random variables
randn('state',sum(100*clock));
rand('state',sum(100*clock));

% We check that we have to make or not the training (of the length is 0, we
don't train).
if ne(length(fptrain),0)
    % Here we make the training:

    % We change the format of the file to make the training
    [lrep]=formato_lectura_secuencial(fptrain,'vtrain');

   % We calculate the classifiers
   for ig=1:ng
         for ic=1:nc,
            % Messages for the design of the classifier
            if men
              chmm_men
          end

nftrain=['c:\temphmm\vtrain_c',num2str(ic),'_g',num2str(ig),'.tmp'];
            % we calculate the model initial of the HMM for each class and
group
```

```
            if iniciar
                hora=clock;
                fprintf('Calcul of the initial model. Hour:
%g:%g\n',hora(4),hora(5))

[A{ic,ig},B{ic,ig},Med{ic,ig},Var{ic,ig},Pi{ic,ig}]=iniciachmm(Ne(ic,ig),Np(i
g),BAKIS,salto,nftrain,lrep{ic,ig},agrup{ig},Ngauss{ig},maxitermi);
            % We start to train the HMM in order to initialize the model
            % with the Baum-Welch (EM) method.
            % The training is terminated when the distance between two models
calculated is under the threshold
            % or when we have made the maximum iteration authorized.
                hora=clock;
                fprintf('End of the calcul of the initial model.  Hour:
%g:%g\n',hora(4),hora(5))
            else
                hora=clock;
                fprintf('Start of Baum Welch. Hour: %g:%g\n',hora(4),hora(5))
            end

            iter=0;
            cfe=umbral+1;logPOs=zeros(size(lrep{ic,ig},1),1);
            while and(abs(cfe)>umbral,iter<maxiter),

               iter=iter+1;
               logPOa=logPOs;
               % Baum-Welch algorithm.

[A{ic,ig},B{ic,ig},Med{ic,ig},Var{ic,ig},Pi{ic,ig},logPOs]=baumc(A{ic,ig},B{i
c,ig},Med{ic,ig},Var{ic,ig},Pi{ic,ig},nftrain,lrep{ic,ig},agrup{ig});
               % Criterion to stop.
               if eq(iter,1),
                   logPObase=mean(logPOs);
                   hora=clock;
                   fprintf('iter. Baum: %g,\tProb.: %g\tStd: %g\tHour:
%g:%g\n',iter,mean(logPOs),std(logPOs),hora(4),hora(5));
               else;
                   cfe=1-(mean(logPOa-logPObase)/mean(logPOs-logPObase));
                   hora=clock;
                   fprintf('iter. Baum: %g,\tProb.: %g\tStd: %g\tCfe:
%g\tHour: %g:%g\n',iter,mean(logPOs),std(logPOs),cfe,hora(4),hora(5));
               end
            end;
            fprintf('End of the HMM, group %g, class %g.\n',ig,ic);
          % We save a file with all the variables of the HMM
            eval([guardar]);
         end;
      fprintf('End of the training.\n')
   end
end
% We save in a file all the varaiables
eval([guardar]);

return

% originaly this function lead to the test part.
% but i seperate the test part  for time efficiency.
```

CHMM_TEST

```
% CHMM test part
% modified from Grupo De Procesado Digital De Senales(GPDS)
% Universidad de Las Palmas de Gran Canaria
% CHMM function
% by Ewaldo Zihan
function chmm(fhmm,fptest)

eval(['load ',fhmm]);

% We check if we have to make or no the test
if ne(length(fptest),0)
    % If the length>0 we make it:
    % We convert the test set into the adequat format for the HMM
    [lrep]=formato_lectura_secuencial(fptest,'vtest');

    % Messages :
    if men,
        fprintf('Input parameters of the HMM: %s\n',fptest);
        fprintf('Number of the class: %g\n',nc);
        fprintf('Number of the group: %g\n',ng);
    end;
    for ig=1:ng,
        for ic=1,
            hora=clock;
            fprintf('Calcul of the output of the group:: %g, class: %g.
Number of the repetition: %g. Hour:
%g:%g\n',ig,ic,size(lrep{ic,ig},1),hora(4),hora(5))
            % We create a file to store the output (results) of the HMM:

fidsalhmm=fopen(['c:\temphmm\salhmm_c',num2str(ic),'_g',num2str(ig),'.tmp'],'
w');
          % We open the file containing the sequence of parameters for the
test

fidvtest=fopen(['c:\temphmm\vtest_c',num2str(ic),'_g',num2str(ig),'.tmp'],'r'
);
            % we make the test
            dimrep=agrup{ig}(end)-agrup{ig}(1);
            nrep=size(lrep{ic,ig},1);

            for ir=1:nrep,
                %hora=clock;
                %fprintf('\tCalcul of the output of the HMM: %g, class: %g.
Repetition: %g. Hour: %g:%g\n',ig,ic,ir,hora(4),hora(5))
                % We read the repetition
                vlt{1}=fread(fidvtest,lrep{ic,ig}(ir)*dimrep,'double');
                vlt{1}=reshape(vlt{1},lrep{ic,ig}(ir),dimrep);
                % We evaluate the probability for each model
                salida=zeros(nc,1);
                for ihmm=1:nc;

salida(ihmm,1)=probsecc(A{ihmm,ig},B{ihmm,ig},Med{ihmm,ig},Var{ihmm,ig},Pi{ih
mm,ig},vlt{1},agrup{ig});
                end
```

```
                fwrite(fidsalhmm,salida,'double');
            end
            fclose(fidsalhmm);
            fclose(fidvtest);
        end;
    end;

    % Join the outputs of the HMM salhmm to keep all together
    fprintf('Union of the outputs salhmmm.\n');
    for ig=1:ng
        for ic=1:nc

fid=fopen(['c:\temphmm\salhmm_c',num2str(ic),'_g',num2str(ig),'.tmp'],'r');
            nrep=size(lrep{ic,ig},1);
            salhmm{ic,ig}=cell(nrep,1);
            for ir=1:nrep
                salhmm{ic,ig}{ir}=fread(fid,nc,'double');
            end
            fclose(fid);
        end
    end


   fprintf('FIN test.\n');
    % We save the outputs of the HMM.
    eval([guardar]);
end
return
```

Probabilitas

```
% This function load the log of probability from the hmm
% and find the total log of probability from the parameters
% modified from resulhmm function
% by Ewaldo Zihan

function logPO=probabilitas(fhmm)
eval(['load ',fhmm])

logPO=zeros(nc,ng);
fprintf('Number of class: %g\n',nc);
fprintf('Number of groups: %g\n',ng);

ig=1;
for ic=1:nc
   % it makes the test for this realisation:
   for ir=1:size(salhmm{ic,ig},1)
      for ig=1:ng
         % each realisation is made of a group of vectors
         logPO(:,ig)=salhmm{ic,ig}{ir};
      end
   end
end
```

```
% sum the logPO from parameter 1 and parameter 2
logPO=logPO(:,1)+logPO(:,2);
```

DETIK

```
function durasi10s=detik(lama,durasi)
% counting the number of column in lama second part of the song
[nb,nc]=size(durasi);
durasi10s=1;
jumlah=0;
if durasi(1,durasi10s)<=10,
    while jumlah<lama && durasi10s<=nc,
        jumlah=jumlah+durasi(1,durasi10s);
        durasi10s=durasi10s+1;
    end
    durasi10s=durasi10s-1;
else
durasi10s=1;
end
```