UNIVERSITAS INDONESIA

SIMULASI QUERY BY SINGING/HUMMING UNTUK MUSIK DANGDUT DENGAN MENGGUNAKAN METODE DYNAMIC TIME WARPING

SKRIPSI

TEDDY FEBRIANTO

0606029486

FAKULTAS TEKNIK

DEPARTEMEN TEKNIK ELEKTRO

DEPOK

JUNI 2010

UNIVERSITAS INDONESIA

SIMULASI QUERY BY SINGING/HUMMING UNTUK MUSIK DANGDUT DENGAN MENGGUNAKAN METODE DYNAMIC TIME WARPING

SKRIPSI

Diajukan sebagai salah satu syarat memperoleh gelar sarjana

TEDDY FEBRIANTO

0606029486

FAKULTAS TEKNIK

DEPARTEMEN 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 : Teddy Febrianto
NPM : 0606029486
Tanda Tangan :
Tanggal : 14 Juni 2010

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Teddy Febrianto
NPM	: 0606029486
Program Studi	: Teknik Elektro
Judul Skripsi	: Simulasi Query by Singing/Humming untuk Musik Dangdut Dengan Menggunakan Metode Dynamic Time Warping

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

Puji syukur saya panjatkan kehadirat Tuhan YME, karena atas segala rahmat dan penyertaan-Nya saya dapat menyelesaikan skripsi ini. Saya menyadari bahwa skripsi ini tidak akan terselesaikan tanpa bantuan dari berbagai pihak. Oleh karena itu, saya mengucapkan terima kasih kepada :

1. Bapak Prof Dr. Ir. Dadang Gunawan, M.Eng, selaku pembimbing yang membantu memberikan arahan dan nasihat sehingga saya dapat menyelesaikan skripsi ini;
2. Bapak Indrabayu, ST, MT, M.Bus, Sys, selaku pembimbing materi yang banyak sekali membantu dan memotivasi saya dalam mengerjakan skripsi ini;
3. Para peneliti sebelum ini yang juga memberikan sumber bacaan yang banyak bagi saya;
4. Teman – teman satu bimbingan dan satu angkatan dengan saya, Abdul Aziz Muslim dan Ewaldo Zihan, yang telah banyak bertukar pikiran dan bekerjasama dalam menyelesaikan skripsi ini;
5. Papa, Mama, kakak dan adik saya yang selalu memberikan dukungan kepada saya;
6. Anne Widiastri, Ardyan Indra, Indah Rianti, Achmad Ari Syahbani, Cindy Chairunissa, dan Indra Gumilang yang bersedia menjadi sampel dalam skripsi ini;
7. Teman-teman asisten Laboratorium Telekomunikasi, teman- teman Elektro 2006 dan seluruh Sivitas akademik Departemen Teknik Elektro yang tidak dapat saya sebutkan satu persatu.

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

Depok, Juni 2010

Teddy Febrianto

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Teddy Febrianto
NPM : 0606029486
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 :

SIMULASI QUERY BY SINGING/HUMMING UNTUK MUSIK DANGDUT DENGAN MENGGUNAKAN METODE DYNAMIC TIME WARPING

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 : 14 Juni 2010
Yang menyatakan

Teddy Febrianto

ABSTRAK

Nama : Teddy Febrianto
Program studi : Teknik Elektro
Judul : Simulasi Query by Singing/Humming untuk Musik Dangdut Dengan Menggunakan Metode Dynamic Time Warping

Query by Singing/Humming (QbSH) adalah sistem pencarian lagu berdasarkan pada senandung sebagai dasar pencariannya. Skripsi ini akan membahas mengenai perancangan sistem QbSH menggunakan *Dynamic Time Warping* (DTW) sebagai metode pencocokan melodi. DTW sendiri memiliki beberapa spesifikasi dan bersifat data *dependent*. Oleh karena itu, pada kesempatan ini akan dirancang suatu simulasi QbSH untuk menentukan spesifikasi DTW yang tepat untuk musik dangdut dan keroncong. Pengukuran performansi dilakukan berdasarkan parameter MRR (*Mean Reciprocal Rank*). Berdasarkan simulasi terhadap tiga tipe DTW diperoleh hasil tipe DTW 1 memberikan hasil MRR yang paling tinggi. Selain itu untuk spesifikasi *free end point* didapatkan nilai *gulley* yang memberikan nilai MRR tertinggi adalah antara 0,15 -0,4. Nilai MRR tertinggi yang didapatkan adalah sebesar 0,67.

Kata kunci : QbSH, Melodi, DTW.

ABSTRACT

Name : Teddy Febrianto
Study program: Electrical Engineering
Title : Simulation of Query by Singing/Humming for Dangdut Music Using Dynamic Time Warping

Query by Singing/Humming (QbSH) is a music search engine which is based on hum input as the basic of searching. This thesis will talk about QbSH system planning using Dynamic Time Warping (DTW) as the matching engine between the hum input and the database. DTW itself has coupler specification and is compatible for dependent data. In this opportunity, QbSH simulation will be build to determine suitable DTW type for dangdut music genre. The performance will be measured based on MRR (Mean Reciprocal Rank). According to the simulation of three DTW types, the result show that DTW type 1 gives the highest MRR. In addition to free end point specification, gulley values are about 0.15 to 0.4 will give the highest MRR. The highest value of MRR is 0.67.

Key words : QbSH, melody, DTW.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PERNYATAAN ORISINALITAS ii
HALAMAN PENGESAHAN iii
UCAPAN TERIMA KASIH iv
LEMBAR PERSETUJUAN PUBLIKASI KARYA ILMIAH v
ABSTRAK vi
ABTRACT vii
DAFTAR ISI viii
DAFTAR TABEL xi
DAFTAR GAMBAR xii
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Perumusan Masalah 4
1.3 Tujuan 4
1.4 Batasan Masalah 4
1.5 Metode Penelitian 4
1.6 Sistematika Penulisan 5
BAB 2 MUSIK, QUERY BY SINGING/HUMMING, DAN DYNAMIC TIME WARPING 6
2.1 Musik 6
2.1.1 Beberapa Istilah dalam Musik 6
2.1.2 Representasi Musik 8
2.1.2.1 Format *Score* 8
2.1.2.2 Format Audio/*Waveform* 9
2.1.2.3 Format *Hybrid* 11
2.1.3 Musik Dangdut 11
2.2 *Query By Singing/Humming* 12
2.2.1 Music Information Retrieval 12
2.2.2 Pengertian umum *Query by Singing/Humming* (QbSH) 12
2.2.3 Prinsip Dasar *Query by Singing/Humming* 13

2.2.3.1 Pembuatan *Database* 15
2.2.3.2 Pemrosesan Senandung Sebagai *Query* 15
2.2.3.3 Perbandingan Masukan *Query* dengan *Database* 17
2.2.3.4 Penampilan Hasil *Query* 18
2.3 Dynamic Time Warping 18
2.3.1 Konsep DTW 18
2.3.2 Variasi DTW 20
2.3.2.1 *Step Size* dan *Local Weight* 20
2.3.2.2 *Boundary Conditions* 21
2.3.2.3 Perhitungan Jarak 22
BAB 3 PERANCANGAN SIMULASI 23
3.1 Diagram Perancangan Simulasi QbSH 23
3.2 Pengolahan *Query* 24
3.2.1 Pembentukan *Frame* (*Framing*) 25
3.2.2 Metode *Pitch Tracking* 26
3.2.3 Metode *Note Segmentation* 27
3.2.4 *Melody Representation* untuk Query 28
3.3 Pembentukan *Database* 29
3.4 DTW yang Digunakan 30
3.4.1 *Step size* dan *Local Weight* yang Digunakan 30
3.4.2 Perhitungan Jarak 31
3.4.3 *Free End* 31
3.5 Simulasi 32
BAB 4 HASIL SIMULASI DAN ANALISIS 34
4.1 Data Hasil Simulasi dan Pengolahan Data 34
4.1.1 Hasil Uji Coba Tipe DTW 1 dengan *Manhattan Distance* 34
4.1.2 Hasil Uji Coba Tipe DTW 2 dengan *Manhattan Distance* 36
4.1.3 Hasil Uji Coba Tipe DTW 3 dengan *Manhattan Distance* 37
4.2 Analisis 38
4.2.1 Analisis Tipe DTW 38
4.2.2 Analisis *Gulley* 44
4.2.3 Analisis MRR Tertinggi 45

4.2.4 Analisis Pengaruh Jenis Kelamin Sampel Terhadap MRR 45
BAB 5 KESIMPULAN..... 50
DAFTAR REFERENSI 51
LAMPIRAN 53

DAFTAR TABEL

Tabel 4.1 Hasil Uji Coba Tipe DTW 1 dengan *Manhattan Distance* 35
Tabel 4.2 Hasil Uji Coba Tipe DTW 2 dengan *Manhattan Distance* 36
Tabel 4.3 Hasil Uji Coba Tipe DTW 3 dengan *Manhattan Distance* 37
Tabel 4.4 Perbandingan Jarak Tipe 1 dan 3 untuk lagu “Kopi Dangdut”..... 43
Tabel 4.5 Pengaruh Kenaikan *Theshold* Durasi terhadap MRR 47
Tabel 4.6 Perbandingan Total MRR setelah Optimalisasi *Threshold* Durasi 48

DAFTAR GAMBAR

Gambar 2.1 Contoh *Score* dari Beethoven's Fifth 9
Gambar 2.2 *Waveform* dengan Frekuensi 4Hz..... 9
Gambar 2.3 Detik ke 7.3 sampai 7.8 dalam Beethoven's fifth Bernstein..... 10
Gambar 2.4 Diagram Blok dari QbSH 14
Gambar 2.5 Contoh *Cost Matrix* 19
Gambar 2.6 *Path* Minimum yang Dipilih 20
Gambar 2.7 Contoh *Step Size* dan *Local Weight*..... 20
Gambar 2.8 *Boundary Conditions*..... 22
Gambar 3.1 Diagram Sistem QbSH yang Dirancang 23
Gambar 3.2 Diagram Alir Pengolahan *Query*..... 25
Gambar 3.3 Ukuran *Frame* dan *Overlap* 26
Gambar 3.4 Gambar Pencarian Periode Dasar dengan Menggunakan ACF 27
Gambar 3.5 Proses *Note Segmentation*..... 28
Gambar 3.6 *Relative Pitch* 29
Gambar 3.7 Program untuk Membuat *Database*..... 29
Gambar 3.8 Tipe *Step size* dan *Local Weight* DTW..... 30
Gambar 3.9 *Gulley*..... 31
Gambar 3.10 Tampilan Program Simulasi..... 32
Gambar 4.1 Grafik Uji Coba Tipe DTW 1 dengan *Manhattan Distance*..... 35
Gambar 4.2 Grafik Uji Coba Tipe DTW 2 dengan *Manhattan Distance*..... 36
Gambar 4.3 Grafik Uji Coba Tipe DTW 3 dengan *Manhattan Distance*..... 37
Gambar 4.4 Grafik MRR dari Tipe-Tipe DTW..... 38
Gambar 4.5 *Cost matrix* untuk Lagu "Penasaran" 41
Gambar 4.6 Grafik Pengaruh Kenaikan *Threshold* Durasi terhadap MRR 47
Gambar 4.7 Grafik MRR Sebelum dan Sesudah Optimalisasi 48

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan dunia informasi pada masa sekarang ini semakin berkembang pesat. Hal ini terlihat dari kecepatan penyampaian informasi yang semakin meningkat seiring dengan kemudahan dalam mengakses informasi. Metode pencarian informasi yang semakin baik pun terus dikembangkan untuk jenis informasi tertentu. Salah satu pencarian informasi yang sedang dikembangkan saat ini adalah aplikasi pada pencarian lagu.

Pada sistem pencarian lagu yang ada sekarang, metode pencariannya rata-rata masih berupa *category-based browsing* atau *text-base searching* yang berdasarkan metadata [1]. Metadata merupakan informasi tambahan yang disisipkan pada lagu biasanya berupa judul lagu, nama penyanyi, nama album, dll. Oleh karena itu, untuk mencari lagu harus diketahui judul lagu atau informasi-informasi tambahan lainnya berupa nama penyanyi, nama album, dan lain-lain. Metode ini tidak efektif karena terkadang seseorang mengingat lagu tertentu tetapi tidak mengetahui judul atau informasi tambahannya, melainkan hanya mengingat sepenggal melodi dari lagu tersebut. Dengan demikian diperlukan suatu metode sistem pencarian yang bekerja berdasarkan konten dari lagu itu sendiri bukan berdasarkan metadata dimana sistem ini biasa disebut dengan sistem *query by content*, atau yang terlebih dahulu dikenal dengan nama *content-based music retrieval* [2].

Sistem pencari yang dibahas dalam skripsi ini adalah sistem *Query by Singing/Humming* (QbSH). Dengan sistem QbSH ini, lagu dapat dicari hanya dengan menyenandungkan melodi dari lagu tersebut. Sebenarnya masih ada sistem pencari lagu lain yang bekerja berdasarkan konten dari lagu seperti: *query by whistling* dan *query by tapping*, namun metode pencarian menggunakan senandung lebih aplikatif. Bila dibandingkan dengan *query by tapping sistem*, QbSH lebih unggul karena kebanyakan orang lebih mudah mengingat melodi suatu lagu dibandingkan dengan ritmenya ketika mendengarkan suatu lagu.

Sedangkan bila dibandingkan dengan sistem lain yang sama-sama menggunakan melodi sebagai dasar pencarian seperti *query by whistling* sistem QbSH juga lebih unggul karena tidak semua orang bisa bersiul.

QbSH terdiri dari beberapa bagian (subsistem) yang membangun secara keseluruhan sistem ini [3]. Salah satu bagian yang menjadi perhatian adalah pada bagian *melody maching* atau *matching engine*. Bagian ini merupakan inti dari sistem QbSH, karena pada bagian ini akan dilakukan pencarian lagu berdasarkan masukan *query* yang diberikan. Bagian ini juga merupakan bagian dari sistem yang paling banyak mengkonsumsi waktu.

Pada tahun 2001, Wei Chai menghitung kemiripan melodi dari masukan dengan *database* berdasarkan *euclidean distance, cosine similarity, longest common sub-sequence* (LCS), *Longest common consecutive sub-sequence* (LCCS) dan *modified LCS* [1]. Keempat metode tersebut dibandingkan dan didapatkan LCS, LCCS, *modified* LCS memberikan hasil yang lebih baik untuk sistem QbSH yang dibuat. Metode LCS dan LCCS tersebut menggunakan prinsip *dynamic programming*. Oleh karena itu, penelitian QbSH selanjutnya lebih difokuskan pada metode *dynamic programming*. Wei Chai juga menggunakan konsep *dynamic programming* sebagai *melody matching*[2].

Kemudian Jean Louis menggunakan metode *melody matching* yang dikenal sebagai *Dynamic Time Warping* (DTW)[3]. DTW sendiri juga menggunakan konsep *dynamic programming*. Konsep DTW sendiri sudah pernah diperkenalkan oleh Roger Jang[4].

Selanjutnya beberapa metode mulai diujicobakan pada sistem QbSH. Metode-metode yang digunakan tersebut antara lain, *Hidden Markov Models* (HMM), DTW (dengan menggunakan *melodic contour* dan *note interval* sebagai *feature*), N-gram, dan *CubyHum*[5]. Hasilnya DTW (dengan *melodic contour*) dan HMM memberikan hasil yang cukup baik.

DTW sendiri memiliki beberapa tipe dan dapat dimodifikasi algoritmanya. Roger Jang memberikan dua tipe DTW yang dapat digunakan pada QbSH [4]. Berbagai algoritma DTW terus dikembangkan, hal ini dikarenakan performa dari

DTW berbagai tipe tergantung pada *database* yang digunakan[4]. Jadi tidak ada jaminan bahwa salah satu tipe selalu menjadi yang terbaik untuk semua jenis *database* yang digunakan.

Perkembangan QbSH sendiri di Indonesia masih cenderung kurang. Sistem QbSH mulai dibuat dengan menggunakan *database* dari lagu-lagu Indonesia, termasuk di dalamnya lagu keroncong[6]. Metode *matching* yang digunakannya adalah menggunakan metode ANN (*Artificial Neural Network*). Hasil yang didapatkan cukup baik, namun sayangnya penelitian ini masih dibatasi pada jumlah *database* yang tidak terlalu besar bila dibandingkan dengan penelitian yang dilakukan di luar negeri.

Sistem QbSH yang dicobakan pada *database* lagu dangdut tentunya akan menjadi hal yang sangat menarik. Pemilihan metode akan sangat berpengaruh terhadap hasil yang didapatkan. Hal ini disebabkan karena pemilihan metode harus disesuaikan dengan *database* yang digunakan.

Lagu dangdut merupakan salah satu warisan dari budaya bangsa Indonesia. Namun sayangnya lagu-lagu ini mulai ditinggalkan oleh masyarakat Indonesia sendiri. Hal ini dikarenakan pengaruh perkembangan zaman yang membuat masyarakat lebih menyukai *genre* pop, *rock*, *jazz*, dan lain-lain. Oleh karena itu, dengan membuat sistem QbSH dengan menggunakan *database* musik dangdut, harapannya dapat ikut serta dalam melestarikan budaya bangsa.

Berdasarkan latar belakang tersebut akan dirancang simulasi dengan menggunakan metode DTW [3][4]. Kemudian akan dicoba untuk menggunakan *database* dari musik dangdut. Sistem QbSH dengan menggunaan DTW selama ini hanya diujikan pada lagu-lagu dari luar negeri.

1.2 Perumusan Masalah

Sistem QbSH dengan metode *Dynamic Time Warping* (DTW) yang selama ini diujikan hanya berdasarkan pada *database* untuk lagu-lagu dari luar negeri. Metode DTW sendiri bersifat *database dependent*[4]. Selama ini belum penah ada

yang menggunakan metode DTW untuk *database* musik dangdut, sehingga tipe DTW yang sesuai untuk musik dangdut masih belum dapat ditentukan.

1.3 Tujuan

Tujuan dari pembuatan skripsi ini adalah merancang simulasi QbSH menggunakan metode DTW. Dengan merancang simulasi ini maka akan diketahui spesifikasi DTW, ditinjau dari tipe *step size* dan parameter *free end point (gulley)* yang paling tepat untuk *database* musik dangdut yang digunakan pada simulasi ini.

1.4 Batasan Masalah

Pada skripsi ini masalah dibatasi pada penggunaan metode DTW sebagai *matching engine* pada sistem QbSH. *Database* yang digunakan sebanyak 15 lagu dangdut dan untuk pengujian sistem digunakan 60 sampel suara yang berasal dari masing-masing 3 orang laki-laki dan wanita.

1.5 Metode Penelitian

Metode penelitian yang digunakan antara lain:

1. Studi Kepustakaan.
   Mempelajari dasar teori musik, konsep sistem QbSH, dan metode DTW dari berbagai buku, jurnal, dan artikel terkait lainnya.
2. Pengumpulan Data.
   Mengumpulkan data-data yang diperlukan, dalam hal ini adalah MIDI lagu dangdut yang digunakan sebagai *database* dan sampel suara yang direkam sendiri.
3. Simulasi Perangkat Lunak.
   Merancang simulasi dan melakukan pengujian QbSH dengan menggunakan spesifikasi DTW yang berbeda-beda dan membandingkannya.

1.6 Sistematika Penulisan

Pembahasan dalam skripsi ini dibagi menjadi lima bagian, yaitu:

BAB 1 PENDAHULUAN

Bab 1 berisi gambaran permasalahan secara umum yang diangkat dalam penelitian ini. Bab ini menjelaskan latar belakang masalah, perumusan masalah, tujuan, batasan masalah, metode penelitian dan sistematika penulisan penelitian ini.

BAB 2 MUSIK, *QUERY BY SINGING/HUMMING* DAN DYNAMIC TIME WARPING

Bab 2 berisi tentang tinjauan literatur, termasuk didalamnya pembahasan tentang musik, QbSH, dan DTW secara umum.

BAB 3 PERANCANGAN SIMULASI

Bab 3 berisi tentang metode DTW dan penerapannya pada QbSH. Selain itu juga akan dibahas tentang perancangan simulasi dari QbSH dengan menggunakan DTW.

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.

BAB 2
MUSIK, *QUERY BY SINGING/HUMMING* DAN DYNAMIC TIME WARPING

2.1 Musik

2.1.1 Beberapa Istilah dalam Musik

Musik dapat dianalogikan sebagai bahasa yang digunakan manusia untuk berkomunikasi. Dalam berkomunikasi, manusia menggunakan rangkaian kalimat untuk mengekspresikan perasaan dan keinginan dirinya. Demikian halnya dengan musik. Musik adalah salah satu cara manusia berinteraksi dengan manusia lainnya. Dengan musik, dapat disampaikan maksud dan tujuan dengan cara yang berbeda. Dalam bertutur kata manusia menggunakan rangkaian kalimat, dimana kalimat-kalimat tersebut tersusun atas kata-kata yang dibentuk oleh berbagai huruf abjad. Sedangkan musik dapat terbentuk dari rangkaian tangga nada (*scale*). Tiap-tiap nada dalam tangga nada mempunyai karakteristik tersendiri, diantaranya adalah:

a. *Pitch* adalah ukuran tinggi-rendah suatu nada yang dinyatakan dalam besaran frekuensi. Lebih tepatnya *pitch* adalah frekunsi dari gelombang sinusoidal yang ditangkap oleh pendengar dari sumber bunyi[6].

b. *Loudness* adalah intensitas keras lemahnya gelombang bunyi. Dalam analisis biasanya *loudness* dinyatakan dalam *mean square power* yang kemudian diubah dalam skala *logarithmic* / *decibel* [7]. Hal ini dilakukan untuk mempermudah analisis bila terjadi variasi level *power* yang luas.

c. Durasi adalah ukuran lama nada yang dimainkan dalam suatu satuan waktu. Persepsi seseorang terhadap frekuensi suatu nada (*pitch*) dapat berbeda tergantung pendengarnya tetapi persepsi setiap orang terhadap durasi suatu nada sama. Pada musik durasi berfungsi mengatur artikulasi. Contoh durasi yang ekstrim adalah *staccato* dan *legato*. Pada *staccato* nada dimainkan secara singkat sehingga jarak antara nada jelas.

Sedangkan pada *legato* nada dimainkan panjang-panjang sehingga jarak antara nada hampir tidak terasa.

d. *Timbre* adalah warna suara atau pembeda antara dua jenis sumber bunyi. Dengan *timbre* dapat dibedakan suara gitar atau suara piano walaupun memainkan nada yang sama. Hal ini disebabkan *timbre* gitar dan piano berbeda. Dalam representasi grafik gelombang suara, perbedaan *timbre* bisa dilihat dari bentuk amplitudo gelombang.

Dalam dunia musik hanya ada 12 nada dalam satu oktaf, yang 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 1/2 (setengah). Sebagai contoh, nada C menuju C# berjarak setengah, C menuju D berjarak satu, C menuju D# berjarak satu setengah dan seterusnya [6].

Susunan dari nada-nada yang membentuk sebuah makna disebut melodi. Melodi merupakan bagian dari musik yang paling mempengaruhi perasaan manusia. Jika melodi dari suatu lagu didengarkan, maka dapat langsung diketahui apakah lagu tersebut bersifat sedih atau bersemangat. Selain itu, melodi merupakan bagian yang paling berkesan dalam suatu lagu sehingga menjadi bagian utama dari suatu lagu.

Kombinasi dari dua nada atau lebih yang dimainkan secara bersamaan akan membentuk *Chord*. *Chord* berbeda dengan melodi, dimana melodi merupakan rangkaian dari nada yang dimainkan satu persatu. [7]

Tune adalah bentuk melodi yang paling sederhana, 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 sederhana. Monoponik hanya tersusun atas satu melodi saja. Bila satu melodi diiringi oleh suara lain *chord*

contohnya, maka disebut homoponik. Ketika ada dua atau lebih melodi yang dimainkan secara bersama-sama maka tekstur ini disebut poliponik.

Persepsi seseorang terhadap melodi dari suatu lagu berbeda-beda. Oleh karena itu, melodi sering disebut *perceptual feature* dari musik. Hal yang membuat suatu melodi lebih menarik dan mudah diingat adalah kombinasi dari *melodic contour* dan *rhytme*. *Melodic contour* merupakan perubahan dari nada dalam melodi. Perubahan ini biasanya direpresenrasikan dalam 3 level yang disimbolkan dengan ”+” (perubahannya naik), “–“ (perubahannya turun) dan “0” (tidak ada perubahan nada). Sedangkan *rhytme* adalah pola khusus dari musik yang berkaitan dengan waktu [1].

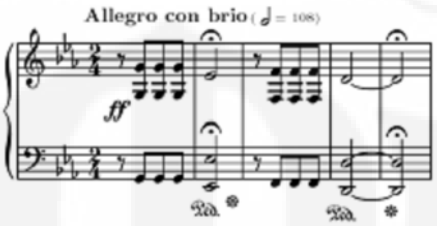
2.1.2 Representasi Musik

Perkembangan teknologi menyebabkan munculnya berbagai macam bentuk media penyimpanan musik. Dahulu musik disimpan dalam bentuk tertulis (*score*) sehingga bila ingin menikmati musik yang disimpan harus terlebih dahulu menerjemahkan *score* tersebut. Akan tetapi, saat ini musik direpresentasikan dalam format WAV (*Waveform Audio Format*), mp3, wma, MIDI dll, yang memungkinkan musik dapat dinikmati dengan mudah. Secara garis besar musik direpresentasikan dalam tiga bentuk yaitu: *score*, audio dan *hybrid format*.

2.1.2.1 Format *Score*

Format *score* ini merupakan bentuk representasi musik yang telah dikenal lama dalam musik klasik dengan sebutan “*piece of* musik”. Pada format *score* musik dikodekan kedalam bentuk grafis dengan tingkat frekuensi pada sumbu vertikal dan perubahan tehadap waktu pada sumbu horizontal [8]. Format *score* ini diibaratkan sebagai resep pada masakan atau dengan kata lain hanya berisi bagaimana memainkan musiknya bukan musik itu sendiri. Gambar 2.1 merupakan contoh dari format *score*. Pada format *score pitch* dan musik*al onset* (waktu awal mulainya suatu nada) direpresentasikan secara grafis. Tempo biasanya ditulis langsung seperti *allegro con brio* dan *andante con moto,* untuk tempo yang bersifat local seperti *accelerando* atau *ritardando*. *Loudness* juga ditulis secara langsung seperti *piano, forte, crescendo* dan *diminuendo*. Karena

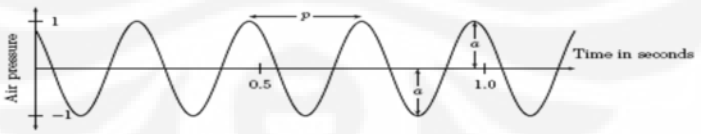
format *score* hanya berupa resep untuk membuat musik, maka musik hasil terjemahan dari *score* bisa bervariasi bergantung dari orang yang menerjemahkannya.



Gambar 2.1 Contoh *Score* dari Beethoven's Fifth [8]

2.1.2.2 Format Audio / *Waveform*

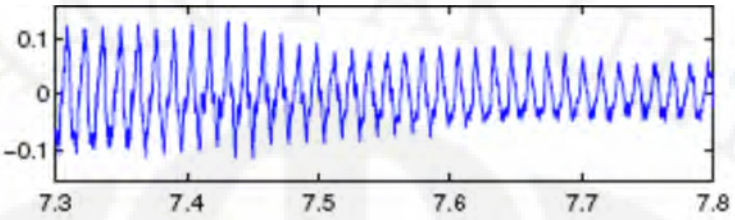
Waveform merupakan format sinyal akustik yang belum mengalami kompresi. Oleh karena itu, ukuran *file*-nya masih cukup besar. Sinyal suara berasal dari getaran yang dihasilkan oleh sumber bunyi contohnya: pita suara, senar yang bergetar atau diafragma pada drum. Getaran ini mengakibatkan perubahan kerapatan di udara kemudian sampai ke telinga manusia. Sinyal suara ini biasanya digambarkan dengan sumbu vertikal perubahan tekanan udara dan sumbu horizontal seperti terlihat pada Gambar 2.2. Dalam Gambar 2.2 yang dimaksud dengan periode adalah waktu antara dua puncak gelombang. Sedangkan frekuensi adalah kebalikan dari periode. Gelombang suara yang memiliki frekuensi tetap seperti pada Gambar 2.2 disebut *pure tone*, frekuensinya disebut *pitch* [8].



Gambar 2.2 *Waveform* dengan Frekuensi 4Hz [8]

Ketika memainkan sebuah nada pada suatu alat musik, gelombang hasil (musik*al tone*) keluaranya sangat berbeda bentuknya dengan *pure tone*. Ketidaksamaan ini diakibatkan oleh superposisi dengan *noise* dari alat, vibrasi dan

efek transien dari proses pemainan nada. Musikal *tone* seperti dapat dilihat pada Gambar 2.3.



Gambar 2.3 Detik ke 7.3 sampai 7.8 dalam Beethoven's fifth Bernstein [8]

Berbeda dengan *score* yang seperti resep untuk membuat musik, *waveform* sudah mencerminkan musik itu sendiri. Namun informasi seperti *onset time*, *pitch* dan durasi diberikan secara eksplisit. Selain itu seperti telah disebutkan diatas satu nada saja yang dihasilkan alat musik sudah merupakan gabungan dengan *noise* dari alat, vibrasi dan lain-lain, sehingga analisa dalam bentuk *waveform* sulit. Terutama dalam musik dimana semua nada dari gitar, piano, suara penyanyi, dan lain-lain, tercampur menjadi satu gelombang analisanya sangat kompleks.

Sinyal suara yang direpresentasikan dalam bentuk *waveform* seperti yang telah dijelaskan diatas adalah sinyal analog. Maksud analog disini adalah sinyal memiliki amplitudo dan waktu yang bersifat kontinu. Karena komputer hanya bisa menerima data dijital maka sinyal suara yang analog harus didijitalkan. Proses pendijitalan ini terdiri dari dua tahap yaitu *sampling* dan kuantisasi. *Sampling* adalah pencuplikan sinyal dengan interval waktu yang tetap. Tahap kedua adalah kuantisasi atau didekatkan pada nilai-nilai tertentu saja. Sebagai contoh sebuah CD musik memiliki *sampling rate* 44100 Hz dan 16 bit *encoding* artinya dalam satu detik 44100 titik disample dan dikuantisasi ke 65536 level. Proses dijitalisasi ini merupakan tranformasi yang bersifat *lossy* maksudnya ada informasi yang hilang selama proses ini berlangsung. Hilangnya informasi ini mengakibatkan ketidaksamaan dengan gelombang asal suara, ini yang disebut *aliasing* atau *quantization error*. Pada musik dijital besar kesalahan sudah diperhitungkan agar tidak tertangkap oleh telinga manusia [8].

2.1.2.3 Format *Hybrid*

Hybrid format adalah format yang bisa mengkodekan karakteristik nada dengan baik seperti *score* dan menampilkan representasi yang dinamis seperti *waveform* format. Bentuk *hybrid* yang paling sering digunakan adalah MIDI. MIDI merupakan kepanjangan dari Musik*al Instrument Dijital Interface*. MIDI adalah suatu standar yang memungkinkan berbagai *instrument* musik dijital dari berbagai merek bisa dimainkan bersama.

MIDI memungkinkan musisi mengontrol instrumen *dijital* secara *real time*. Contohnya pada piano dijital, ketika tuts piano dijital ditekan, 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. Oleh karena itu, jika kita ingin memainkan musik tanpa menekan tuts, kita dapat memberikan input berupa MIDI *messages* yang berisi informasi *note-on*, kecepatan dan *note-off* [8].

Sama seperti *score*, MIDI hanya merupakan pesan yang menunjukan bagaimana suatu musik dimainkan dan bagaimana cara memainkannya. Dalam MIDI tiap-tiap *timbre* memiliki *channel* masing-masing sehingga tidak terjadi pencampuran antar *timbre* sehingga analisis dalam MIDI lebih mudah.

2.1.3 Musik Dangdut

Dangdut merupakan salah satu dari *genre* seni musik yang menjadi ciri khas dari budaya Indonesia. Bentuk musik ini berasal dari musik Melayu tahun 1940-an. Dalam evolusi menuju bentuk kontemporer sekarang masuk pengaruh unsur-unsur musik India (terutama dari penggunaan tabla) dan Arab (pada cengkok dan harmonisasi). Sebagai musik populer, dangdut juga terbuka terhadap pengaruh bentuk musik lain, seperti keroncong, *rock*, pop, bahkan *house* musik.

2.2 *Query by Singing/Humming* (QbSH)

2.2.1 *Music Information Retrieval* (MIR)

Music Information Retrieval (MIR) adalah pengambilan informasi dari konten suatu musik, informasinya dapat berupa penyanyi, judul, *genre*, dll [8] Maksud dari pengambilan informasi berdasarkan *content* adalah informasi yang diambil benar-benar berasal dari isi musik tersebut bukan berasal dari teks sisipan-sisipan (judul, nama penyanyi dan *genre* biasanya disisipkan pada MP3). MIR bertujuan untuk memberikan kemudahan dalam mengakses dan menikmati musik. Contohnya dengan adanya pendeteksian *mood* lagu, seseorang yang sedang sedih dapat memilih musik yang memiliki *mood* gembira untuk menyemangatinya, dengan pendeteksian *chord* seseorang mengetahui *chord* musik yang sedang didengarnya atau bahkan dapat ikut memainkan *chord* juga, begitu pula dengan musik-*lyric synchronization*, atau dengan *query by content* yang memungkinkan pencarian lagu dengan menyanyikan, mensenandungkan melodinya atau dengan bertepuk tangan mengikuti ritme musik.

MIR sendiri sudah memiliki asosiasi pada tingkat intenasional, yaitu MIREX yang merupakan kepanjangan dari *Music Information Retrieval Exchange*. Setiap tahun MIREX mengadakan kompetisi tahunan dalam rangka meningkatkan kualitas dari MIR itu sendiri, sehingga nantinya MIR dapat diimplementasikan untuk masa yang akan datang.

2.2.2 Pengertian Umum *Query by Singing/Humming* (QbSH)

Salah satu bagian riset dari MIR yang cukup menarik banyak perhatian adalah *query by content*. *Query by content* adalah pencarian lagu berdasarkan content dari lagu tersebut, yang dimaksud konten disini terbatas pada ritme (dikenal dengan *query by taping*) atau melodi dari lagu tersebut (dikenal dengan *query by singing*/*humming*/*whistling*). Sistem percarian lagu yang dikenal sekarang hanya dapat mencari berdasarkan data yang disisipkan saja (judul, nama penyanyi,dll) dengan kata lain jika tidak diketahui judul, nama penyanyi, dll. Seluruh *database* harus dicari lagu yang diinginkan, hal ini tentunya sangat merepotkan [9]. Padahal yang pertama diingat orang dari lagu adalah isi dari lagu

tersebut seperti lirik, melodi dan ritme baru kemudian judul, nama penyanyi,dll. Jadi yang seharusnya menjadi dasar pencarian adalah konten dari lagunya.

Query by Singing/Humming (QbSH) merupakan salah satu cabang dari *Music Information Retrieval* (MIR). QbSH adalah suatu metode pencarian lagu dengan menggunakan senandung. Pencarian berdasarkan senandung ini dibutuhkan karena terkadang pencari lagu tidak mengetahui atau lupa dengan judul lagu atau nama penyanyinya, melainkan hanya mengetahui potongan lagu dari lagu saja. Potongan lagu tersebut dapat berupa suara vokal senandung. Dengan adanya QbSH ini maka akan dimungkinkan suatu pencarian suatu lagu dari suatu *database* dengan bersenandung sebagai *query*.

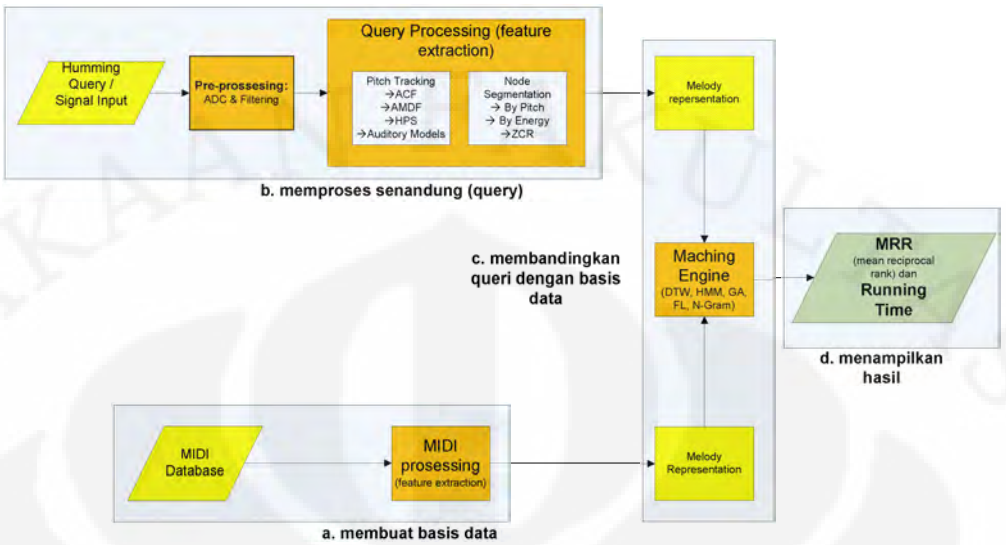
Dalam system QbSH pada intinya adalah pencocokan antara melodi yang disenandungkan oleh pencari lagu dengan melodi dari *database*, kemudian sistem akan memberikan *ranking* terhadap lagu yang paling mirip melodinya dengan lagu yang dimaksud ada dalam peringkat pertama. Tujuan dari diberikan *ranking* bukannya hanya satu kandidat adalah untuk mengantisisipasi bila ada beberapa versi lagu atau ada lagu yang memiliki kemiripan melodi.

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

a. Menentukan bagian mana yang paling diingat dari suatu melodi.
b. Melodi pada suatu lagu telah tercampur dengan bermacam-macam nada lain bahkan terdapat lebih dari dua melodi pada suatu lagu, sangat penting untuk menentukan melodi mana yang lebih dominan dan mudah diingat.
c. Sistem QbSH lebih menekankan pada kemiripannya saja bukan mencari melodi yang sama persis.

2.2.3 Prinsip Dasar *Query by Singing/Humming*

Diagram blok dari proses *Query by Singing/Humming* adalah sebagai berikut:



Gambar 2.4 Diagram Blok dari QbSH [2]

Proses *Query by Singing/Humming* memiliki beberapa langkah [3], diantaranya:

a. Membuat *Database* (*database*)

   Database ini dapat dibuat berdasarkan format audio lagu tertentu. Format lagu yang akan dibuat sebagai *database* biasanya berupa MIDI. Selanjutnya dari suatu format suatu lagu tersebut akan diambil beberapa jenis *feature* yang mencirikan suatu melodi dari lagu tersebut. Kemudian *feature* yang diambil tersebut akan menjadi representasi simbol yang disimpan dalam *database*.

b. Memproses Senandung Sebagai *Query* (*Query Feature Extraction*)

   Format audio dari *query* yang sering digunakan adalah WAV (*Waveform Audio Format*). Selanjutnya dari WAV tersebut akan diambil juga beberapa jenis *feature* yang sesuai dengan *database* yang telah dibuat, sehingga akan didapatkan representasi simbol dari WAV tersebut. Representasi simbol dari *query* ini yang kemudian akan digunakan sebagai *query* dalam pencarian pada *database*.

c. Membandingkan *Query* dengan Melodi yang Terdapat pada *Database* (*Matcing Engine*)

Proses ini merupakan proses pencarian melodi dari *database* yang sesuai dengan *query* yang diinginkan. Proses ini merupakan proses yang akan membutuhkan waktu yang cukup lama. Perbandingan biasanya dilakukan dengan membandingkan setiap representasi simbol dari *query* dengan setiap entri pada *database*.

d. Menampilkan Hasil (MRR)

Hasil yang diberikan tidak harus diberikan dalam satu hasil yang terbaik, melainkan dalam daftar yang berisikan sepuluh sampai dua puluh hasil yang terbaik.

2.2.3.1 Pembuatan *Database*

Database dibuat sebagai *domain* dari sistem QbSH. Untuk membuat *database* dibutuhkan lagu. Format audio dari lagu yang akan dijadikan *database* dapat berupa MIDI, WAV, MP3, dan lain-lain. Namun pada umumnya untuk membuat suatu *database* dari sebuah lagu digunakan format MIDI. Penggunaan format ini dimaksudkan untuk memudahkan dalam pengambilan *feature* dari sebuah lagu. Selain itu format MIDI juga memiliki ukuran yang sangat kecil jika dibandingkan dengan format audio.

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. Metode pengambilan *feature* ini akan cenderung lebih mudah jika format lagu yang digunakan ada MIDI. Selanjutnya *feature* yang telah diekstrak dari lagu akan menjadi representasi melodi dan disimpan dalam *database* tersebut.

2.2.3.2 Pemoresan Senandung Sebagai *Query*

Format dari senandung yang digunakan tidak dapat berupa MIDI, hal ini dikarenakan senandung berasal dari suara manusia. Format yang biasa digunakan sebagai senandung adalah WAV. Selanjutnya dari format audio 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.

Proses ekstraksi *feature* biasanya dimulai dengan tahap *preprosessing*, yaitu *sampling* dan *filtering* (penyaringan). Selanjutnya proses dilanjutkan dengan *pitch tracking* (pengambilan *pitch*) dan *note segmentation* (pemisahan antar not). Penjelasan singkat dari proses-proses tersebut adalah sebagai berikut:

a. *Sampling* dan *Filtering*

   Proses *samping* ini dimaksudkan untuk mengubah sinyal masukan yang berupa sinyal analog menjadi sinyal dijital. Selanjutnya sinyal dijital tersebut akan disaring (*filter*) dengan menggunakan *low-pass filter*. Hal ini dimaksudkan untuk mengurangi *noise* yang memiliki komponen frekuensi tinggi.

b. *Pitch Tracking*

   Pitch tracking adalah suatu metode untuk mendapatkan frekuensi dasar (*fundamental frequency*) F0 dari suatu sinyal suara, yang nantinya menjadi *feature* untuk representasi melodi. Beberapa metode dapat digunakan dalam *pitch tracking* antara lain: *Time Domain Method, Frequency domain Method, dan Auditory Model-Based Method. Time Domain Method* merupakan metode yang paling sering digunakan untuk mengestimasi F0. 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 *Frequency domain Method* merupakan metode dengan menggunakan *two-way mismatch procedure*. Metode ini adalah mencari frekuensi yang paling mendekati frekuensi dasar (F0) dari frekuensi sinyal masukan. Terakhir untuk metode *Auditory Model-Based Method,* konsep dasarnya adalah memodelkan sinyal masukan menjadi dalam respon frekuensi dari telinga tengah manusia. Kelemahan dari metode ini adalah tidak cocok dipakai untuk sinyal yang memiliki *pitch* cukup tinggi. Konsep ini lebih sering dipakai pada aplikasi *speech recognition*. [7].

c. *Note Segmentation*

Note segmentation adalah suatu metode untuk mendapatkan durasi dari suatu not atau dalam dunia musik disebut juga sebagai nilai not. *Note segmentation* berkaitan erat dengan *onset* dan *offset detection*. *Onset* adalah penanda dari mulainya suatu not, sedangkan *offset* adalah penanda dari berakhirnya suatu not. *Note segmentation* juga berfungsi sebagai penanda apakah dalam suatu segmen signal tertentu berada dalam keadaan diam, terkena *noise*, atau terdapat suara. *Note segmentation* dapat dilakukan dengan dua cara, yaitu berdasarkan amplitudo atau energi dan berdasarkan *pitch* [7].

Segmentasi berdasarkan amplitudo adalah membagi segmen dengan melihat amplitudo sinyal dalam domain waktu. Proses segmentasi ini sederhana dan sangat mudah di implementasikan. Namun segmentasi ini tidak tepat jika digunakan dalam kondisi *real-time*. Selain itu, jika ingin menggunakan segmentasi jenis ini maka setiap pengguna aplikasi yang ingin mencari lagu harus menyelipkan suatu tanda ("da" atau "ta") diantara not yang disenandungkannya. Hal ini tentunya akan menyulitkan pengguna aplikasi ini[7].

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[7].

2.2.3.3 Pembandingan Masukan *Query* dengan *Database*

Bagian ini merupakan inti dari sistem QbSH, karena pada bagian ini akan 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: [5]

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

2.2.3.4 Penampilan Hasil *Query*

Sampai saat ini hasil dari sistem QbSH masih jauh dari sempurna, oleh karena itu, hasil yang ditampilkan biasanya masih dalam bentuk daftar sepuluh hasil terbaik yang mendekati *query*. The MUSART mendefinisikan suatu nilai yang menyatakan hasil QbSH dengan menggunakan berbagai metode. Nilai tersebut adalah *mean reciprocal rank* (MRR), yang dirumuskan sebagai berikut [5]:

$$MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{r_i} \tag{2.1}$$

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

2.3 Dynamic Time Warping

Dynamic Time Warping (DTW) adalah suatu teknik untuk mengurangi nilai ketaksamaan antara dua *template*/*sequence* yang disebabkan oleh pergeseran waktu [10]. Dengan kata lain, DTW juga dapat dikatakan sebagai teknik untuk mengarahkan dua *senquence* yang bersifat *time dependent* sehingga dapat dihitung jarak terpendek diantara keduanya.

2.3.1 Konsep DTW

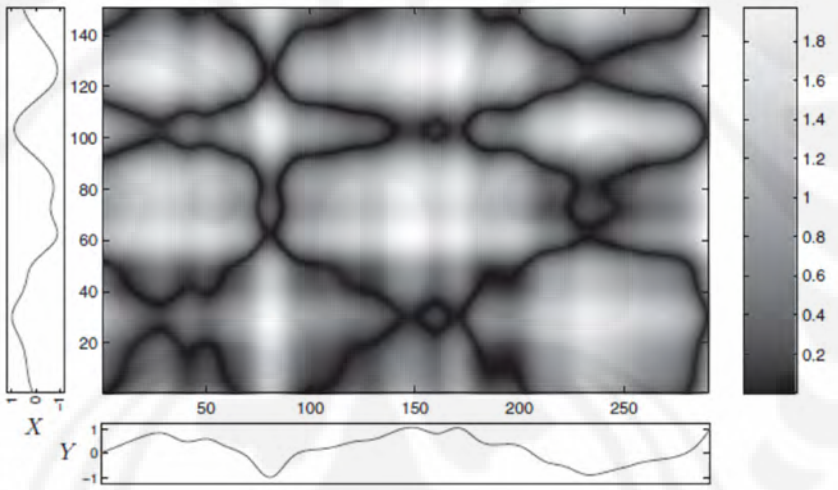
Tujuan dari DTW adalah untuk membandingkan dua *time dependent sequence*, X := (x_1, x_2, …, x_N) yang memiliki panjang N dan Y := (y_1, y_2, …, y_M) yang memiliki panjang M. Selanjutnya dibuat suatu ruang *feature* $\mathcal{F}$, dimana x_n

dan $y_m \in \mathcal{F}$, dan n $\in$ [1:N] dan m $\in$ [1:M]. Untuk membandingkan *feature* x dan y, kemudian di definisikan *local cost measure* atau *local distance measure* yang didefinisikan sebagai fungsi [8]:

$$c : F \times F \text{ untuk } \mathrm{R} \geq 0 \tag{2.2}$$

nilai c(x,y) akan menunjukan nilai minimum jika x dan y memiliki kemiripan satu sama lain, dan c(x,y) akan menunjukkan nilai maksimum apabila tidak ada kemiripan satu sama lain.

Selanjutnya jika dihitung besar *local cost measure* untuk setiap pasangan dari setiap elemen pada X dan Y, maka akan didapatkan *cost matrix* C $\in$ R^{NXM}, yang didefinisikan C(n,m) := c(x_n, y_m). Tujuan selanjutnya adalah mencari nilai X dan Y yang menghasilkan besar *cost* minimum.

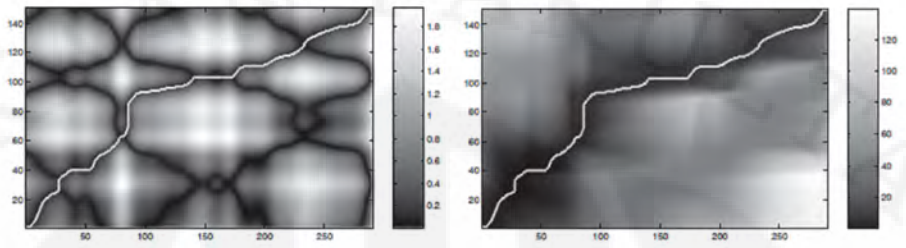


Gambar 2.5 Contoh *Cost Matrix* [8]

Pada Gambar 2.5 diatas menunjukan *cost matrix* dimana daerah yang berwarna hitam menunjukan nilai *cost* yang paling minimum, demikian juga sebaliknya.

Terlihat pada Gambar 2.5 bahwa daerah yang dilalui kurva dengan *cost* minimum merupakan tempat dimana kedua *sequence* tersebut *matching*. Terlihat bahwa banyak *path* yang dapat dipilih hingga sampai ke (n,m). Gambar 2.6

merupakan contoh pemilihan *path* dengan *cost* paling minimum dari *cost matrix* di atas (Gambar 2.5).



Gambar 2.6 *Path* Minimum yang Dipilih [8]

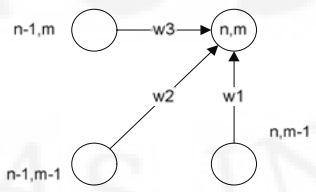
Pencariaan *path* biasanya dilakukan dengan menggunakan *accumulate cost matrix* D(n,m). Dalam pembentukannya, D(n,m) sangat dipengaruhi oleh *step size* dan *local weight* yang akan dijelaskan pada bagian selanjutnya.

2.3.2 Variasi DTW

Berbagai variasi modifikasi dari DTW dapat dilakukan untuk meningkatkan tingkat akurasi dan kecepatan dari proses komputasi[8]. Berbagai variasi yang dapat dilakukan antara lain dapat dilakukan dengan menvariasikan *step size*, *local weight*, *boundary conditions* dan perhitungan jarak.

2.3.2.1 *Step Size* dan *Local Weight*

Step size merupakan salah satu komponen penting yang digunakan untuk menentukan rute yang paling optimal dari *path*. Pemilihan *step size* akan berpengaruh terhadap hasil dan kecepatan proses komputasi dari perhitungan DTW. *Local weight* adalah pembobotan yang dilakukan pada *step size* untuk mendapatkan tingkat akurasi yang lebih baik. Gambar 2.7 adalah contoh *step size* dan *local weight* dan penerapannya dalam perhitungan D(n,m).



Gambar 2.7 Contoh *Step Size* dan *Local Weight*

Step size merupakan arah yang diperbolehkan, dalam hal ini 0^o,45^o, dan 90^o. *Local Weight* ditunjukan oleh nilai w1, w2 dan w3. Berdasarkan Gambar 2.7 di atas terlihat bahwa untuk membentuk D(n,m) dapat dilakukan dengan perhitungan sebagai berikut:

$$D(n,m) = \min \begin{Bmatrix} D(n,m-1) + w_1 \bullet c(x_n, y_n) \\ D(n-1,m-1) + w_2 \bullet c(x_n, y_n) \\ D(n-1,m) + w_3 \bullet c(x_n, y_n) \end{Bmatrix} \quad (2.3)$$

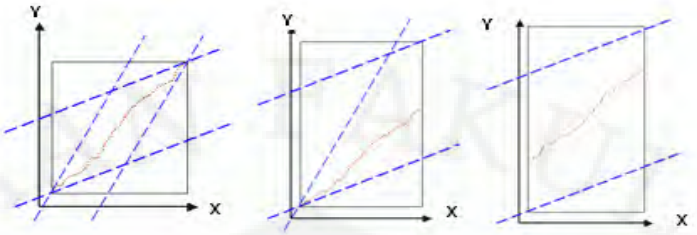
Perhitungan diatas dapat diselesaikan dengan menggunakan metode *dynamic programming*. Konsep dari *dynamic programming* adalah prinsip optimalitas yang menyatakan bahwa apapun keadaan awal dan akhir, keputusan saat ini harus memegang konsep optimalitas yang tergantung dari keputusan sebelumnya [10]. Terlihat dari rumus diatas apabila akan mencari besar D(n,m) maka terlebih dahulu harus mempertimbangkan D(n,m-1), D(n-1,m-1), dan D(n-1,m).

2.3.2.2 *Boundary Conditions*

Boundary conditions adalah suatu aturan yang menentukan keadaan awal dan akhir dari suatu *sequence* pada DTW. Terdapat tiga jenis *boundary conditions*, yaitu[4]:

1. *Anchored Beginning and Anchored End* artinya jarak dari kedua *time dependent sequence* X dan Y harus dihitung dari posisi *cost matrix* c(1,1) sampai c(N,M).
2. *Anchored Beginning and Free End* artinya jarak dari kedua *time dependent sequence* X dan Y harus dihitung dari posisi *cost matrix* c(1,1) dan ujungnya bebas, tidak harus berakhir pada c(N,M). *Boundary condition* ini biasanya diterapkan pada QbSH.
3. *Free Beginning and Free End* artinya jarak dari kedua *time dependent sequence* X dan Y dihitung dari posisi awal dan akhir bebas. *Boundary condition* ini biasanya diterapkan pada *speaker dependent*.

Gambar 2.8 menunjukan tipe boundary conditions yang digunakan pada metode DTW.



Gambar 2.8 *Boundary Conditions*[4] (dari kiri ke kanan: *Anchored Beginning and Anchored End, Anchored Beginning and Free End, Free Beginning and Free End)*

2.3.2.3 Perhitungan Jarak

Cost matrix dapat dihitung dengan menggunakan beberapa perhitungan jarak. Beberapa perhitungan jarak yang sering digunakan pada DTW adalah:

1. *Manhattan Distance*. Jarak dihitung sebagai selisih dua nilai yang diabsolutkan[8].

$$c(x_n, y_m) = |x_n - y_m| \tag{2.4}$$

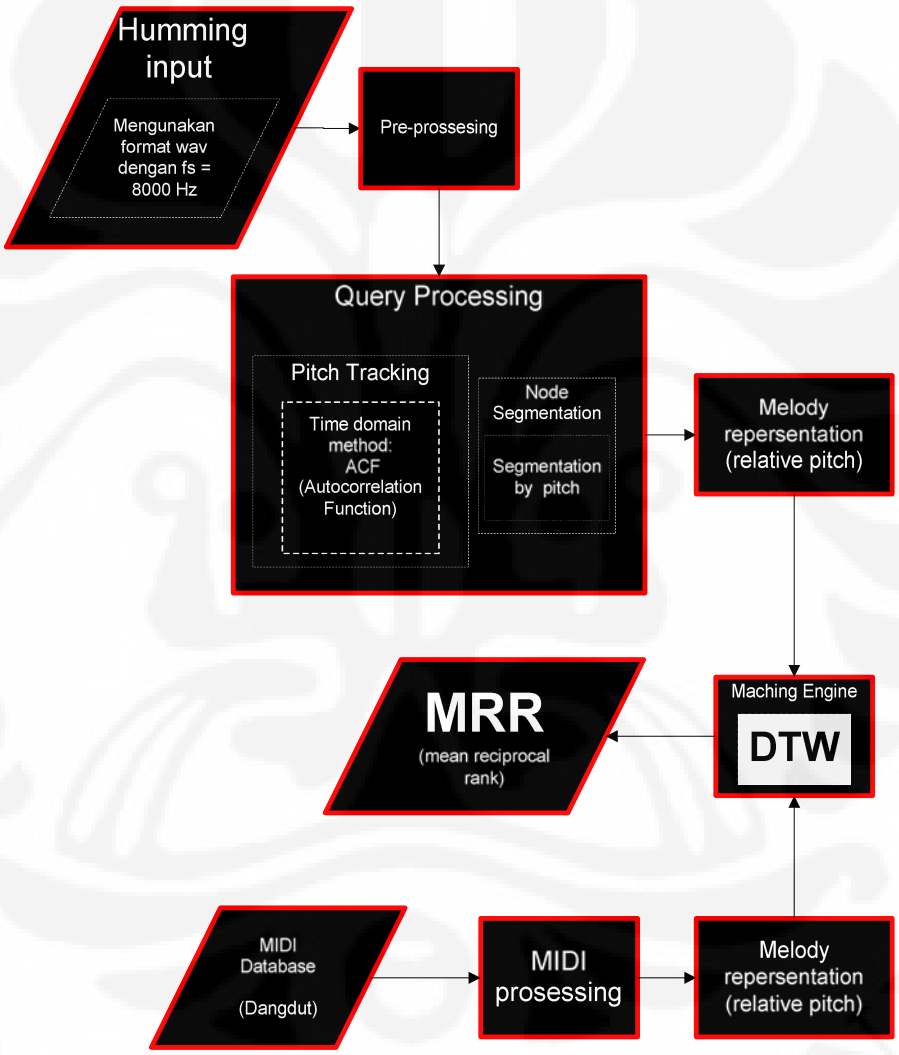
2. *Squared Euclidean Distance*. Jarak dihitung sebagi selisih dua nilai yang dikuadratkan[11].

$$c(x_n, y_m) = (x_n - y_m)^2 \tag{2.5}$$

BAB 3
PERANCANGAN SIMULASI

3.1 Diagram Perancangan Simulasi QbSH

Gambar 3.1 ini adalah gambar diagram perencanaan simulasi dari skripsi ini.



Gambar 3.1 Diagram Sistem QbSH yang Dirancang

Secara umum terdapat 3 proses penting yang akan dilakukan pada simulasi ini. Ketiga proses tersebut adalah pengolahan *query*, pembentukan *database*, dan

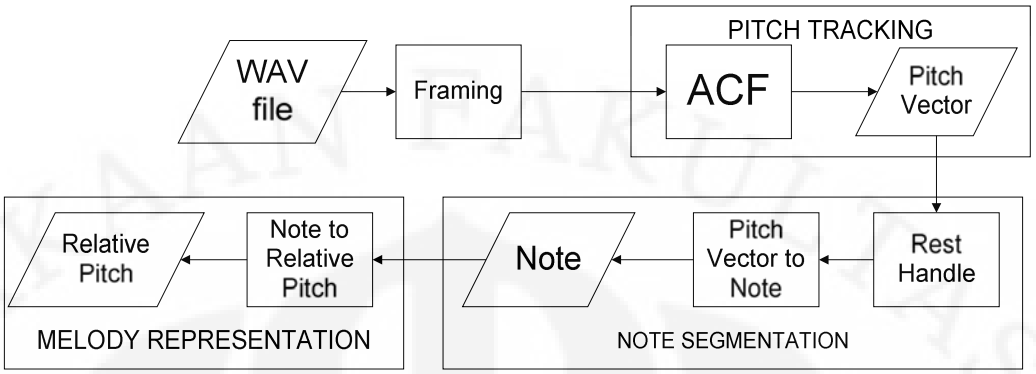
proses *matching engine*. Pada pengolahan *query* terjadi beberapa proses, diantaranya adalah proses *framing*, *pitch tracking* dan *note segmentation*. Inti dari proses ini adalah mengubah sinyal audio menjadi *melody representation*. Pada proses pembentukan *database* dilakukan proses perubahan dari format MIDI menjadi *melody representation*. Selanjutnya pada *matching engine* akan dibandingkan *melody representation* yang dihasilkan pada *query* dengan *database*. *Matching engine* yang digunakan pada simulasi ini adalah DTW.

3.2 Pengolahan *Query*

Query sebagai masukan yang digunakan adalah format audio WAV. Penggunaan format WAV dipilih untuk mengikuti penelitian-penelitian sebelumnya. Format WAV merupakan format dari sinyal audio tanpa menggunakan kompresi. Dengan begitu dalam pemprosesan *query* tidak diperlukan proses dekompresi lagi (waktu proses menjadi lebih cepat). Selain itu apabila konsep QbSH ini dibawa pada kondisi *real-time* penggunaan format WAV sebagai *query* akan menjadi keuntungan tersendiri, mengingat masukan dari *microphone* yang juga menggunakan format WAV. Sinyal audio yang digunakan sebagai *query* menggunakan frekuensi *sampling* sebesar 8000 Hz dengan tipe *channel mono*. Sampel *query* diambil dari 6 orang yang terdiri dari 3 laki-laki dan 3 wanita, dan masing-masing menyanyikan 10 potongan lagu berbeda. Potongan lagu dinyanyikan dari awal mulai lagu selama 10 detik.

Pengolahan *query* ini dilakukan dengan menggunakan bantuan *utility toolbox* [13], *speech and audio processing toolbox*[14], dan *melody recognition toolbox* [15].

Gambar 3.2 berikut ini adalah diagram alir pengolahan *query*.



Gambar 3.2 Diagram Alir Pengolahan *Query*

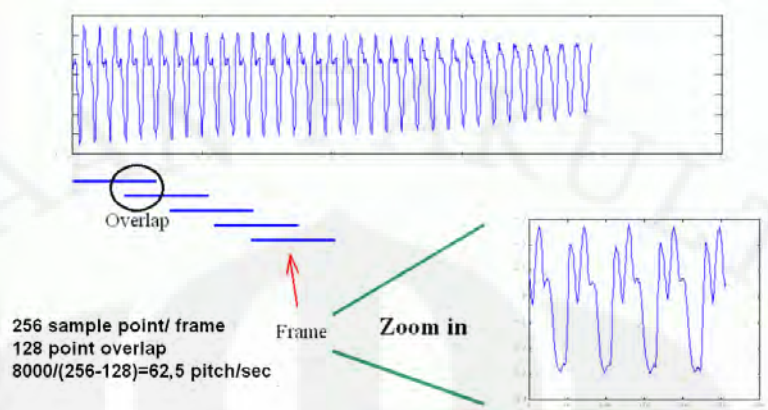
3.2.1 Pembentukan *Frame* (*Framing*)

Sebelum melakukan *pitch tracking*, suatu sinyal input terlebih dahulu harus dipotong menjadi beberapa *frame*. Ukuran *frame* yang digunakan tergantung dari *range* frekuensi dasar yang ingin diperoleh. Pada sistem QbSH ini 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 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[4].

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

$$frame_{\max} = 2 \bullet \frac{f_s}{f_{0\min}} = 2 \bullet \frac{8000}{62,5} = 256 \text{ sample point}$$

Jadi pada simulasi ini akan digunakan lebar *frame* sebesar 256 *sample point*. Untuk mencegah terjadinya diskontinuitas maka *frame* dibuat saling *overlap* sebesar 128 *sample point*.



Gambar 3.3 Ukuran *Frame* dan *Overlap* [4]

3.2.2 Metode *Pitch Tracking*

Metode *pitch tracking* yang digunakan adalah dengan menggunakan *time domain method* dengan menggunakan fungsi ACF (*Autocorrelation Function*). Konsep dasar dari metode ini adalah untuk mengukur korelasi dalam domain waktu dari suatu sinyal dengan sinyal tersebut ketika mengalami pergeseran waktu (*time-shifted version*).

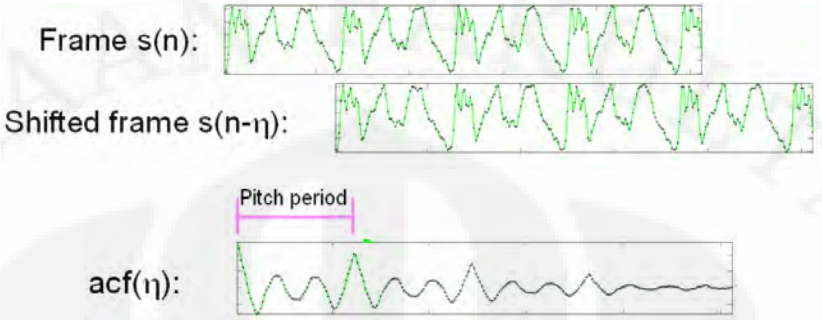
Setelah sinyal terbagi menjadi beberapa *frame*, setiap *frame* tersebut akan dicari besar dari periode dasar. Setelah didapatkan besar periode dasar tersebut, maka akan didapatkan besar frekuensi dasar yang merupakan *pitch* dari *frame* tersebut. Prosesnya adalah sebagai berikut:

a. Diketahui suatu sinyal dalam suatu *frame* s(n)
b. Sinyal s(n) tersebut digeser sebanyak η sehingga didapatkan sinyal s(n- η)
c. Menghitung ACF (*autocorrelation function*) untuk setiap nilai η yang dimulai dari satu sampai fungsi ACF menghasilkan nilai yang maksimal [4].

$$\mathrm{ACF}(\eta)=\sum_{n=t}^{t+frame-1} s(n)s(n-\eta) \tag{3.2}$$

d. Selisih antara nilai η yang menghasilkan lokal maksimum dari fungsi ACF merupakan besar dari periode dasar dari sinyal tersebut.

e. Setelah mendapatkan periode dasar maka akan didapatkan besar frekuensi dasar yang merupakan *pitch* pada *frame* tersebut.



Gambar 3.4 Gambar Pencarian Periode Dasar dengan Menggunakan ACF [4]

Proses ACF ini akan terus berlangsung untuk *frame* yang lain, sehingga didapatkan urutan *pitch* dari suatu sinyal audio yang disebut sebagai *pitch vector*. Durasi satu *pitch vector* adalah sama dengan lebar *frame*, yaitu sebesar 256 *sample point* atau setara dengan *sample point* dibagi dengan frekuensi cuplik, yaitu 256/8000 = 0,032 detik.

3.2.3 Metode *Note Segmentation*

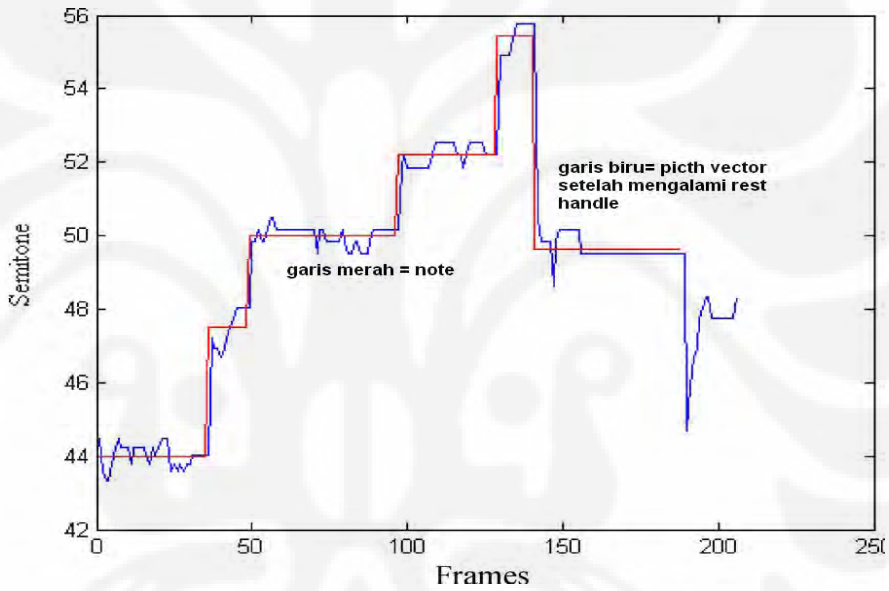
Secara umum metode *note segmentation* yang digunakan adalah jenis *segmentation by pitch*. Artinya *note* dibentuk dan dipisahkan berdasarkan adanya perubahan *pitch* yang terjadi[12].

Pitch vector yang dihasilkan pada proses *pitch tracking* kemudian akan mengalami proses *note segmentation*. Tujuan dari *note segmentation* adalah mendapatkan urutan *note* (melodi) dari suatu sinyal audio dengan cara menggabungkan *pitch vector* yang memiliki kemiripan.

Sebelum *pitch vector* diubah menjadi *note*, terlebih dahulu nilai 0 *pitch vector* dihilangkan. Nilai 0 pada *pitch vector* menunjukkan suatu keadaan *silent*. Metode untuk menghilangkan nilai 0 pada *pitch vector* dinamakan *rest handle*[15]. Selanjutnya setelah mengalami *rest handle*, *pitch vector* kemudian akan diubah menjadi *note* dengan bantuan fungsi *pv2note* pada *toolbox* [15]. Untuk mengubah dari *pitch vector* menjadi *note* maka harus memperhatikan aturan berikut[16]:

1. Suatu *note* baru akan terbentuk apabila perbedaan antara *pitch* yang sekarang dengan *pitch* yang selanjutnya memiliki perbedaan yang lebih besar dari 0,55 *semitones*. Kemudian nilai *note* tersebut adalah nilai tengah dari kumpulan elemen *pitch* yang membentuk *note* tersebut.
2. Jika *threshold* durasi (durasi dari *note*) kurang dari 0,128 detik (4 *pitch vector*) maka *note* tersebut akan menjadi satu dengan *note* sebelumnya yang berada dibawah *threshold* 0,55 *semitones.*

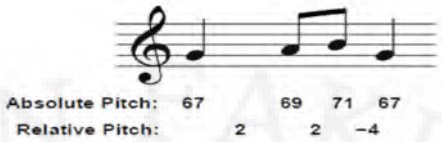
Proses note segmentation dapat dilihat pada Gambar 3.5.



Gambar 3.5 Proses *Note Segmentation*[16]

3.2.4 *Melody Representation* untuk *Query*

Setelah *note* didapatkan dari suatu *query*, maka selanjutnya adalah bentuk *note* tersebut diubah menjadi bentuk lain yang lebih baik untuk dibandingkan dengan *database*. Dalam hal digunakan bentuk *relative pitch* [5]. *Relative pitch* adalah interval *pitch* antar dua *note* yang berdekatan. Tujuan dari digunakannya *relative pitch* ini adalah untuk mengantisipasi terjadinya *key transposition*. *Key transposition* merupakan perubahan nada dasar saat menyanyikan sebuah lagu. Gambar 3.6 menunjukan contoh *relative pitch* untuk suatu potongan lagu.

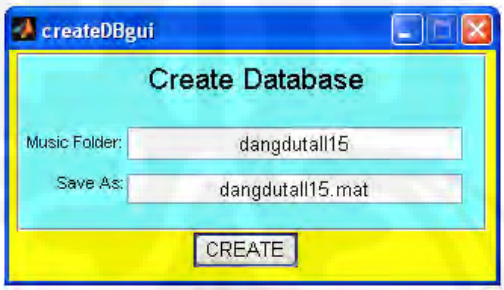


Gambar 3.6 *Relative Pitch* [5]

3.3 Pembentukan *Database*

Database yang digunakan merupakan kumpulan dari *file* dengan format MIDI. Format MIDI dipilih karena pada penelitian-penelitian sebelumnya juga menggunakan format MIDI. Penggunaan format MIDI ini memberikan keutungan terhadap besar dari *database* yang akan dibuat. Format MIDI memiliki ukuran yang kecil dibandingkan dengan format audio (WAV, mp3, au, dan lain-lain). Besar satu file lagu format MIDI biasanya berkisar antara 10 sampai 200 kilo *byte*.

Database yang digunakan berisikan lagu-lagu dangdut sebanyak 15 buah yang dipilih secara acak. MIDI yang digunakan merupakan monoponik yang terdiri dari *channel* vokal saja. Proses pembentukan *database* dilakukan dengan menggunakan bantuan MIDI *toolbox* [17].



Gambar 3.7 Program untuk Membuat *Database*

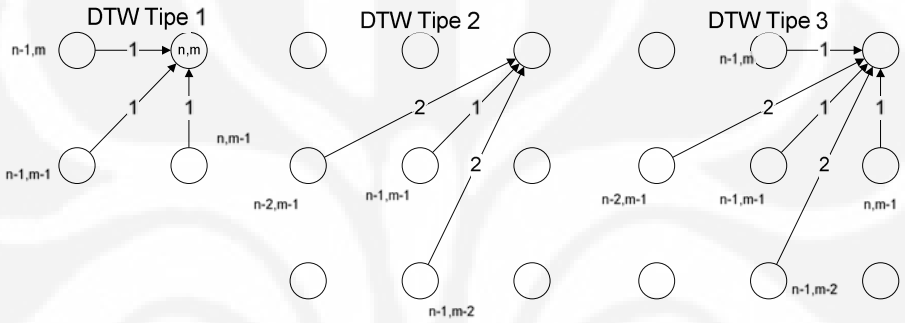
Gambar 3.7 menunjukkan tampilan dari program pembuatan *database*. Pada bagian "*Music Folder*" akan dituliskan nama *folder* yang berisikan file MIDI dari lagu dangdut. Selanjutnya pada bagian "*Save As*" akan dituliskan nama file hasil pembuatan *database* dalam bentuk .MAT. *Feature* yang dihasilkan dari pembuatan *database* ini adalah *note* maka agar dapat diproses dalam *matching engine*, bentuk *note* ini harus diubah juga ke dalam bentuk *relative pitch* (proses yang sama seperti pada *query*).

3.4 DTW yang Digunakan

Pada simulasi ini, penulis akan mencoba berbagai jenis DTW ditinjau dari sisi *step size*, *local weight*, dan *free end*.

3.4.1 *Step size* dan *Local Weight* yang Digunakan

Pada skripsi ini didefinisikan 3 tipe DTW berdasarkan *step size* dan *local weight*. Gambar 3.8 menunjukan 3 tipe DTW yang digunakan pada simulasi ini.



Gambar 3.8 Tipe *Step size* dan *Local Weight* DTW [4][18]

Berikut ini adalah 3 tipe DTW yang digunakan pada simulasi ini.

1. DTW tipe 1

   Pada tipe 1 ini memiliki 3 *step size*, yaitu (n-1,m), (n,m-1), dan (n-1,m-1). Setiap *step size* memiliki besar *local weight* yang sama, yaitu 1[4].

$$D(n,m) = \min \begin{Bmatrix} D(n,m-1) + c(x_n, y_m) \\ D(n-1,m-1) + c(x_n, y_m) \\ D(n-1,m) + c(x_n, y_m) \end{Bmatrix} \quad (3.3)$$

2. DTW tipe 2

   Pada tipe 2 ini memiliki 3 *step size*, yaitu (n-2,m-1), (n-1,m-1), dan (n-1,m-2). *Step size* (n-2 ,m-1) dan (n-1,m-2) memiliki besar *local weight* 2, sedangkan untuk *step size* (n-1,m-1) memiliki besar *local weight* 1[4].

$$D(n,m) = \min \begin{Bmatrix} D(n-1,m-2) + 2 \bullet c(x_n, y_m) \\ D(n-1,m-1) + c(x_n, y_m) \\ D(n-2,m-1) + 2 \bullet c(x_n, y_m) \end{Bmatrix} \quad (3.4)$$

3. DTW tipe 3

Pada tipe 3 ini memiliki 5 *step size*. *Step size* dan *local weight* yang digunakan pada simulasi ini adalah gabungan dari tipe 1 dan 2, yaitu (n-1,m), (n-2,m-1),(n-1,m-1),(n-1,m-2) dan (n,m-1).

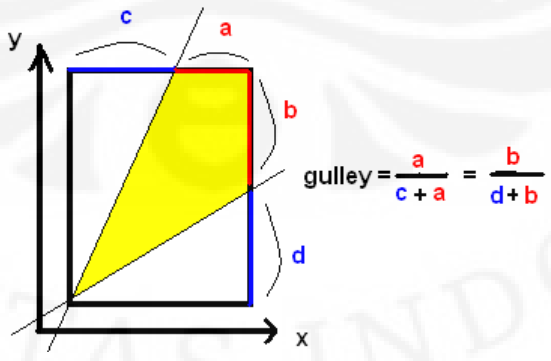
$$D(n,m) = \min \begin{Bmatrix} D(n-1,m-2) + 2 \bullet c(x_n, y_m) \\ D(n,m-1) + c(x_n, y_m) \\ D(n-1,m-1) + c(x_n, y_m) \\ D(n-1,m) + c(x_n, y_m) \\ D(n-2,m-1) + 2 \bullet c(x_n, y_m) \end{Bmatrix} \quad (3.5)$$

3.4.2 Perhitungan Jarak

Pada simulasi ini akan digunakan jenis perhitungan jarak *manhattan distance.*

3.4.3 *Free End*

QbSH yang dirancang dengan kondisi *boundary condition anchored beginning and free end*. Untuk menentukan skala *free end* maka dibuatlah suatu parameter yang disebut dengan *gulley*. *Gulley* adalah perbandingan antara daerah *free end* dengan dimensi *cost matrix*. Gambar 3.9 merupakan ilustrasi dari *gulley*.



Gambar 3.9 *Gulley*

Pada simulasi ini akan dilakukan ujicoba *gulley* dari 0 sampai 0.5 dengan interval 0.05.

3.5 Simulasi

Tujuan utama dari simulasi ini adalah untuk menentukan unjuk kerja dari setiap tipe DTW yang digunakan untuk *database* musik dangdut. Seperti yang dikatakan Profesor Roger Jang [4], "Pada kenyataannya, performa dari DTW tergantung pada *database* yang digunakan. Jadi tidak ada jaminan bahwa salah satu tipe selalu menjadi yang terbaik untuk semua jenis *database* yang digunakan". Oleh karena itu, dengan adanya simulasi ini akan dilihat tipe DTW yang lebih baik untuk *database* yang menggunakan musik dangdut.

Pada simulasi ini akan dikombinasikan antara tipe DTW berdasarkan *step size*, *local weight*, dan *gulley* untuk *database* dan *query* yang sama. Berikut ini adalah tampilan program simulasi. Parameter yang digunakan untuk mengukur performa dari variasi DTW adalah *Mean Reciprocal Range* (MRR).

Gambar 3.10 Tampilan Program Simulasi

Pada Gambar 3.10 terlihat tampilan dari program simulasi. *Check box* "Live Recording" digunakan untuk melakukan perekaman dari *query*. *Text box* dibawahnya digunakan untuk menuliskan nama *file* dari *query* yang akan dibandingkan dengan *database*. Untuk memasukkan *file database* dapat dilakukan pada *text box* "*File Database*". Selanjutnya parameter dari DTW yang digunakan dapat diatur pada kotak "Parameter DTW". Tombol "RUN" digunakan untuk menjalankan program simulasi.

Simulasi dilakukan dengan menggunakan *software* MATLAB 7.1 SP3. Pengambilan sampel *query* dilakukan dalam keadaan natural, *noise* diusahakan seminimum mungkin. Perangkat *hardware* yang digunakan adalah komputer dengan *processor core 2 duo* T6400 @ 2 GHz dan 2 Gb RAM.

BAB 4
HASIL SIMULASI DAN ANALISIS

4.1 Data Hasil Simulasi dan Pengolahan Data

Simulasi dilakukan dengan memvariasikan parameter tipe DTW dan *gulley*. Keseluruhan pengujian meliputi:

1. Uji coba sampel dengan tipe DTW 1, perhitungan jarak menggunakan *manhattan distance*, dan *gulley* dari 0 sampai 0.5.
2. Uji coba sampel dengan tipe DTW 2, perhitungan jarak menggunakan *manhattan distance*, dan *gulley* dari 0 sampai 0.5.
3. Uji coba sampel dengan tipe DTW 3, perhitungan jarak menggunakan *manhattan distance*, dan *gulley* dari 0 sampai 0.5.

Pengujian dilakukan terhadap 60 sampel *query* dan untuk setiap pengujian dilakukan perubahan parameter *gulley* sebanyak 11 kali (dari 0 sampai 0,5 dengan interval 0.05). Jadi untuk satu pengujian dilakukan simulasi sebanyak 660 kali.

Setelah melakukan pengujian terhadap tiga variasi tersebut, diperoleh hasil uji coba yang ditampilkan dalam Tabel 4.1 sampai 4.3. Pada setiap tabel akan ditampilkan nilai MRR dari setiap hasil pengujian kelompok sampel tertentu. Sampel disebut satu kelompok apabila dinyanyikan oleh orang yang sama. Setiap orang akan menghasilkan 10 buah sampel.

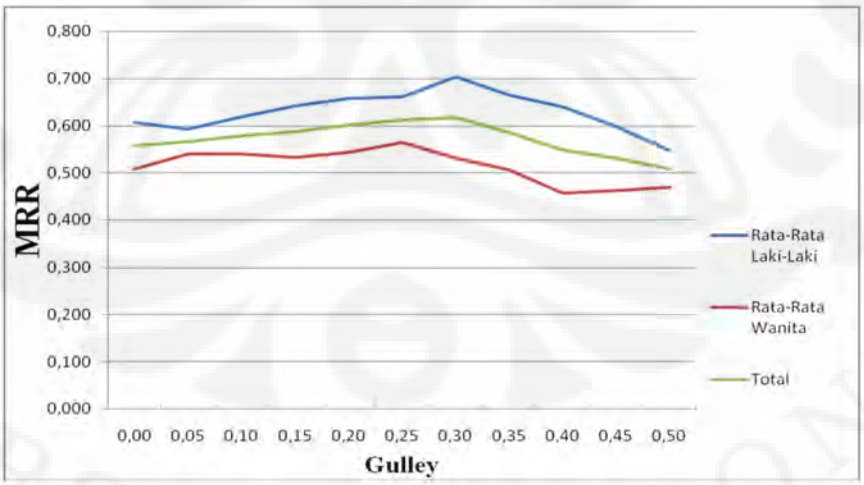
4.1.1 Hasil Uji Coba Tipe DTW 1 dengan *Manhattan Distance*

Pada Tabel 4.1 dapat dilihat hasil MRR untuk tipe DTW 1 dengan menggunakan *manhattan distance*.

Tabel 4.1 Hasil Uji Coba Tipe DTW 1 dengan *Manhattan Distance*

Gulley Factor	MRR								
	Laki-Laki1	Laki-Laki2	Laki-Laki3	Wanita1	Wanita2	Wanita3	Rata-Rata Laki-Laki	Rata-Rata Wanita	Total
0,000	0,5136	0,6267	0,675	0,5825	0,336	0,6067	0,605	0,508	0,557
0,050	0,5136	0,585	0,675	0,6417	0,3649	0,615	0,591	0,541	0,566
0,100	0,5053	0,6667	0,681	0,6325	0,3976	0,59	0,618	0,540	0,579
0,150	0,5225	0,7167	0,6843	0,5569	0,3958	0,6436	0,641	0,532	0,587
0,200	0,5678	0,75	0,6533	0,5258	0,4525	0,6533	0,657	0,544	0,600
0,250	0,7444	0,65	0,5867	0,5944	0,4525	0,6476	0,660	0,565	0,613
0,300	0,7944	0,6	0,7143	0,5167	0,3525	0,725	0,703	0,531	0,617
0,350	0,7144	0,5833	0,6944	0,5143	0,4002	0,6033	0,664	0,506	0,585
0,400	0,6878	0,5417	0,6867	0,4708	0,3694	0,5292	0,639	0,456	0,548
0,450	0,6848	0,5083	0,6	0,4694	0,3508	0,5658	0,598	0,462	0,530
0,500	0,5965	0,4417	0,6	0,4841	0,3508	0,5733	0,546	0,469	0,508

Gambar 4.1 menunjukkan grafik hasil uji coba tipe DTW 1.



Gambar 4.1 Grafik Uji Coba Tipe DTW 1 dengan *Manhattan Distance*

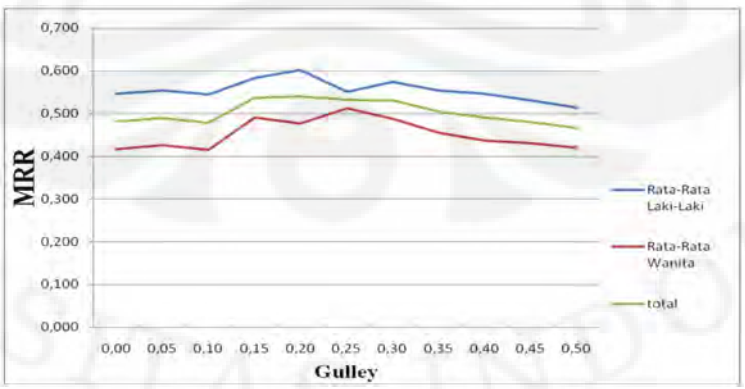
4.1.2 Hasil Uji Coba Tipe DTW 2 dengan *Manhattan Distance*

Pada Tabel 4.2 dapat dilihat hasil MRR untuk tipe DTW 2 dengan menggunakan *manhattan distance*.

Tabel 4.2 Hasil Uji Coba Tipe DTW 2 dengan *Manhattan Distance*

Gulley Factor	MRR								
	Laki-Laki1	Laki-Laki2	Laki-Laki3	Wanita1	Wanita2	Wanita3	Rata-Rata Laki-Laki	Rata-Rata Wanita	Total
0,000	0,2935	0,5935	0,7518	0,3718	0,3027	0,5748	0,546	0,416	0,481
0,050	0,2935	0,6152	0,7518	0,3686	0,3424	0,5628	0,554	0,425	0,489
0,100	0,3971	0,5992	0,6343	0,3825	0,4391	0,4219	0,544	0,415	0,479
0,150	0,4285	0,6492	0,6733	0,4242	0,4174	0,6278	0,584	0,490	0,537
0,200	0,4283	0,7125	0,665	0,3626	0,4294	0,6394	0,602	0,477	0,540
0,250	0,4346	0,5125	0,7033	0,4593	0,451	0,6261	0,550	0,512	0,531
0,300	0,5235	0,5	0,6992	0,3986	0,4426	0,6218	0,574	0,488	0,531
0,350	0,5108	0,5	0,6492	0,3986	0,3867	0,5775	0,553	0,454	0,504
0,400	0,5108	0,4667	0,6617	0,3986	0,3817	0,5275	0,546	0,436	0,491
0,450	0,5108	0,4667	0,6117	0,3986	0,3817	0,5075	0,530	0,429	0,480
0,500	0,5108	0,4167	0,6117	0,3986	0,3517	0,5075	0,513	0,419	0,466

Gambar 4.2 menunjukkan grafik hasil uji coba tipe DTW 2.



Gambar 4.2 Grafik Uji Coba Tipe DTW 2 dengan *Manhattan Distance*

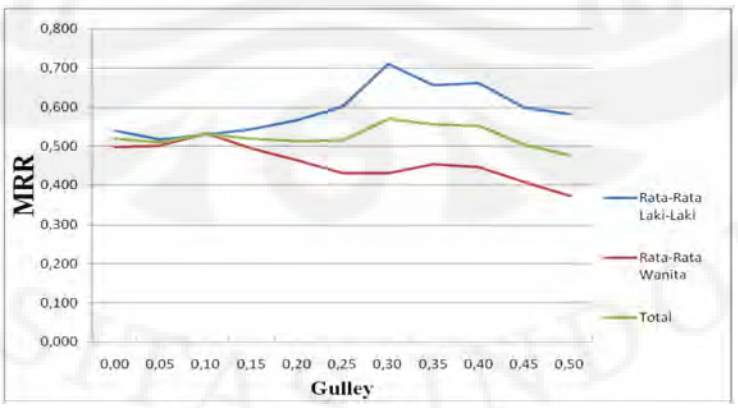
4.1.3 Hasil Uji Coba Tipe DTW 3 dengan *Manhattan Distance*

Pada Tabel 4.3 dapat dilihat hasil MRR untuk tipe DTW 3 dengan menggunakan *manhattan distance*.

Tabel 4.3 Hasil Uji Coba Tipe DTW 3 dengan *Manhattan Distance*

Gulley Factor	MRR								
	Laki-Laki1	Laki-Laki2	Laki-Laki3	Wanita1	Wanita2	Wanita3	Rata-Rata Laki-Laki	Rata-Rata Wanita	Total
0,000	0,4004	0,456	0,7587	0,3442	0,4249	0,7225	0,538	0,497	0,518
0,050	0,4004	0,39	0,7587	0,301	0,4799	0,7243	0,516	0,502	0,509
0,100	0,3594	0,4358	0,7933	0,3567	0,4799	0,7619	0,530	0,533	0,531
0,150	0,4951	0,3858	0,745	0,3283	0,4299	0,7233	0,542	0,494	0,518
0,200	0,516	0,4333	0,745	0,3343	0,3854	0,67	0,565	0,463	0,514
0,250	0,5367	0,3976	0,8667	0,4075	0,286	0,5983	0,600	0,431	0,515
0,300	0,85	0,3793	0,9	0,3792	0,3596	0,55	0,710	0,430	0,570
0,350	0,8083	0,3593	0,8	0,3792	0,3152	0,6667	0,656	0,454	0,555
0,400	0,8333	0,3462	0,8	0,3736	0,2924	0,67	0,660	0,445	0,553
0,450	0,745	0,3321	0,7167	0,3722	0,295	0,5533	0,598	0,407	0,502
0,500	0,725	0,3012	0,7167	0,3562	0,2728	0,4867	0,581	0,372	0,476

Gambar 4.3 menunjukkan grafik hasil uji coba tipe DTW 3.



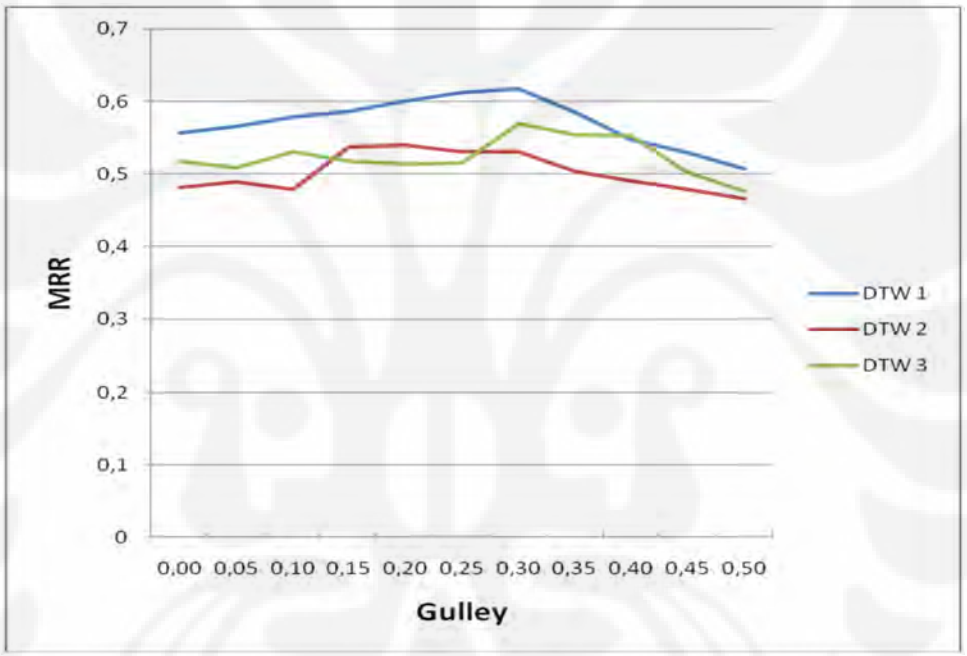
Gambar 4.3 Grafik Uji Coba Tipe DTW 3 dengan *Manhattan Distance*

4.2 Analisis

Secara umum analisis yang dilakukan pada penelitian ini mencakup beberapa hal, yaitu analisis tipe DTW, *gulley*, dan pengaruh jenis kelamin sampel.

4.2.1 Analisis Tipe DTW

Berdasarkan data yang didapatkan diperoleh bahwa tipe DTW 1 merupakan tipe yang menhasilkan nilai MRR tertinggi untuk hampir seluruh kondisi *gulley*. Selanjutnya disusul oleh tipe DTW 3 dan terakhir tipe DTW 2. Gambar 4.4 menunjukkan grafik perbandingan MRR tipe DTW 1,2 dan 3.



Gambar 4.4 Grafik MRR dari Tipe-Tipe DTW

Tipe DTW sangat berkaitan dengan karakterisitik dari *database*. Telah didefinisikan tiga tipe DTW yang digunakan pada simulasi ini. Tipe 1 dan 2 hanya memiliki satu persamaan *step size*, yaitu (n-1,m-1), sedangkan untuk tipe 3 merupakan gabungan dari *step size* pada tipe 1 dan 2.

Berdasarkan hasil yang diperoleh, *step size* pada tipe DTW 1 memberikan hasil MRR yang paling tinggi. Ini menunjukkan bahwa *step size* pada tipe 1 merupakan *step size* yang tepat untuk *database* musik dangdut yang digunakan oleh penulis. *Step size* yang tepat adalah *step size* yang tidak dimiliki oleh tipe 2,

yaitu *step size* horizontal (n-1,m) dan vertikal (n,m-1). Tipe 3 memiliki *step size* ini, maka nilai MRR juga lebih baik dari tipe 2. *Step size* ini sangat berkaitan erat dengan performa dari DTW.

Step size (n-1,m) dan (n,m-1) merupakan *step size* yang akan sangat mempengaruhi perhitungan jarak terhadap dua *vector* yang memliki nilai komponen berulang. Dari beberapa lagu pada *database* dapat kita lihat terjadi banyak perulangan *pitch* atau nada.

Hal ini dapat dilihat dari banyaknya nilai 0 yang muncul pada *relative pitch*. Nilai 0 pada *relative pitch* menunjukkan bahwa antara *pitch* tersebut dengan *pitch* sebelumnya tidak mengalami kenaikan atau penurunan interval atau dengan *pitch* sebelumnya.

Berikut ini akan ditampilkan potongan melodi awal dari beberapa lagu pada *database* yang memiliki banyak perulangan *pitch*.

1. Adu Buyung.

   Pitch : 74 76 76 76 76 76 76 74 76 78 78 74 76 76

   Relative Pitch: 2 0 0 0 0 0 -2 2 2 0 -4 2 0 0

2. Colak-Colek.

   Pitch: 60 65 65 65 65 65 67 67 68 67 67 65 67 68

   Relative Pitch: 5 0 0 0 0 2 0 1 -1 0 -2 2 1 0

3. Darah Muda.

   Pitch: 71 71 71 69 69 72 72 71 71 71 71 71 69 72

   Relative Pitch: 0 0 -2 0 3 0 -1 0 0 0 0 -2 3 -1

4. Jablai.

   Pitch: 71 69 68 69 69 74 76 76 76 76 76 76 76 76

   Relative Pich: -2 -1 1 0 5 2 0 0 0 0 0 0 0 0

5. Penasaran.

   Pitch: 69 69 69 69 69 69 69 69 69 69 69 69 71 72

   Relative Pitch : 0 0 0 0 0 0 0 0 0 0 0 0 2 1 -1

6. Syahdu.

   Pitch: 71 71 71 69 71 71 71 71 71 69 67 69 69 69

   Relative Pitch: 0 0 -2 2 0 0 0 0 -2 -2 2 0 0 -2

7. Kopi Dangdut.

   Pitch: 69 69 69 71 71 72 72 72 72 72 72 72 72 72

   Relative Pitch: 0 0 2 0 1 0 0 0 0 0 0 0 0 0

8. Mbah Dukun.

   Pitch: 52 59 59 57 59 59 59 57 57 57 57 62 62 62

   Relative Pitch: 7 0 -2 2 0 0 -2 0 0 0 5 0 0 2

Seperti yang telah diketahui sebelumnya setiap lagu dari *database* akan dibandingkan dengan *query*. Jarak *query* dengan salah satu lagu pada *database* akan semakin kecil apabila *query* tersebut memiliki kemiripan termasuk didalamnya kemiripan dalam jumlah perulangan *pitch*. Namun pada kenyataannya pada sebuah *query* sulit untuk didapatkan perulangan pitch yang tepat. Hal ini dikarenakan *pitch* pada *query* dibentuk dengan menggunakan *note segmentation by pitch* yang tidak memungkinkan terjadinya perulangan *pitch*. Besar kemungkinan jumlah perulangan *pitch* pada *query* tidak akan sama dengan perulangan *pitch* pada *database* untuk lagu yang dimaksud. Namun dengan adanya *step size* (n-1,m) dan (n,m-1) masalah ini dapat sedikit teratasi.

Berikut ini adalah contoh proses pencarian jarak DTW untuk lagu "Penasaran" sebagai salah satu lagu yang memiliki banyak perulangan nada. *Relative pitch* untuk *query* lagu "Penasaran" adalah:

2.0391, 0.8880, 0, 10.2673, -11.7914, 1.5241, -0.4269, 0.4269, -2.1400, 2.1400, -0.2248, 0.2248, 0, 0.4617, 3.6308, -0.5677, -1.0824, 1.0824, 1.1546, -1.4294, -0.8076, 0, -0.2584, -1.0041,

Terlihat bahwa pada urutan *relative pitch* pada *query* tidak memberikan perulangan. Bandingkan dengan *relative pitch* dari *database* untuk lagu "Penasaran" adalah:

0 0 0 0 0 0 0 0 0 0 0 2 1 -1 1 0 0 -1 0 0 0 1 -1 -2 -2 2 -2

Apabila kedua *sequence* melodi ini dibentuk menjadi *cost matrix*, maka hasilnya adalah seperti yang ditunjukan pada Gambar 4.5.

		Database												
	start	0	0	0	0	0	0	0	0	0	0	0	2	
QUERY	2,04	2,04	2,04	2,04	2,04	2,04	2,04	2,04	2,04	2,04	2,04	2,04	0,04	
	0,888	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89	1,11	
	0	0	0	0	0	0	0	0	0	0	0	0	2	
	10,27	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	8,27	
	-11,8	11,8	11,8	11,8	11,8	11,8	11,8	11,8	11,8	11,8	11,8	11,8	13,8	
	1,524	1,52	1,52	1,52	1,52	1,52	1,52	1,52	1,52	1,52	1,52	1,52	0,48	
	-0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	2,43	
	0,427	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	1,57	
	-2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	4,14	
	2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	2,14	0,14	
	-0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	2,22	
	0,225	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	1,78	
	0	0	0	0	0	0	0	0	0	0	0	0	2	
	0,462	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	1,54	
	3,631	3,63	3,63	3,63	3,63	3,63	3,63	3,63	3,63	3,63	3,63	3,63	1,63	
	-0,57	0,57	0,57	0,57	0,57	0,57	0,57	0,57	0,57	0,57	0,57	0,57	2,57	end point

Gambar 4.5 *Cost matrix* untuk Lagu "Penasaran"

Cost matrix yang terbentuk pada Gambar 4.5 mengalami *reverse*/pembalikan untuk arah ke atas dan bawah. Pada bab sebelumnya (gambar 2.6) terlihat bahwa arah DTW adalah dari pojok kiri bawah menuju kanan atas, namun pada Gambar 4.5 arah DTW adalah dari pojok kiri atas menuju kanan bawah. Hal ini disebabkan proses pembentukan matriks pada *software* simulasi (MATLAB) dimulai dari pojok kiri atas.

Gambar 4.5 memperlihatkan arah yang berwarna kuning merupakan langkah pada tipe 1 dan 3, sedangkan arah yang berwarna merah adalah langkah pada tipe 2. Terlihat pada gambar 4.5, tipe 2 tidak dapat mengkompensasi perulangan *relative pitch*, karena tidak memiliki *step size* (n-1,m) dan (n,m-1).

Namun, pada tipe 1 dan 3 dapat mengkompensasi perulangan tersebut. Akibatnya hasil perhitungan jarak pada tipe 2 menjadi tidak seakurat tipe 1 dan 3. Hal ini terjadi juga pada lagu lain yang memiliki banyak perulangan *pitch*. Akibatnya nilai MRR dari tipe 2 menjadi lebih kecil dibandingkan tipe 1 dan 3.

Perbedaan tipe 1 dan tipe 3 terletak pada jumlah *step size*. Jumlah *step size* pada tipe 3 lebih banyak dibandingkan dengan tipe 1. Semakin banyak jumlah *step size* akan memberikan toleransi pemerataan yang semakin besar. Hal ini tentunya akan menjadi kerugian untuk *database* yang memliki lagu-lagu dengan melodi yang hampir serupa. Jumlah *step size* yang semakin banyak tentunya akan membuat jarak antara *query* dengan suatu lagu dan lagu lainnya semakin berdekatan.

Tabel 4.4 merupakan contoh hasil perhitungan jarak antara *query* lagu "Kopi Dangdut" dengan menggunakan tipe 1 dan 3.

Tabel 4.4 Perbandingan Jarak Tipe 1 dan 3 untuk lagu “Kopi Dangdut”

RANKING	TIPE 1		TIPE 3	
	Jarak	Judul Lagu	Jarak	Judul Lagu
01.	35.1	**kopi-dangdut**	28.9	Syahdu
02.	38.2	Colak-Colek	29.0	sekuntum-mawar-merah
03.	41.5	SMS	30.4	Colak-Colek
04.	41.7	Syahdu	32.2	**kopi-dangdut**
05.	42.4	Penasaran	32.5	goyang-dombret
06.	42.7	Mabuk-Janda	32.5	Darah-Muda
07.	43.5	goyang-dombret	33.0	SMS
08.	44.3	Kuch-Kuch-Hotahei	33.7	Jablai
09.	44.6	sekuntum-mawar-merah	34.4	Adu-buyung
10.	45.4	Adu-buyung	34.9	mbah-dukun
11.	45.9	Jatuh-bangun-Aku	34.9	Mabuk-Janda
12.	46.8	mbah-dukun	36.2	Kuch-Kuch-Hotahei
13.	48.6	Jablai	37.3	Jatuh-bangun-Aku
14.	49.6	Darah-Muda	38.1	Penasaran
15.	59.1	Bang-toyib	42.5	Bang-toyib

Pada Tabel 4.4 diatas terlihat bahwa perhitungan jarak dengan menggunakan tipe 1 memberikan hasil yang memiliki selisih cukup besar, sedangkan pada tipe 3 selisihnya lebih kecil. Hal ini dikarenakan *step size* pada tipe 3 lebih banyak sehingga toleransi perhitungan jarak akan semakin besar. Akibatnya, hasil perhitungan jarak akan mendekati satu sama lain. Terlihat pada tabel diatas lagu “Kopi Dangdut” menduduki peringkat pertama apabila menggunakan tipe 1, namun apabila menggunakan tipe 3 akan menduduki peringkat keempat.

Berdasarkan alasan-alasan tersebut, maka tipe DTW 1 akan memberikan hasil MRR yang paling maksimal apabila dibandingkan dengan tipe yang lain.

4.2.2 Analisis *Gulley*

Berdasarkan Gambar 4.4 terlihat bahwa setiap tipe DTW akan memiliki satu nilai *gulley* yang akan memberikan nilai MRR tertinggi. Pada tipe DTW 1 dan 3 nilai MRR tertinggi didapatkan pada saat *gulley* bernilai 0,3, sedangkan pada tipe 2 adalah pada saat *gulley* bernilai 0,2. Apabila nilai *gulley* melebihi atau kurang dari nilai tersebut maka akan memberikan nilai MRR yang lebih kecil.

Gulley merupakan perbandingan antara daerah yang terdeteksi dengan dimensi dari *cost matrix*. Sehingga dapat dikatakan *gulley* berkaitan erat dengan perbedaan panjang melodi dan tempo pada *database* dan *query*. Pada simulasi ini panjang melodi dari *database* selalu lebih panjang dibandingkan *query* karena durasi dari *database* adalah keseluruhan lagu, sedangkan *query* hanya 10 detik. Oleh karena itu, melodi pada *database* harus dipotong sebelum dibandingkan dengan *query*.

Pemotongan jumlah melodi pada *database* belum sepenuhnya menunjukkan potongan lagu pada *database* dan *query* memiliki persamaan persepsi melodi. Hal ini dikarenakan perbedaan proses pembentukan melodi pada *query* dan *database*. Pada *database* sangat dimungkinkan untuk terjadi perulangan *pitch*, sedangkan pada *query* tidak dimungkinkan terjadinya perulangan *pitch* karena proses pembentukannya menggunakan *note segmentation by pitch*. Selain itu, juga terjadi perbedaan tempo antara *query* dan *database*. Berikut ini adalah contoh persepsi melodi yang tidak sama meskipun jumlah melodinya sama.

Relative pitch untuk *query* lagu “kopi dangdut” :

3.4741 0.4877 -1.4237 0.9360 0 -0.4743 -1.7772 -0.8246 3.0761
-2.6687 -2.6687 1.4743 2.0391 1.8240 0.4877

Relative pitch untuk *database* lagu “kopi dangdut”

0 2 1 0 0 0 0 0 0
-1 -2 3 -3 0 0

Pada urutan *relative pitch* diatas terlihat bahwa melodi pada *query* lebih lambat dibandingkan dengan *database*. Hal ini dikarenakan pada *database* terjadi banyak perulangan *pitch* (nilai 0 pada *relative pitch*) sedangkan pada *query* tidak.

Selain faktor diatas, faktor tempo juga sangat berpengaruh terhadap persepsi melodi. Terlihat pada contoh diatas *query* memiliki tempo yang lebih cepat. Terlihat bahwa *relative pitch* yang memiliki kemiripan, yaitu -2,6687 berada pada urutan kesepuluh sedangkan pada *database* -3 berada pada urutan ketigabelas.

Berdasarkan analisis diatas terlihat bahwa *gulley* akan menjadi parameter penentu yang memberikan hasil maksimal. Besarnya nilai *gulley* yang disimulasikan adalah dari 0 sampai 0.5. Nilai *gulley* menunjukkan selisih persepsi melodi antara *query* dan *database* yang dipengaruhi oleh proses *note segmentation* dan perbedaan tempo.

4.2.3 Analisis MRR Tertinggi

Berdasarkan simulasi yang dilakukan didapatkan hasil MRR tertinggi untuk percobaan 60 *query* adalah sebesar 0,617. Spesifikasi DTW untuk menghasilkan nilai MRR tersebut adalah dengan menggunakan tipe 1 dan nilai *gulley* 0,3. Tipe 1 memberikan nilai tertinggi karena memiliki *step size* (n,m-1) dan (n-1,m). Selain itu pada tipe 1 jumlah *step size* nya juga tidak sebanyak tipe 3. Nilai *gulley* 0.3 menunjukkan bahwa perbedaan persepsi melodi *query* rata-rata adalah 0,3 kali lebih lambat atau 0,3 kali lebih cepat dibadingkan dengan *database*.

4.2.4 Analisis Pengaruh Jenis Kelamin Sampel Terhadap MRR

Berdasarkan Tabel 4.1, 4.2 dan 4.3 terlihat bahwa jenis kelamin berpengaruh terhadap hasil MRR. Terlihat bahwa sampel dengan jenis kelamin wanita memiliki nilai MRR yang lebih kecil dibandingkan dengan sampel laki-laki.

Setelah membandingkan hasil MRR antara sampel laki-laki dan wanita, terlihat bahwa penyebab perbedaan nilai MRR itu tidak disebabkan oleh tipe DTW atau pun *gulley*. Melainkan lebih disebabkan karena proses *note segmentation by pitch*. Hasil *note* segmentation untuk *query* yang berasal dari

sampel laki-laki dan perempuan mengalami perbedaan yang cukup besar, terutama berkaitan dengan jumlah *pitch* yang dihasilkan.

Sebagai contoh pada lagu "Jablai" seorang sampel laki-laki dapat menghasilkan 27 *pitch* untuk satu melodinya, sedangkan seorang sampel wanita menghasilkan 55 *pitch*. Akibatnya, tingkat akurasi dari pengenalan menjadi berkurang. Hal ini terlihat dari peringkat lagu "Jablai" yang jika dinyanyikan oleh sampel laki-laki dapat menduduki peringkat satu, tetapi jika dinyanyikan oleh sampel wanita hanya menduduki peringkat enam.

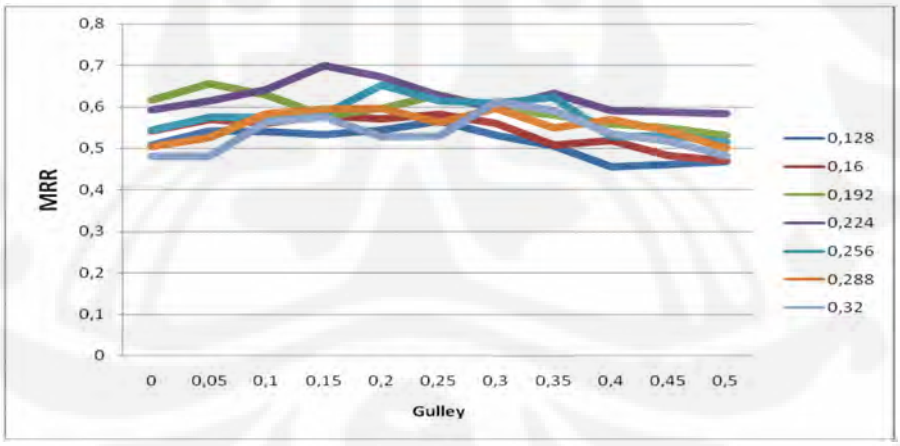
Jumlah *pitch* yang lebih banyak untuk sampel wanita disebabkan karena karakteristik dasar suara wanita yang cenderung memiliki frekuensi lebih tinggi. Frekuensi yang lebih tinggi ini akan cenderung sulit untuk dipertahankan. Akibatnya *pitch vector* yang terbentuk menjadi tidak stabil. *Pitch vector* yang tidak stabil ini akan menyebabkan proses *note segmentation* menghasilkan *pitch* yang tidak sesuai. *Pitch vector* yang tidak stabil tersebut akan membentuk *pitch* baru, sehingga secara kesuruhan jumlah *pitch* yang dihasilkan menjadi lebih banyak dari seharusnya.

Salah satu cara untuk mengatasi masalah tersebut adalah dengan menambahkan besar *threshold* durasi dari *pitch*/*note*. Dengan menambahkan nilai *threshold* durasi tersebut maka *pitch vector* yang tidak stabil tersebut menjadi sulit untuk membentuk *pitch* baru. Namun penambahan besar *threshold* ini juga tidak boleh terlalu besar, karena apabila terlalu besar akan menyebabkan ada *pitch* yang terlewati.

Pada bab sebelumnya telah disebutkan bahwa besar dari *threshold* durasi adalah sebesar 0,128 detik atau setara dengan 4 *pitch vector*. Artinya adalah dibutuhkan minimal 4 *pitch vector* dengan nilai berdekatan untuk membetuk satu *pitch*. Apabila jumlah *pitch vector* kurang maka tidak akan terbentuk *pitch*. Selanjutnya dilakukan simulasi untuk meningkatkan kualitas MRR untuk sampel wanita dengan cara meningkatkan besar *threshold* durasi tersebut. Untuk simulasi ini digunakan tipe DTW 1, karena tipe DTW 1 memberikan hasil MRR yang tertinggi pada simulasi sebelumnya. Tabel 4.5 menunjukkan pengaruh kenaikan *threshold* durasi terhadap MRR pada tipe DTW 1.

Tabel 4.5 Pengaruh Kenaikan *Threshold* Durasi terhadap MRR

Gulley	MRR						
	0,128 (4 pv)	**0,16 (5 pv)**	**0,192 (6 pv)**	**0,224 (7 pv)**	**0,256 (8 pv)**	**0,288 (9 pv)**	**0,32 (10 pv)**
0	0,508	0,5414	0,6156	0,5927	0,5442	0,5041	0,4811
0,05	0,541	0,5684	0,6553	0,6154	0,5751	0,5246	0,4811
0,1	0,54	0,5595	0,6283	0,642	0,5747	0,5827	0,5638
0,15	0,532	0,5787	0,5786	0,6989	0,5814	0,5943	0,5751
0,2	0,544	0,5708	0,5967	0,6729	0,6531	0,5955	0,5284
0,25	0,565	0,5817	0,6278	0,6269	0,6147	0,5613	0,5298
0,3	0,531	0,5595	0,5959	0,5964	0,6089	0,5959	0,6123
0,35	0,506	0,507	0,5801	0,6334	0,6247	0,5494	0,5912
0,4	0,456	0,5175	0,5587	0,5922	0,5281	0,5686	0,5356
0,45	0,462	0,4829	0,5487	0,5873	0,5307	0,5408	0,5168
0,5	0,469	0,4695	0,5309	0,5842	0,5148	0,5001	0,4821

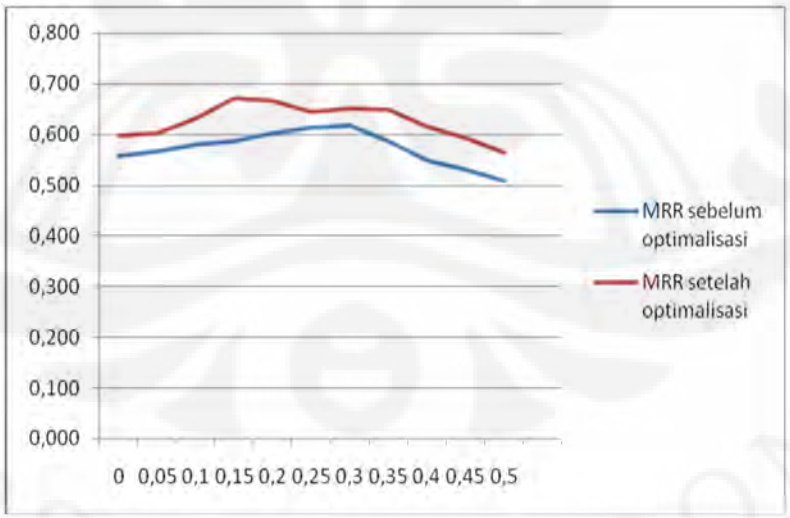


Gambar 4.6 Grafik Pengaruh Kenaikan *Threshold* Durasi terhadap MRR

Berdasarkan Gambar 4.6 terlihat bahwa nilai *threshold* sebesar 0.224 memberikan hasil yang paling maksimum. Kemudian didapatkan peningkatan nilai MRR secara keseluruhan ditunjukkan oleh Tabel 4.6.

Tabel 4.6 Perbandingan Total MRR setelah Optimalisasi *Threshold* Durasi

Gulley	MRR				
	Laki-Laki	Wanita (4 pv)	wanita (7 pv)	Total (dengan wanita 4 pv)	Total (dengan wanita 7 pv)
0	0,605	0,508	0,593	0,557	0,599
0,05	0,591	0,541	0,615	0,566	0,603
0,1	0,618	0,540	0,642	0,579	0,630
0,15	0,641	0,532	0,699	0,587	**0,670**
0,2	0,657	0,544	0,673	0,600	0,665
0,25	0,660	0,565	0,627	0,613	0,644
0,3	0,703	0,531	0,596	**0,617**	0,650
0,35	0,664	0,506	0,633	0,585	0,649
0,4	0,639	0,456	0,592	0,548	0,615
0,45	0,598	0,462	0,587	0,530	0,593
0,5	0,546	0,469	0,584	0,508	0,565



Gambar 4.7 Grafik MRR Sebelum dan Sesudah Optimalisasi

Terlihat pada Gambar 4.7, MRR tertinggi mengalami peningkatan dari 0,617 (untuk *gulley* 0,3) menjadi 0,67 (untuk *gulley* 0,15) dengan menggunakan tipe DTW 1.

BAB 5
KESIMPULAN

Berdasarkan hasil uji coba simulasi perangkat lunak pada sistem QbSH dengan menggunakan DTW yang telah dilakukan dalam percobaan ini, maka dapat diambil beberapa kesimpulan, yaitu:

1. Tipe DTW 1 merupakan tipe DTW yang memberikan hasil MRR maksimal untuk *database* musik dangdut yang digunakan pada simulasi ini.
2. Nilai *gulley* yang memberikan hasil MRR paling maksimal untuk *database* musik dangdut yang digunakan pada simulasi ini adalah dari nilai 0,15 sampai 0,4.
3. Berdasarkan simulasi yang dilakukan didapatkan nilai MRR tertinggi sebesar 0,617 dengan menggunakan tipe DTW 1 dan *gulley* sebesar 0,3.
4. Jenis Kelamin dari sampel mempengaruhi hasil MRR. Sampel wanita memiliki nilai yang lebih rendah dibandingkan dengan sampel laki-laki.
5. Nilai MRR untuk sampel wanita dapat ditingkatkan dengan mengatur *threshold* durasi pada proses *note segmentation*. Sehingga didapatkan peningkatan nilai MRR menjadi 0,67 dengan menggunakan tipe DTW 1 dan *gulley* sebesar 0,15.

DAFTAR REFERENSI

[1] Wei Chai. (2001). *Melody retrieval on the web*. Thesis. Massachusetts Institute of Technology. diakses November 16, 2009 dari http://alumni.media.mit.edu/~chaiwei/papers/msthesis.pdf

[2] I-Yang Lee. (1999). *Content-based music retrieval from acoustic input*. Thesis. Department of Computer Science National Tsing Hua University. diakses November 20, 2009 dari http://wayne.cs.nthu.edu.tw/~window/paper/thesis/cvgip99.**doc**

[3] Durrieu, Jean-Louis. (2006). *Music information retrieval a query by humming system, segmentation of the song and aproximative melody matching based on the DTW algorithm*. diakses November 16, 2009 dari http://www.tsi.enst.fr/~**durrieu**/publis/**durrieu**_a_**QbH**_system.**pdf**

[4] 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/

[5] Danneberg, Roger. B., et al. (2007). *A comparative evaluation of search techniques for query-by-humming using the MUSART testbed*. *Journal of the American Society for Information Science and Technology*, vol. 58, no. 2, pp.687–701, diakses November 20, 2009 dari http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.4028&rep=rep1&type=pdf

[6] Kusuma, Wahyu & Irwan Arifin. (2006). *Analisis penyamaan pitch interval sinyal melodi senandung untuk pencarian lagu*. Seminar Ilmiah Nasional Komputer dan Sistem Inteligen (KOMMIT 2006). diakses Desember 2, 2009 dari http://repository.gunadarma.ac.id:8000/Representasi_Chord_Wahyu_Kusuma_dkk_edit_835.pdf

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

[8] Muller, Meinard. (2007). *Information retrieval for music and motion*. Berlin: Springer.

[9] Dickerson, Kyle Britton.(2009) *Musical query-by-content using self-organizing maps*. Thesis. Brigham Young University. diakses November 20, 2009 dari http://contentdm.lib.byu.edu/ETD/image/etd2995.pdf

[10] Djiangnarto. Gonawan. (2005). *Perancangan pembuatan perangkat lunak pengenalan suara untuk mengoperasikan suatu program dalam windows*. Skripsi. Universitas Kristen Petra Surabaya. diakses Desember 12, 2009. dari http://digilib.petra.ac.id/viewer.php?submit.x=0&submit.y=0&submit=prev&page=1&qual=high&submitval=prev&fname=%2Fjiunkpe%2Fs1%2Finfo%2F2005%2Fjiunkpe-ns-s1-2005-26402028-7608-kenal_suara-cover.pdf

[11] Yaniv, Ran & David Burshtein. *An Enhanced Dynamic Time Warping Model for Improve Estimation of DTW Parameters*. IEEE Transaction on Speech and Audio Signal Processing, Vol. 11, No. 3, May 2003.

[12] Lie Lu, & Frank Seide. (2008). *Mobile ringtone search through query by humming.* diakses November 16, 2009 dari http://thamakau.usc.edu/Proceedings/ICASSP%202008/HTML/SessionIndex.**pdf**

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

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

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

[16] Jyh-Shing Roger Jang & Hong-Ru Lee. *A General Framework of Progressive Filtering and Its Application to Query by Singing/Humming*. IEEE Transaction on Audio, Speech, and Language Processing, Vol. 16, No. 2, February 2008.

[17] Schutte, Ken.”MATLAB and MIDI”, available from the link at author’s homepage at” http://www.kenschutte.com/midi”.

[18] Ellis, Dan. “Dynamic Time Warp in Matlab”, available from the link at author’s homepage at” http://labrosa.ee.columbia.edu/matlab/dtw/”.

LAMPIRAN

Berikut ini adalah *source code* dari *software* simulasi QbSH yang dibuat atau dimodifikasi oleh penulis. Untuk *source code* fungsi-fungsi pendukung dapat diakses langsung pada *website* yang tertera pada daftar referensi.

1. GUI dari Program Utama QbSH

QbSH_DTW_GUI1.m

```
function varargout = QbSH_DTW_gui1(varargin)
%ini adalah GUI dari program QBSH dibuat dengan bantuan GUIDE dari
%matlab
%by Teddy Febrianto
% QBSH_DTW_GUI1 M-file for QbSH_DTW_gui1.fig

% Last Modified by GUIDE v2.5 13-Jun-2010 12:33:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @QbSH_DTW_gui1_OpeningFcn, ...
                   'gui_OutputFcn',  @QbSH_DTW_gui1_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 QbSH_DTW_gui1 is made visible.
function QbSH_DTW_gui1_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes QbSH_DTW_gui1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = QbSH_DTW_gui1_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
% --- Executes on button press in liverecord.
```

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

function wavfile_Callback(hObject, eventdata, handles)

function wavfile_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in run.
function run_Callback(hObject, eventdata, handles)

% Main File QBSH
% modified from
% Roger Jang, 20091007
%by Teddy Febrianto

% ====== Membuka Database
fprintf('Membuka database...');
tic
databasefile=get(handles.databasetag,'String');
load(databasefile);
gulley1=get(handles.gulleytag, 'String');
gulley=str2num(gulley1);

distance=1; %menggunakan 1. manhattan distance dan 2. untuk
euclidean distan
fprintf(' ===> %.2f sec\n', toc);
% ====== men-Set Parameter untuk QBSH
subtaskId=get(handles.tipedtw,'Value');
qbshParam=qbshParamSet(subtaskId);
% ====== format dari data input (menggunakan perekaman langsung
atau tidak
liveRecording=get(handles.liverecord,'Value');
% ====== format dari output (berapa jumlah lagu yang akan
ditampilkan pada
% GUI
outputSongNum=15;
% ====== men-Set Parameter yang digunakan untuk perekaman
duration=10;
fs=8000;
nbits=8;


    % ====== Perekaman menggunakan MATLAB

    if liveRecording
        fprintf('Bersiap-siap untuk perekaman selama %d-detik. ',
duration);
        set(handles.run, 'String','siap-siap');
        pause (1);fprintf('\n 3');
        set(handles.run, 'String','3');
```

```
        pause (1);fprintf('\n 2');
        set(handles.run, 'String','2');
        pause (1);fprintf('\n 1');set(handles.run, 'String','1');
        pause (1);set(handles.run, 'String','Mulai');pause(0.1);
        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('Proses perekaman selesai (%s).\n', waveFile);
        set(handles.run, 'String','Selesai');
        pause(0.3);
        set(handles.run, 'String','RUN');
        pause(0.1);
    else
        waveFile=get(handles.wavfile,'String');
    end
    % ====== Proses Pitch tracking
    fprintf(' Proses Pitch tracking untuk query');
    tic;
    pitch=pitchTracking(waveFile, qbshParam.pt, 0);
    fprintf(' ===> %.2f sec\n', toc);
    % ====== Proses matching
    fprintf('Membandingkan query dengan %d lagu pada database',
length(songDb));
    tic;
    [distVec, minIndexVec, scoreVec] =
melodyCompare4plusgulley(songDb, pitch, qbshParam.mr,gulley);
    fprintf(' ===> %.2f sec\n', toc);
    % ====== Menampilkan hasil
    [junk, id4sort]=sort(distVec);
    rankprint='Hasilnya adalah sebagai berikut:';
    for i=1:outputSongNum
        index=id4sort(i);
        songName=songDb(index).songName;
        items=split(songName, '_');
        if length(items)>1, songName=items{1}; end
        rankprint= sprintf('%s \n %d %s',rankprint, i,  songName);
        set(handles.rank1,'string', rankprint);
    end
    fprintf('\n');
    pause(0.1);


%end
%==========================================================================
======
%==========================================================================
======

function tipedtw_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
```

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

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all
properties.
function rank1_CreateFcn(hObject, eventdata, handles)

function edit3_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function edit3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function gulleytag_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function gulleytag_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function disntancetag_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function disntancetag_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function distancetag_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function distancetag_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end
```

2. Fungsi qbshParamSet

qbshParamSet.m

```
function qbshParam=qbshParamSet(subtaskId)
% qbshParamSet: Set up parameters for QBSH
%   Usage: qbshParam=qbshParamSet(subtaskId, anchorPos)
%       subTaskId: 1 for subtask 1, 2 for subtask 2, dst of QBSH
%
% modified from
%   Roger Jang, 20090925
%by Teddy Febrianto


% ====== Parameter untuk melody recognition
qbshParam.mr=mrParamSet;    % default parameters

% ====== Parameter untuk pitch tracking
qbshParam.pt=ptParamSet(8000, 8);

% ====== parameter untuk berbagai subtaksId yang berbeda
switch subtaskId
    case 1
        qbshParam.mr.method='dtwc1'; %metode DTW 1
        qbshParam.mr.useRest=0; %tipe resthandle
        qbshParam.mr.comparisonType='wave2midi';
        qbshParam.pt.clarityThresholding=0;
        qbshParam.pt.mainFunc='wave2pitchByAcf'; %metode pitch
tracking
    case 2
        qbshParam.mr.method='dtwc2'; %metode DTW 2
        qbshParam.mr.useRest=0; %tipe resthandle
        qbshParam.mr.comparisonType='wave2midi';
        qbshParam.pt.clarityThresholding=0;
        qbshParam.pt.mainFunc='wave2pitchByAcf'; %metode pitch
tracking
    case 3
        qbshParam.mr.method='dtwc3'; %metode DTW 3
        qbshParam.mr.useRest=0; %tipe resthandle
        qbshParam.mr.comparisonType='wave2midi';
        qbshParam.pt.clarityThresholding=0;
        qbshParam.pt.mainFunc='wave2pitchByAcf'; %metode pitch
tracking
end
```

3. Fungsi Melody Compare

melodyCompare4plusgulley.m

```
function [distVec, minIndexVec, scoreVec] =
melodyCompare4plusgulley(songDb, pitch, mrParam,gulley,distance)
%modified from Roger Jang
%by Teddy Febrianto
```

```
pitch=pitch(:)';                % mentranspose matrix pitch
timeUnit=256/8000;              %besar frame 256 sample point
pitchTh=0.55;
minNoteDuration=0.13;           %besar threshold durasi untuk note
segmentation
pitch=restHandle(pitch, mrParam.useRest);       % melakukan fungsi
resthandle
pitch=pv2note(pitch,timeUnit, pitchTh, minNoteDuration,
0); %melakukan note segmentation
pitch=pitch(1:2:end)
pitch=pitch2contourm(pitch)         % merubah pitch menjadi
relative pitch
pitchLen = length(pitch);           % Length of the pitch vector
[- µ°ª¦V¶qªø«×¡]ÂI¼Æ¡^]
maxSongLen = round(mrParam.lengthRatio*pitchLen);   % The song
length is equal to mrParam.lengthRatio times the pitch length
[¼Ð·Ç°q¦±ªø«×³Ì¦h¥u¦³¿é¤J- µ°ª¦V¶qªø«×ªº mrParam.lengthRatio - ¿]
pitch=pitch(1:mrParam.pvrr:end);            % Down-sample to
reduce computation [- °ÂI¥H´î§C¹Bºâ¶q]
songNum=length(songDb);
distVec=zeros(1, songNum);
minIndexVec=zeros(1, songNum);

switch mrParam.method
  case 'dtwc1'
      C=[1 1 1;1 0 1;0 1 1];%mendefinisikan step size dan local
weight untuk tipe DTW 1
        for i=1:length(songDb),
            song=songDb(i).note(1:2:end);
            song=song';
            % ====== Mengambil lagu dari database
            song=restHandle(song, mrParam.useRest); % melakukan
fungsi resthandle
            song=pitch2contourm(song); %mengubah pitch menjadi
relative pitch
            songLen = length(song);
            song=song(1:min(pitchLen,songLen)); %melakukan
pemotongan jumlah pitch dari lagu pada database
            songLen = length(song);
            song=song(1:mrParam.pvrr:end);
            % ====== Perhitungan jarak dengan menggunakan DTW
            Mq=0;
            G=gulley;
            Mq=costmat(pitch,song);    % membuat cost matrix
            [p,q,D,sc,dist] = dpfast1(Mq, C,0 , G); %melakukan
perhitungan DTW
            distVec(i)=dist;
        end
     case 'dtwc2'
      C=[1 1 1;1 2 2;2 1 2]; %mendefinisikan step size dan local
weight untuk tipe DTW 2
        for i=1:length(songDb),
            song=songDb(i).note(1:2:end);
            song=song';
```

```
            % ====== Menambil lagu dari database
            song=restHandle(song, mrParam.useRest); % melakukan
fungsi resthandle
            song=pitch2contourm(song); %mengubah pitch menjadi
relative pitch
            songLen = length(song);
            song=song(1:min(pitchLen,songLen)); %melakukan
pemotongan jumlah pitch dari lagu pada database
            songLen = length(song);
            song=song(1:mrParam.pvrr:end);
            % ====== Perhitungan jarak dengan menggunakan DTW
            Mq=0;
            G=gulley;
            Mq=costmat(pitch,song);  % membuat cost matrix
            [p,q,D,sc,dist] = dpfast1(Mq, C, 0 , G);%melakukan
perhitungan DTW
            distVec(i)=dist;
        end
    case 'dtwc3'
      C=[1 1 1;1 0 1;0 1 1;1 2 2;2 1 2]; %mendefinisikan step size
dan local weight untuk tipe DTW 3
        for i=1:length(songDb),
%           fprintf('%d/%d\n', i, length(songDb));
            song=songDb(i).note(1:2:end);
            song=song';
            % ====== Menambil lagu dari database
            song=restHandle(song, mrParam.useRest);% melakukan
fungsi resthandle
            song=pitch2contourm(song);%mengubah pitch menjadi
relative pitch
            songLen = length(song);
            song=song(1:min(pitchLen,songLen)); %melakukan
pemotongan jumlah pitch dari lagu pada database
            songLen = length(song);
            song=song(1:mrParam.pvrr:end);
            % ====== Perhitungan jarak dengan menggunakan DTW
            Mq=0;
            G=gulley;
            Mq=costmat(pitch,song);% membuat cost matrix
            [p,q,D,sc,dist] = dpfast1(Mq, C, 0 , G);%melakukan
perhitungan DTW
            distVec(i)=dist;
        end

    otherwise
        error('Unknown method!');
end

% [a, b] = gBellParam(1.27, 0.95, 1.60, 0.5);
%scoreVec tidak di pakai untuk simulasi ini
a=1.60;
b=6.37;
scoreVec=100./(1+(distVec/a).^(2*b));
```

4. Fungsi untuk Merubah Pitch Menjadi Relative Pitch

Pitch2contourm.m

```
function contour=pitch2contourm(pitch)
%fungsi untuk merubah pitch menjadi relative pitch
%Created by : Teddy Febrianto
pitch1=pitch;
pitch2=circshift(pitch,[0,1]);
contour=pitch1-pitch2;
contour=circshift(contour,[0,-1]);
```

5. Fungsi untuk Menghitung Cost Matrix

costmat.m

```
function d=costmat(t,r);
%fungsi untuk menghitung cost matrix dari dua buat vektor
% Created by: Teddy Febrianto
[rows,N]=size(t);
[rows,M]=size(r);
%manhattan distance
d=abs((repmat(t(:),1,M)-repmat(r(:)',N,1)));
```

6. GUI dari Program Utama Pembuatan Database

createDBgui.m

```
function varargout = createDBgui(varargin)
%ini adalah GUI dari program QBSH dibuat dengan bantuan GUIDE dari
%matlab
%by Teddy Febrianto
% CREATEDBGUI M-file for createDBgui.fig

% Last Modified by GUIDE v2.5 29-May-2010 21:52:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @createDBgui_OpeningFcn, ...
                   'gui_OutputFcn',  @createDBgui_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 createDBgui is made visible.
function createDBgui_OpeningFcn(hObject, eventdata, handles,
varargin)
```

```
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

function varargout = createDBgui_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;

function musicfolder_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function musicfolder_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function filename_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all
properties.
function filename_CreateFcn(hObject, eventdata, handles)
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)

midiOrWaveList = get(handles.musicfolder,'String');
ptParam=ptParamset;
if exist(midiOrWaveList)==7 % This is in fact a directory
    midiData=recursiveFileList(midiOrWaveList, 'mid');
    waveData=recursiveFileList(midiOrWaveList, 'wav');
    if isempty(waveData)
        allData=midiData;
    elseif isempty(midiData)
        allData=waveData;
    else
        allData=[midiData, waveData];
    end
    contents={allData.path};
elseif exist(midiOrWaveList)==2     % A list of midi/wave files
    contents=textread(midiOrWaveList, '%s', 'delimiter', '\n',
'whitespace', '');
else
    error('The given file/folder does not exist!');
end

fileNum=length(contents);

for i=1:fileNum
```

```
    fileName=contents{i};
    fprintf('%d/%d: Adding "%s" to song database...\n', i, fileNum,
fileName);
    songDb(i).path=fileName;
    [parentDir, songDb(i).songName, songDb(i).fileType,
version]=fileparts(songDb(i).path);
    switch lower(songDb(i).fileType)
        case '.mid'
            songDb(i).note=midiFile2note(songDb(i).path);
        case '.wav';
            songDb(i).pv=pitchTracking(songDb(i).path, ptParam);
        otherwise
            fprintf(sprintf('Unknown file type: %s\n',
songDb(i).fileType));
    end
end

for i=1:length(songDb)
    if strcmp(songDb(i).fileType, '.mid')
        [songDb(i).pv,
songDb(i).noteStartIndex]=note2pv(songDb(i).note,
(ptParam.frameSize-ptParam.overlap)/ptParam.fs, inf, 0);
    end
end
fileSave=get(handles.filename,'String');
save(fileSave,'songDb');
```