Pengenalan Suara Menggunakan Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) Sebagai Ekstrasi Ciri

SKRIPSI

oleh:

MOHAMMAD MAHENDRA JAYA WARDANA

0610960046

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2011

Pengenalan Suara Menggunakan Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) Sebagai Ekstrasi Ciri

SKRIPSI

oleh:

MOHAMMAD MAHENDRA JAYA WARDANA

0610960046

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2011

LEMBAR PENGESAHAN SKRIPSI

Pengenalan Suara Menggunakan Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) Sebagai Ekstrasi Ciri

Oleh:
MOHAMMAD MAHENDRA JAYA WARDANA
0610960046-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 21 Maret 2011
dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Drs. Marji, MT	**Nurul Hidayat,SPd.,MSc**
NIP. 196708011992031001	**NIP. 196804302002121001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf A., MSc.
NIP. 196709071992031001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama : Mohammad Mahendra Jaya Wardana
NIM : 0610960046
Jurusan : Ilmu Komputer
Penulis Tugas Akhir berjudul : Pengenalan Suara Menggunakan Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) Sebagai Ekstrasi Ciri.

Dengan ini menyatakan bahwa :

1. **Isi dari tugas Akhir yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Tugas Akhir ini.**
2. **Apabila dikemudian hari ternyata Tugas Akhir yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 21 Maret 2011
Yang menyatakan,

(Mohammad Mahendra Jaya Wardana)
NIM. 0610960046

Pengenalan Suara Menggunakan Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) Sebagai Ekstrasi Ciri

ABSTRAK

Pengenalan suara adalah teknologi masa depan yang menggantikan cara interaksi manusia dengan komputer dengan menggabungkan beberapa disiplin ilmu seperti pengenalan sinyal dan pengenalan pola dimana interaksi user dengan sistem dapat dilakukan dengan memberikan inputan suara.

Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) sebagai penyimpan karakteristik dari suatu sinyal suara setelah dari proses MFCC dilakukan proses pemetaan oleh *vector quantization* (VQ) yang disebut *cluster* dimana *cluster* tersebut dikelompokkan dengan algoritma *K-mean* yang kemudian dicari dengan pengukuran jarak *euclidean distance* untuk melakukan perbandingan dari data input dengan data yang ada pada *database*.

Berdasarkan uji coba yang dilakukan terhadap 30 kata yang diucapkan oleh 3 orang laki-laki 3 orang perempuan, dihasilkan untuk data *learning* bahwa pada *codebook* 16 nilai *Word Error Rate* (WER) adalah 63.33%, *codebook* 32 nilai WER 50%, *codebook* 64 nilai WER 46.67%, *codebook* 128 nilai WER 40% dan pada *codebook* 256 didapat nilai WER 47%. Pada *codebook* 16 sampai 128 nilai WER semakin kecil, sedangkan pada *codebook* 256 nilai WER mengalami peningkatan. Sedangkan untuk data *non-learning* diketahui bahwa pada *codebook* 16 nilai WER adalah 50%, *codebook* 32 nilai WER 40%, *codebook* 64 nilai WER 30%, *codebook* 128 nilai WER 23.33% dan pada *codebook* 256 didapat nilai WER 26.67%. Sehingga dapat disimpulkan bahwa semakin besar jumlah *codebook* tidak menjamin dapat meningkatkan keakurasian sistem.

Introduction to Sound (Speaker Recognition) Using Fast Fourier Transform algorithm (FFT) Algorithm With Mel Frequency cepstrum coefficient (MFCC) For Feature Extraction

ABSTRACT

Voice recognition is the future technology that replaces human interaction with computers by combining several disciplines such as signal recognition and pattern recognition. Where the user's interaction with the system can be done by giving a voice input.

Algorithm Mel Frequency cepstrum coefficient (MFCC) as the capture Characteristics of a sound signal after the above process mapping process is carried out by vector quantization (VQ), called clusters where the cluster is diklompokkan with K-mean algorithm is then sought by measuring the distance euclidean distance to comparison of input data with existing data in the database.

Based on tests conducted on 30 words spoken by 3 male 3 female, produced for the data in the *codebook* 16 learning that the WER is 63.33%, *codebook* 32 WER value 50%, 64 *codebook* value of 46.67% WER, *codebook* 128 WER value of 40% and the *codebook* 256 obtained the value of 47% WER. In the *codebook* 16 to 128 WER value is shrinking, while in the *codebook* 256 values increased WER. As for the non-learning data in the *codebook* is known that 16 of WER is 50%, 32 *codebook* WER value 40%, 64 *codebook* WER value 30%, *codebook* wer 23:33% 128 value and the *codebook* 256 obtained value of 26.67% WER. It can be concluded that the results for voice recognition correctly with *codebook* size 16 with WER of 63.33%.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin., segala puji dan syukur ke hadirat Allah SWT atas segala rahmat dan hidayah yang telah diberikan-Nya, sehingga skripsi yang berjudul "Pengenalan Suara (S*peaker Recognition*) Menggunakan Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) Sebagai Ekstrasi Ciri" ini dapat diselesaikan. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.

Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW, makhluk paling mulia yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya dan sahabat-sahabatnya.

Dalam penyelesaian skripsi ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Drs. Marji, MT selaku pembimbing utama dan Nurul Hidayat, SPd., MSc selaku pembimbing pendamping dalam penulisan skripsi. Terima kasih atas semua waktu dan bimbingan yang telah diberikan.
2. Dr. Abdul Rouf A., MSc selaku Ketua Jurusan Matematika Universitas Brawijaya.
3. Drs. Marji, MT selaku Ketua Program Studi Ilmu Komputer Universitas Brawijaya.
4. Lailil Muflikhah, S.Kom, MSc selaku pembimbing akademik.
5. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
6. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya.
7. Orang tua dan keluarga besar penulis atas dukungan materi dan doa restunya kepada penulis.
8. Teman-teman seperjuangan Ilkom 2006 FMIPA UB yang telah banyak memberikan bantuan demi kelancaran penyusunan skrisi ini. Terima kasih atas semangat dan doanya.
9. Dan semua pihak yang tidak bisa penulis sebutkan satu per satu.

Akhirnya, penulis sadari bahwa masih banyak kekurangan dalam penyusuan skripsi ini dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca. Semoga skripsi ini dapat bermanfaat.

Malang, 21 Maret 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman

BAB I
PENDAHULUAN

1.1. Latar Belakang

Manusia merupakan makhluk sosial yang memerlukan komunikasi dengaan sesamanya dalam kehidupan sehari – hari. Suara merupakan salah satu media komunikasi yang paling sering dan umum digunakan oleh manusia. Manusia dapat memproduksi suara dengan mudah dan tanpa memerlukan energi yang besar(Petra Christian, 2003). Banyak persoalan yang terjadi ketika suara dimanfaatkan oleh sebuah sistem karena setiap orang memiliki ciri suara yang berbeda-beda. Suara merupakan modal utama yang dimiliki manusia untuk berkomunikasi dengan orang lain. Manusia dapat mengenali seseorang selain dari wajah (*face recognition*) juga dari suaranya (*voice recognition*). *Voice recognition* sendiri dibagi menjadi dua jenis, yaitu *speech recognition* dan *speaker recognition*. Berbeda dengan *speaker recognition* yang merupakan pengenalan identitas yang diklaim oleh seseorang dari suaranya (ciri khusus dapat berupa intonasi suara, tingkat kedalaman suara, dan sebagainya), *speech recognition* adalah proses yang dilakukan komputer untuk mengenali kata yang diucapkan oleh seseorang tanpa memperdulikan identitas orang terkait (Gresia Melisa, 2008). Dengan suara manusia dapat memberikan informasi maupun perintah. Salah satu teknologi yang memanfaatkan suara adalah proses sistem keamanan, *login* atau *password*. Dimana mengatasi hal tersebut digunakan proses pengenalan suara yang dikeluarkan oleh manusia.

Untuk dapat mengenali ucapan dari seseorang terlebih dahulu komputer diberi kecerdasan sehingga ucapan yang diterima oleh komputer akan direspons dan menanggapnya dengan sesuai. Ada banyak metode yang telah digunakan untuk pencocokan pola suara, diantaranya adalah Pencocokan Pola Suara (*Speech Recognition*) Dengan Algoritma FFT dan Devide Conquer (Gressia Melissa, 2008) dan Pencocokan Pola Suara (*Speech Recognition*) Dengan Algoritma FFT dan Devide Conquer (Nurlaily, 2009 dari kedua jurnal tersebut memiliki kesamaan yaitu transformasi untuk pembuka aplikasi pada windows seperti *microsoft word, excel, power point* sedangkan perbedaan dari skripsi pada penelitian ini bertujuan untuk mengenali siapa pemilik dari suara yang telah diinputkan.

Fast fourier transform (FFT) merupakan salah satu metode untuk transformasi sinyal suara menjadi sinyal frekuensi. Artinya proses perekaman suara disimpan dalam bentuk digital berupa *file WAV*.

Sedangkan implementasi algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) sebagai penyimpan karakterisrik dari suatu sinyal suara setelah dari proses diatas dilakukan proses pemetaan oleh *vector quantization* (VQ) yang disebut *cluster* dimana *cluster* tersebut dikelompokkan dengan algoritma *K-mean* yang kemudian dicari dengan pengukuran jarak *euclidean distance* untuk melakukan perbandingan dari data input dengan data yang ada pada *database*. Dari latar belakang yang telah dipaparkan, maka penulis mengambil judul pada skripsi ini " **Pengenalan Suara Menggunakan Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) Sebagai Ekstrasi Ciri ".**

1.2. Rumusan Masalah

Dalam penulisan ini permasalahan yang akan dibahas adalah :

1. Bagaimana mengimplementasikan algoritma FFT dengan MFCC sebagai ektrasi ciri dalam proses pengenalan suara
2. Berapa tingkat keberhasilan sistem.

1.3. Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah perencanaan dan pembuatan perangkat lunak dengan klasifikasi :

1. File yang diproses adalah *file* berekstensi *Waveform Audio* (*Wav*) 16.000kHz, 16bit, *Mono*.
2. Perintah suara dari pembicara didalam penelitian ini hanya mampu mengenali satu identitas pembicara.
3. Setiap identitas seseorang, perintah suara yang dibuat hanya memiliki satu kata.

1.4. Tujuan Penelitian

Tujuan dari penulisan tugas akhir ini adalah :

1. Merancang algoritma FFT dengan MFCC sebagai ekstrasi ciri dalam proses pengenalan suara yang kemudian disimpan pada *database*.
2. Menghitung tingkat keberhasilan sistem.

1.5. Manfaat Penelitian

Manfaat dari penulisan skripsi ini agar lebih lanjut dapat dimanfaatkan sebagai sistem pengaman, pemberian perintah komputer dan banyak lagi yang lainnya.

1.6. Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan tugas akhir ini adalah:

1. Studi Literatur
   Mempelajari teori yang berhubungan dengan metode FFT dari berbagai sumber.
2. Pendefinisian dan analisis masalah.
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat perancangan perangkat lunak dan mengimplementasikan hasilnya untuk membuat perangkat lunak tersebut.
4. Uji coba dan analisis hasil implementasi
   Menguji coba perangkat lunak tersebut dan menganalisa hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian dievalusi dan disempurnakan.

1.7. Sistematika Penulisan

Sistematika penulisan tugas akhir ini dibagi menjadi lima bab dengan masing-masing bab diuraikan sebagai berikut:

1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penulisan, manfaat penulisan, metodologi pemecahan masalah, dan sistematika penulisan.

2. BAB II TINJAUAN PUSTAKA
   Bab ini membahas mengenai teori-teori penunjang yang membahas konsep dasar dari citra, serta beberapa macam representasinya, serta beberapa teori FFT.

3. BAB III METODE DAN PERANCANGAN
   Bab ini membahas mengenai penggunaan teori-teori dalam perancangan perangkat lunak ini. Serta membahas bagaimana perangkat lunak ini direncanakan, dibuat dan diimplementasikan.

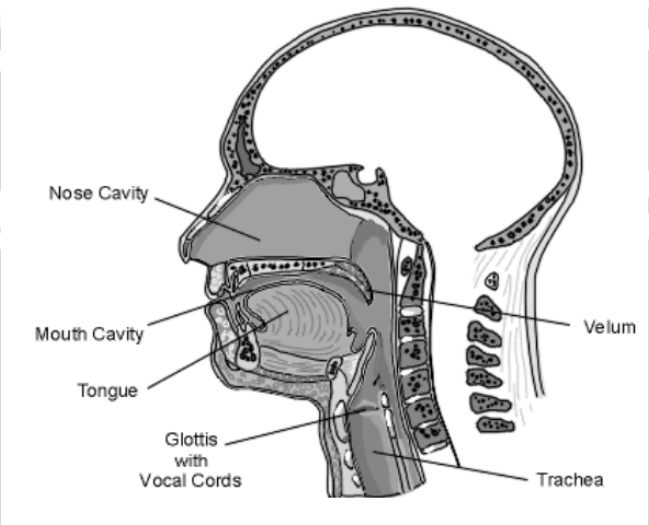
4. BAB IV HASIL DAN PEMBAHASAN
   Pada bab ini akan dilakukan implementasi sistem, serta analisa hasil yang dikeluarkan oleh perangkat lunak, untuk selanjutnya dilakukan analisa hasil untuk mengevaluasi kualitas pengenalan suara setelah mengalami proses pengenalan dengan algoritma FFT

5. BAB V PENUTUP
   Bab ini berisi kesimpulan dan saran dari proses pembuatan sampai proses pengimplementasian perangkat lunak serta untuk pengembangan perangkat lunak lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 Pembentukan Suara Manusia.

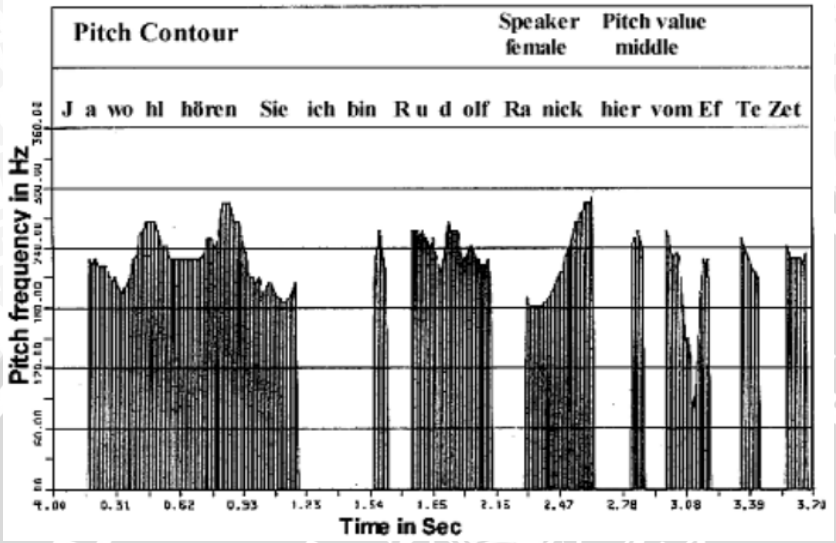
Speech (wicara) dihasilkan dari sebuah kerjasama antara *lungs* (paru-paru), dengan *vocal cords* (pita suara) dan *articulation tract* (*mouth*/mulut dan *nose cavity*/rongga hidung). Gambar 2.1 menunjukkan penampang melintang dari organ wicara manusia. Untuk menghasilkan sebuah *voiced sounds* (suara ucapan), paru-paru / *lungs* menekan udara melalui *epiglottis*(katup tenggorokan), *vocal cords*(pita suara) bergetar, menghirup udara melalui aliran udara dan menghasilkan sebuah gelombang tekanan *quasi-periodic*(kuasi berkala).



Gambar 2.1 Organ Suara Manusia.

Impuls tekanan pada umumnya disebut sebagai *pitch impulses* dan frekuensi sinyal tekanan adalah *pitch frequency* atau *fundamental frequency.* Di dalam Gambar 2.2 sederetan *impulse* (fungsi tekanan suara) dihasikan oleh *vocal cords* untuk sebuah suara. Ini merupakan bagian dari sinyal *voice* (suara) yang mendefinisikan *speech melody* (melodi wicara). Ketika berbicara dengan sebuah frekuensi *pitch* konstan, suara sinyal wicara *monotonous* tetapi dalam kasus normal sebuah perubahan permanen pada frekuensi terjadi. Variasi frekuensi *pitch* dapat dilihat seperti pada Gambar 2.3.

Gambar 2.2 Sedeeretan *Impulse* Yang Sama.

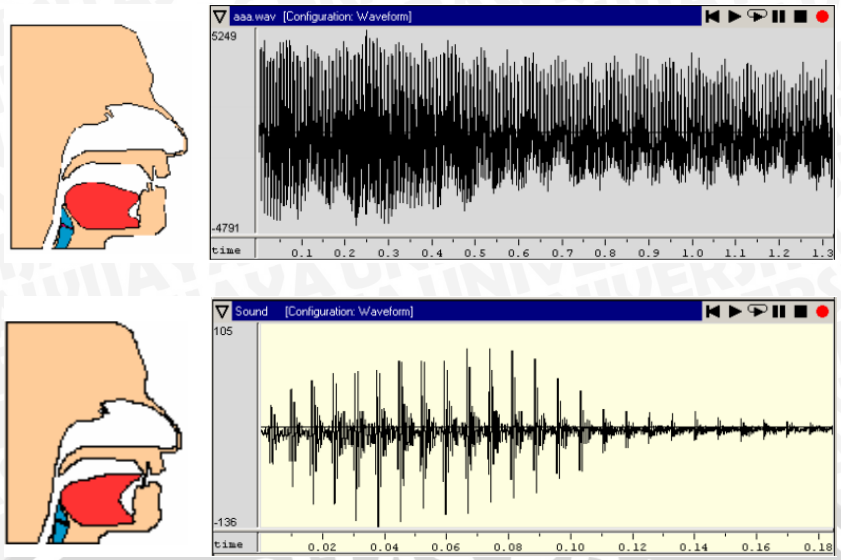


Gambar 2.3 Variasi pada frekuensi pitch.

Impuls *pitch* merangsang udara di dalam mulut, dan untuk suara tertentu (*nasals*) juga merangsang *nasal cavity* (rongga hidung). Ketika rongga beresonansi, akan menimbulkan radiasi sebuah gelombang suara yang mana merupakan sinyal suara. Kedua rongga beraksi sebagai *resonators* dengan karacteristik frekuensi resonansi masing-masing, yang disebut formant *frequencies*. Pada saat rongga mulut dapat mengalami perubahan besar, mampu untuk menghasilkan beragam pola ucapan suara yang berbeda.

Di dalam kasus *unvoiced sounds* (suara tak terucap), exitasi pada *vocal tract* lebih menyerupai *noise* (derau). Gambar 2.4 menampilkan proses produksi suara-suara /a/, dan /f/. Untuk sementara perbedaan bentuk dan posisi pada organ *articulation* diabaikan saja (Tri Budi Santoso, Miftahul Huda, 2010).

.



Gambar 2.4 Proses Produksi Suara /a/ dan /f/

2.2 Pengertian Suara.

Bunyi atau suara adalah kompresi mekanikal atau gelombang longitudinal yang merambat melalui medium. Medium atau zat perantara ini dapat berupa zat cair, padat, gas. Jadi, gelombang bunyi dapat merambat misalnya di dalam air, batu bara, atau udara.

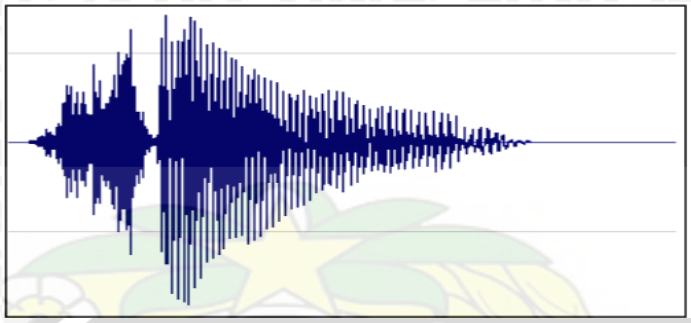
Kebanyakan suara adalah merupakan gabungan berbagai sinyal, tetapi suara murni secara teoritis dapat dijelaskan dengan kecepatan osilasi atau frekuensi yang diukur dalam Hertz (Hz) dan amplitudo atau kenyaringan bunyi dengan pengukuran dalam desibel.

Manusia mendengar bunyi saat gelombang bunyi, yaitu getaran di udara atau medium lain, sampai ke gendang telinga manusia. Batas frekuensi bunyi yang dapat didengar oleh telinga manusia kira-kira dari 20 Hz sampai 20 kHz pada amplitudo umum dengan berbagai variasi dalam kurva responsnya. Suara di atas 20 kHz disebut ultrasonik dan di bawah 20 Hz disebut infrasonik.(Anonymous,2010)

J.Fourier (1768-1830) menyatakan bahwa gelombang kompleks dapat direprentasikan sebagai penjumlahan dari sejumlah glelombang sinusoidal, dengan amplitudo dan frekuensi yang bervariasi. Dari teori *Fourier* dapat disimpulkan bahwa gelombang – gelombang sinusoidal,

yang memiliki amplitudo dan frekuensi berbeda – beda dinamakan deret *Fourier*.

Berikut ini adalah contoh Gambar 2.5 gelombang rekaman suara pada kata "*sorry*" yang dipergunakan untuk menjalankan aplikasi komputer yang di proses ole *software Sound Forge* 8.0.



Gambar 2.5 Rekaman kata "*sorry*".

2.3 Proses *Sampling*.

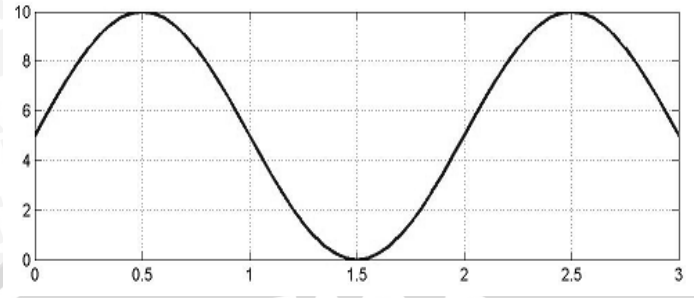
Sampling adalah suatu proses untuk membagi-bagi suatu sinyal kontinyu dalam interval waktu yang telah ditentukan. Sampling ini dilakukan dengan mengubah sinyal analog menjadi sinyal digital dalam fungsi waktu. Perubahan bentuk sinyal ini bertujuan untuk mempermudah memproses sinyal masukan yang berupa analog karena sinyal analog memiliki kepekaan terhadap *noise* yang rendah, sehingga sulit untuk memproses sinyal tersebut. Proses sampling berbeda – beda untuk setiap suara. Bila sampling terhadap suatu sinyal suara tidak akurat maka dapat terjadi *mis-leading* atau hasil yang tidak sesuai dengan aslinya.

Nyquist rate adalah rata-rata sampel minimum yang harus dipakai untuk mencegah timbulnya *aliasing* informasi frekuensi. Besarnya *Nyquist rate* harus dua kali frekuensi tertinggi dalam sinyal.

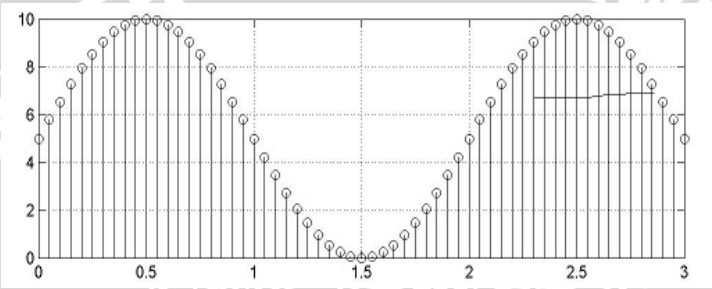
Proses *sampling* dilakukan dengan didasarkan pada asumsi bahwa sinyal percakapan berada pada daerah frekuensi 300 – 3400 Hz. Teori *sampling Nyquist* menyebutkan bahwa frekuensi *sampling* (*sampling rate*) minimal harus dua kali lebih tinggi dari frekuensi maksimum yang akan di_*sampling*. Dimana (f_s) frekuensi *sampling* (f_h) frekuensi sinyal analog tertinggi.

$$f_s \geq 2xf_h \quad (2.1)$$

Aliasing merupakan hasil dari *sampling* secara diskrit pada suatu sinyal yang terlalu rendah sehingga memberikan resolusi yang rendah pula. Gambar 2.7 menunjukkan *sample* sinyal 10 Hz yang nampak menjadi sinyal 5 Hz terjadi *aliasing* ditunjukkan pada Gambar 2.6 (Evi Andriani, 2010).



Gambar 2.6 Bentuk sinyal sinus



Gambar 2.7 Bentuk sinyal sinus yang telah di sampling

Sinyal suara disampling 16 Khz

Adalah sinyal suara yang telah di digitalisasi dengan menggunakan proses *sampling*. Sinyal ini dapat berupa file wav atau sejenisnya.

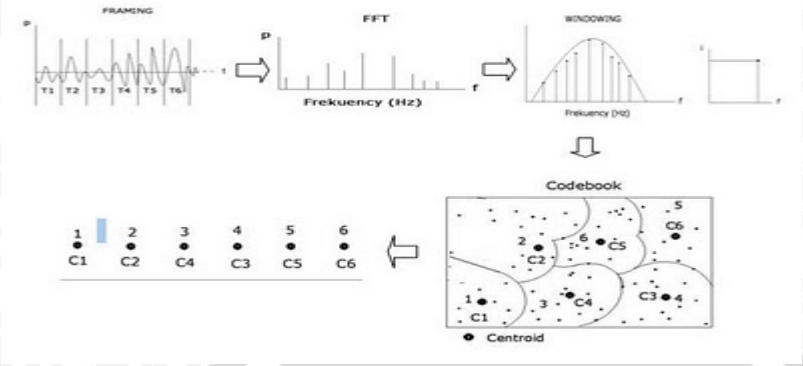
Dalam proses Hcopy dapat ditambahkan baris

SOURCERATE=625

Untuk menginisialisasi bahwa sinyal suara akan di_*sampling* dengan 16 KHz dimana 625 adalah *rate* data dalam *second*.

Ekstraks bertujuan untuk ekstraksi atau *feature extraction* ini adalah untuk mengubah bentuk getaran suara menjadi berbagai tipe

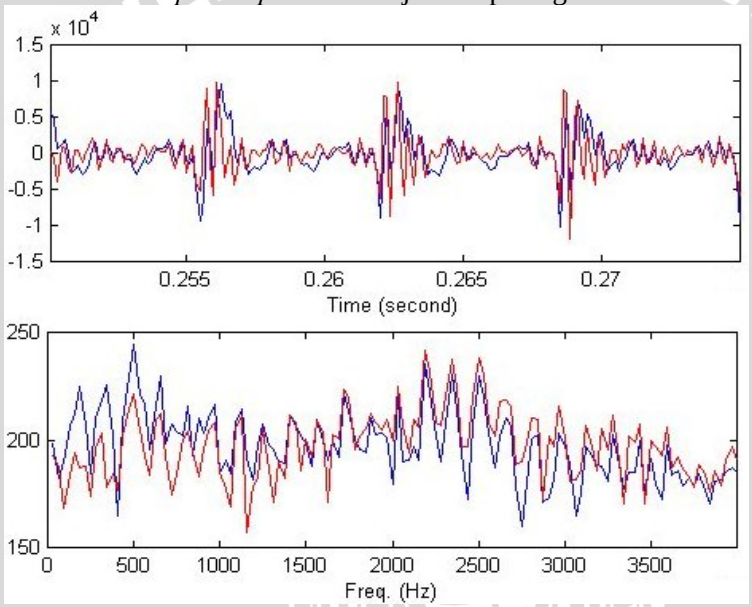
parameter yang merepresentasikan sinyal suara untuk dianalisa. Ekstraks *voice recognition* ditunjukkan pada gambar 2.8.



Gambar 2.8 Ekstraks *Voice Recognation.*

2.4 *Pre-Emphasis.*

pre-emphasis digunakan untuk mengambil *envelope* dari sinyal tersampel untuk memperhalus bentuk sinyal sinyal suara sebelum dan setelah sesudah di *pre-emphasis* ditunjukkan pada gambar 2.9.



Gambar 2.9 Sinyal suara telah di *Pre-Emphasis*

Pada gambar diatas ditunjukkan dengan garis merah adalah hasil proses *pre-emphasis* dimana kedua gambar diatas adalah proses *pre-emphasis* pada *domain* waktu dan *pre-emphasis* pada *domain* frekuensi. Syarat *pre-emphasis* ditunjukkan pada persamaan 2.1.

$$S'_n = S_n - \sigma S_{n-1} \quad (2.1)$$

where $0.96 \leq \sigma \leq 0.99$

Pada proses Hcopy dapat ditambahkan Baris
PREEMCOEF=0.97

Hal ini menunjukkan adanya *pre-emphasis*, dimana *coefisien pre-emphasis* berkisar antara *0.96-0.99*

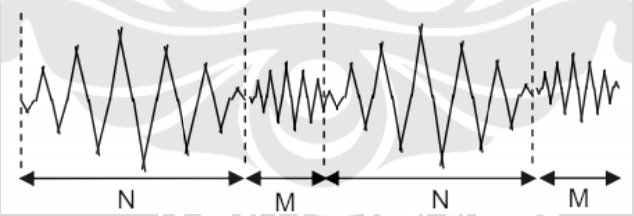
2.5 Ektrasi Gelombang.

Ekstrasi gelombang dilakukan untuk mendapatkan suatu gelombang yang dapat mewakili keseluruhan gelombang. Metode yang digunakan untuk proses ekstrasi meliputi beberapa tahap yaitu :

1. *Frame Blocking*
2. *Windowing*
3. *Fast Fourier Transform* (FFT)
4. *Mel Frequency Cepstrum Coeffisient*(MFCC)

2.5.1 *Frame Blocking.*

Frame Blocking adalah pembagian sinyal audio menjadi beberapa *frame*. Satu *frame* terdiri dari beberapa sampel tergantung tiap berapa detik suara akan disampel dan berapa besar frekuensi samplingnya. Pembagian *frame* tersebut menjadi beberapa N *frame* dengan adanya pemisahan antara *frame* yang satu dengan lainnya sebesar M *frame*. Pada gambar 2.9 terlihat proses *frame blocking*(Ario Muhammad Fanie, 2008).



Gambar 2.10 *Frame Blocking*

2.5.2 *Windowing.*

Tahapan *windowing* berfungsi untuk mengurangi efek diskontinuitas pada masing – masing *frame* yang telah diperoleh dari proses *frame blocking* sebelumnya. Proses ini akan menyebabkan sinyal ke nol pada permulaan dan akhir masing – masing *frame.* Metode yang digunakan dalam proses *windowing* untuk penelitian ini adalah *hamming window* dengaan persamaan 2.2 (Ario Muhammad Fanie, 2008).

$$w(n) = 0{,}54 + 0{,}46 \cos\left(\frac{2\pi n}{N}\right) \quad (2.2)$$

Dimana :

N = lebar *filter*

N =0.1, …….(N-1)/2 untuk N ganjil

=0.1, …….(N/2)-1 untuk N genap

Hasil dari proses *windowing* ini adalah berupa sesuatu sinyal yang bisa dilihat dari persamaan 2.3.

$$y_1(n) = x_1(n)\, w(n) \quad (2.3)$$

Dimana :

$y_1(n)$ = sinyal hasil *windowing*

x_1 = sinyal *input*

w(n) = besaran *windowing*

2.5.3 *Discrete Fourier Transform* (DFT).

DFT merupakan perluasan dari transformasi *Fourier* yang berlaku untuk sinyal – sinyal diskrit dengan panjang yang terhingga. Semua sinyal periodik terbentuk dari gabungan sinyal – sinyal *sinusoidal*.

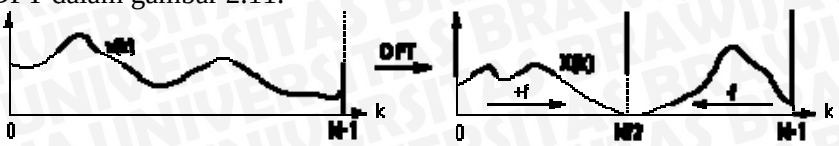
Pertimbangkan suatu rangkaian kompleks x(k) dengan N contoh format $x_0, x_1, x_2, x_3 \ldots x_k \ldots x_{N-1}$ dimana x adalah nomer komplek $x_i = x_{real} + j\, x_{imag}$ dari keterangan tersebut dapat dirumuskan seperti pada persamaan 2.4.

$$X(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(k)\, e^{-jk2\pi n/N} \quad \text{for } n=0..N-1 \quad (2.4)$$

Bisa juga persamaan dapat ditulis seperti pada persamaan2.5.

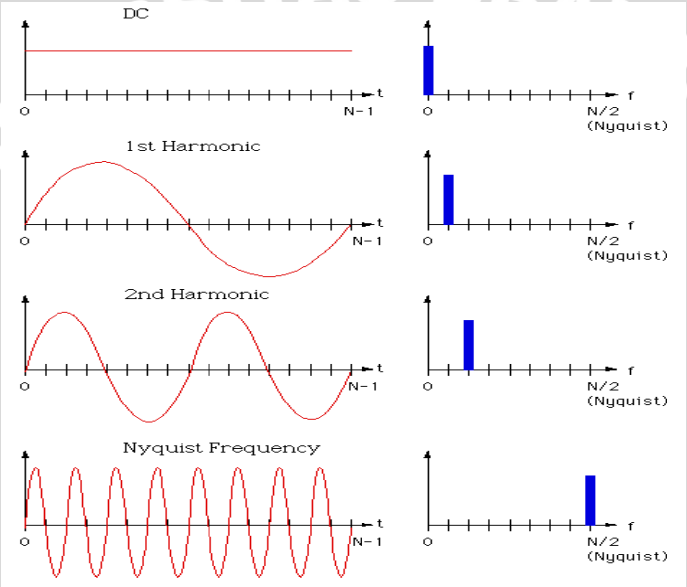
$$x(i) = \sum_{k=0}^{N-1} X(k)\, e^{jk2\pi n/N} \quad \text{for } n=0..N-1 \quad (2.5)$$

Dari persamaan 2.4 dan 2.5 maka daapat digambarkan prubahan pada DFT dalam gambar 2.11.



Gambar 2.11 Proses Perubahan Persamaan DFT

Dengan persamaan 2.4, suatu sinyal suara dalam *domain* waktu dapat dicari frekuensi pembentuknya. Analisa *fourier* pada data suara yaitu untuk mengubah data dari *domain* waktu menjadi data spektrum di *domain* frekuensi. Untuk pemrosesan sinyal suara, data pada *domain* frekuensi dapat diproses dengan lebih mudah dibandingkan data pada *domain* waktu, karena pada *domain* frekuensi keras lemahnya suara diabaikan atau tidak berpengaruh. Dari hubungan antara domai frekuensi dan waktu pada DFT ditunjukkan pada gambar 2.12.



Gambar 2.12 Hubungan DFT *domain* frekuensi dan waktu

Untuk mendapatkan spektrum dari sebuah sinyal dengan DFT diperlukan N buah sampel data berurutan pada *domain* waktu yaitu data x[n] sampai dengan x[n+N-1]. Apabila data tersebut dimasukkan

dalam fungsi DFT maka akan menghasilkan N buah data.(Elliot, D.F. and Rao, K.R. 1 9 8 2)

2.5.4 DFT dalam Bentuk Matriks.

Dengan menerapkan DFT ke FFT, format DFT dapat dinyatakan dalam bentuk matriks. Bentuk umum DFT dalam bentuk matrik dengan X_N (koefisien DFT merupakan matiksNx1), W_N (matriks NxN disebut matriks DFT) x_n(*Input* sinyal diskrit Nx1) dinyatakan dengan persamaan 2.6.

$$X_N = W_N . x_n \tag{2.6}$$

Untuk NxN bentuk matriksnya dapat dilihat sebagai berikut:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \cdots \\ \cdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & W_N^2 & \cdots & W^{N-1} \\ W_N^0 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ W_N^0 & W_N^{n-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \cdots \\ \cdots \\ x[N-1] \end{bmatrix}$$

2.5.5 *Fast Fourier Transform* (FFT).

Transformasi fourier adalah suatu metode yang sangat efisien untuk menyelesaikan transformasi *fourier* diskrit yang banyak dipakai untuk keperluan analisa sinyal seperti pemfilteran, analisa korelasi, dan analisa *spectrum*. Diskrit *Fourier* Transformasi (DFT) adalah deretan yang terdefinisi pada kawasan frekuensi – diskrit yang merepresentasikan transformasi *fourier* terhadap suatu deretan terhingga (*finite duration sequence*). DFT berperan penting untuk implementasi algoritma suatu varitas pengolahan sinyal, karena efisien untuk komputasi berbagai aplikasi.

Fast fourier Transformation atau transformasi *Fourier* cepat, merupakan proses lanjutan dari DFT (Diskrit Fourier Transformation). Transformasi Fourier ini dilakukan untuk mentransformaikan sinyal dari domain waktu ke domain frekuensi. Hal ini bertujuan agar sinyal dapat diproses dalam spectral substraksi. Rumus FFT ditunjukkan pada persamaan 2.7 (Nurlaily, 2009).

$$X[k] = \sum_{n=0}^{N-1} x(n) W_N^{kn} \tag{2.7}$$

Dimana $W_N = e^{-j2\pi/N}$ dari persamaan 2.7 dibagi menjadi bagian genap dan ganjil ditunjukkan pada persamaan 2.8.

$$X[k] = \sum_{ngenap} W_N^{kn} x[n] + \sum_{nganjil} W_N^{kn}\, x[n]$$

$$= \sum_{n-2r}^{N-2} x(n)\, W_N^{nk} + \sum_{r=0}^{N-1} x(n)\, W_N^{nk}$$

$$= \sum_{r=0}^{N/2-1} x(2r)\, W_N^{2rk} + \sum_{r=0}^{N-1} x(2r+1)\, W_N^{(2r+1)k} \qquad (2.8)$$

$$\underline{< n = genap >} \qquad \underline{< n = ganjil >}$$
$$n = 2r \qquad\qquad n = 2r+1$$

Untuk penjabaran W_N^{2rk} dapat ditunjukkan pada persamaan 2.9.

$$W_N^{2rk} = W_{\frac{N}{2}}^{rk}$$

$$W_N^{k(2r)} = e^{-2m*kr\left(\frac{N}{2}\right)} = e^{-2m*kr\left(\frac{N}{2}\right)} = W_{\frac{N}{2}}^{rk}$$

$$W_N^{m+N/2} = W_m^m = e^{-2m} = \cos(-2\pi) + isin(-2\pi) = 1 \qquad (2.9)$$

Jadi X(k) pendefinisian bentuk genap ditunjukkan pada persamaan 2.10.

$$= \sum_{r=0}^{N/2-1} x(2r)\, W_{N/2}^{rk} \qquad (2.10)$$

Jadi X(k) pendefinisian bentuk ganjil ditunjukkan pada persamaan 2.11.

$$W_N^k \sum_{r=0}^{N-1} x(2r+1)\, W_{N/2}^{rk} \qquad (2.11)$$

2.5.6 *Mel Schale Filetrbank*.

Mel schale filterbank digunakan untuk menyesuaikan pendengaran manusia dengan sistem, dimana frekwensi yang tinggi akan mendapatkan *band* yang tinggi pula sehingga pada telinga manusia akan mendengar suatu sinyal yang konstan.

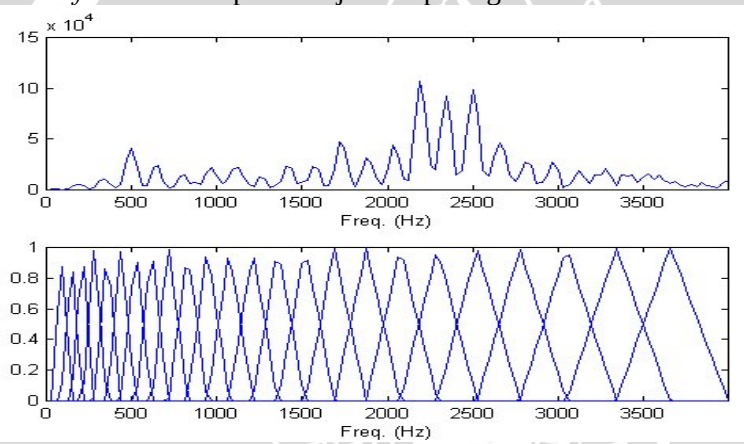
Pada *Hcopy* dapat digunakan baris TARGETKIND untuk menambahkan konfigurasinya pada *Hcopy* dapat digunakan pada *file* konfigurasi dibawah ini.

TARGETKIND=MFCC_X_X_X_X

Pada parameter diatas terdapat parameter X disana adalah menunjukkan operasi yang dilakukan pada *file* dimana dapat dimasukkan dengan parameter berikut:

_E mempunyi energi
_N Energi absolut disuppress
_D Menggunakan Delta Koefisien
_A Menggunakan Koefisien Akselerasi
_C Terkompress
_Z *Zero Mean koefisien statis*
_K Menggunakan CRC ceksum
_0 *Cepstral* Koefisien dimulai dari 0

Frekuensi *filter bank* dapat ditunjukkan pada gambar 2.13.



Gambar 2.13 Mel Schale Filterbank

2.5.7 *Mel Frequency Cepstrum Coeffisient*(MFCC).

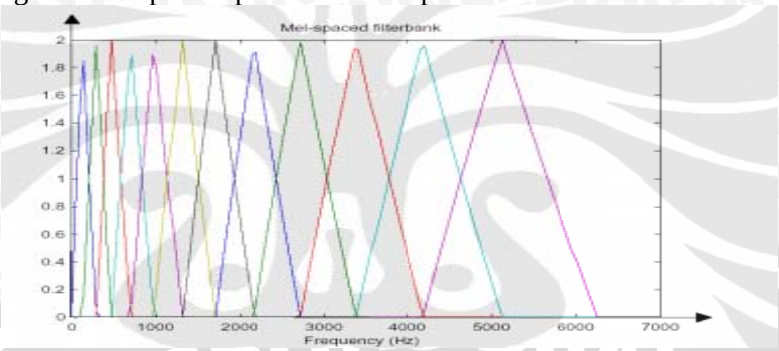
MFCC berfungsi untuk memfilter secara *linier* pada frekuensi rendah di bawah 1000Hz dan secara logaritmatik pada frekuensi tinggi

(2.12)

diatas 1000Hz. Hasil yang diperoleh akan dinyatakan dalam skala *Mel Frequency* (skala mel). Proses ini digunakan untuk menangkap karakterisrik dari suatu sinyal suara. Perhitungan MFCC menggunakan persamaan 2.12.

$$mel(f) = 2595 * log_{10}(1 + f/700) \quad (2.12)$$

Selanjutnya pada Gambar 2.13 ditunjukkan *filter bank* yag didapat dengan menempatkan pusat frekuensi pada skala mel-frekuensi.



Gambar 2.14 *Filter bank* pada pusat frekuensi.

Filter bank yang diaplikasikan dalam domain frekuensi menyederhanakan perhitungan untuk mengambil *triangle-shape window* pada spectrum hasil akhir dari proses MFCC adalah *mel cepstral coefisients,* persamaan 2.13 digunakan untuk menghitung *mel cepstral coefisients* (Ario Muhammad Fanie, 2008).

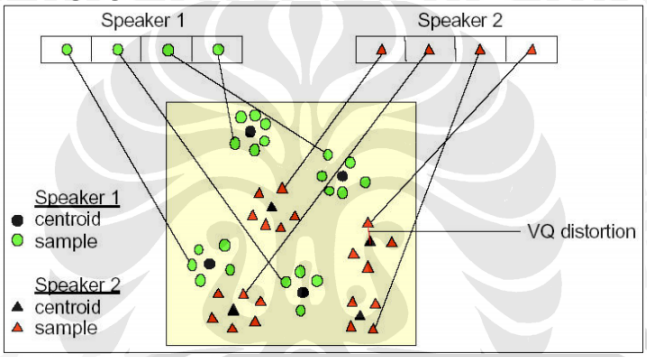
$$\check{C}_n = \sum_{k-1}^{K} (log\hat{S}_k) \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right] k = 1,2,\ldots,K \quad (2.13)$$

2.6 *Vector Quantisation*(VQ).

VQ merupakan pemetaan vektor dari ruang vektor yang besar menjadi wilayah yang terbatas yang disebut dengan *cluster*. Masing – masing *cluster* tersebut dapat direpresentasikan dengan *centroid* yang disebut *codbook*. Kumpulan dari semua *codbook* tersebut disebut dengan *codeword.*

Gambar 2.15 menunjukkan proses vektor kuantisasi. Terdapat 2 sumber suara dari 2 pembicara (*speaker*) dalam ruang akustik dua dimensi. Lingkaran menunjukkan vektor akustik dari suara 1 sementara segitiga merupakan vektor akustik dari suara 2. Dalam tahap pelatihan, *codebook* untuk masing – masing suara yang telah diketahui diperoleh dengan mengumpulkan vektor akustik yang dilatih

menjadi sebuah *cluster*. Hal ini ditunjukkan dengan lingkaran untuk suara 1 dan segitiga hitam untuk suara 2.



Gambar 2.15 *Codebook* dengan vektor kuantisasi.

Jarak dari suatu vektor ke *codeword* terdekat disebut dengan *distortion.* Pada proses identifikasi, suatu masukan dari suara atau gelombang lain yang tidak dikenal akan mengalami proses vektor kuantisasi dengan menggunakan semua *codebook* yang telah dilatih. Selanjutnya dihitung nilai VQ *distortion*, nilai VQ *distortion* yang paling kecil antara *codeword* dari salah satu suara dalam database dengan VQ *codebook* dari gelombang masukan akan digunakam sebagai hasil identifikaasi.

Untuk memperbaiki VQ pada pembentukan *codebook* digunakan *General Loyd Algoritma* (GLA) atau yang dikenal dengan algoritma LBG. Algoritma tersebut diimplementasikan dengan prosedur rekursif sebagai brikut :

1. Melakukan desain suatu vektor *codebook* merupakan *centroid* dari keseluruhan vektor pelatihan.
2. Membuat ukuran *codebook* dua kali lipat dengan membagi masing – masing *current codebook* C_n berdasarkan persamaam 2.14 dan persamaan 2.15.
   $C_n^+ = C_n(1+\varepsilon)$ (2.14)
   $C_n^- = C_n(1-\varepsilon)$ (2.15)
3. *Nearest neighbor Search,* adalah mengelompokkan vektor pelatihan yang berkumpul pada blok tertentu. Kemudian menentukan *codeword* dalam *current codebook* yang terdekat dan memberikan tanda vektor yaitu *cell* yang diasosiasikan dengan *codeword* yang terdekat.

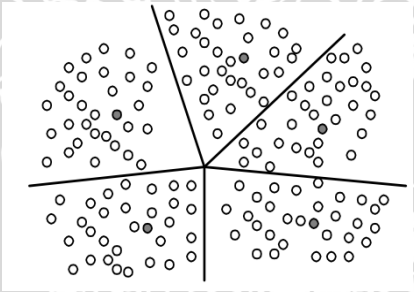
4. *Centroid update,* adalah menentukan *centroid* baru yang merupakan *codeword* yang baru pada masing – masing *cell* dengan menggunakan vektor pelatihan pada *cell* tersebut.
5. Iterasi 1, mengulang langkah 3 dan 4 sampai jarak rata – rata dibawah *present threshld.*
6. Iterasi 2, mengulang langkah 2,3,4 sampai *codebook* berukuran M(Ario Muhammad Fanie, 2008).
7.

2.7 Algoritma *K-means*.

Algoritma *K-means* merupakan satu algoritma yang sering kali digunakan di dalam teknik pengelompokan karena membuat suatu perkiraan yang efisien dan tidak memerlukan banyak parameter. *K-means* menggunakan k kelompok yang telah ditetapkan (k kelompok pertama sebagai *centroid*). Dalam algoritma ini vektor-vektor dikluster berdasarkan atribut menjadi K partisi. Ini menggunakan *K-mean* data dengan distribusi gaussian untuk mengkluster vektor – vektor tersebut. Tujuan *K-mean* adalah untuk memperpendek total varians intra-kluster, V yang ditunjukkan pada persamaam 2.16 (Ali Mustofa, 2007).

$$V = \sum_{i-1}^{k} \sum_{j=\epsilon S_i} |X_j - \mu_i|^2 \qquad (2.16)$$

Dimana ada K kluster S_i, I =1,2,3,…,k dan μ_i adalah *centroid* atau titik *mean* dari semua titik $x_j \in S_i$ gambar 2.15 merupakan ilustrasi *K-mean.*



Gambar 2.15 Ilustrasi *K-mean* membentuk lima kluster

2.8 Pengukuran Jarak

Dalam tahap pengenalan, suara pembicara yang tak dikenal direperentasikan oleh deretan vektor – vektor ciri {x_1, x_2 ….x_i), dan kemudian ini dibandingkan dengan *codebook* dari *database*. Untuk

mengidentifikasi pembicara yang tak dikenali, ini dapat dilakukan dengan pengukuran jarak distorsi dari dua kumpulan vektor yang berdasarkan jarak *Euclidean*. Jarak *Euclidean* adalah jarak antar dua titik yang akan diukur dengan suatu aturan, yang dapat dibuktikan oleh aplikasi teorema *Pythagorean*. Persamaan yang digunakan untuk menghitung jarak *Euclidean* dapat didefisikan dengan jarak *Euclidean* antara dua titik P = (p_1, p_2…p_n) dan Q = (q_1, q_2...q_n) pada persamaan 2.17 merupakan jarak *Euclidean* (Ali Mustofa, 2007).

$$= \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \qquad (2.17)$$

2.9 *Word Error Rate* (WER).

Pengujian dilakukan untuk mengetahui pengaruh ukuran *codebook* terhadap tingkat akurasi kata. Hasil pengujian kemudian dihitung keakuratannya dengan menggunakan Word Error Rate (WER). WER adalah sebuah matriksumum untuk mengukur performansi sistem pengenalan suara. Adapunpenghitungannya dapat dirumuskan pada persamaan 2.18.

$$WER = \frac{data\ uji\ benar}{data\ kata\ yang\ diujikan} \qquad (2.18)$$

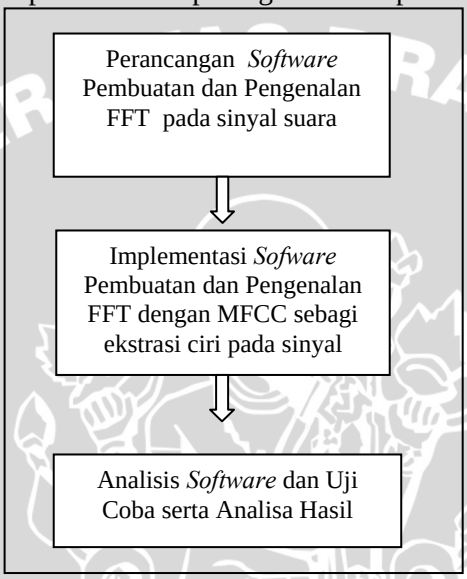
BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metode dan perancangan ini akan dibahas penggunaan metode yang digunakan dalam pembuatan perangkat lunak pengenalan tanda tangan serta langkah-langkah yang dilakukan dalam pembuatan perangkat lunak ini.

Tahapan pembuatan perangkat lunak pengenalan suara untuk *software* pembuka aplikasi ini adalah sebagai berikut:

1. Menganalisa dan merancang perangkat lunak dengan menggunakan penggabungan metode pada penelitian sebelumnya.
2. Membuat perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
3. Uji coba perangkat lunak.

Langkah-langkah pembuatan dapat digambarkan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Pembuatan Perangkat Lunak

3.1 Analisis Perangkat Lunak.

Pada subbab analisis ini akan dibahas mengenai semua hal yang diperlukan dalam proses pembuatan dan pengenalan FFT pada sinyal suara.

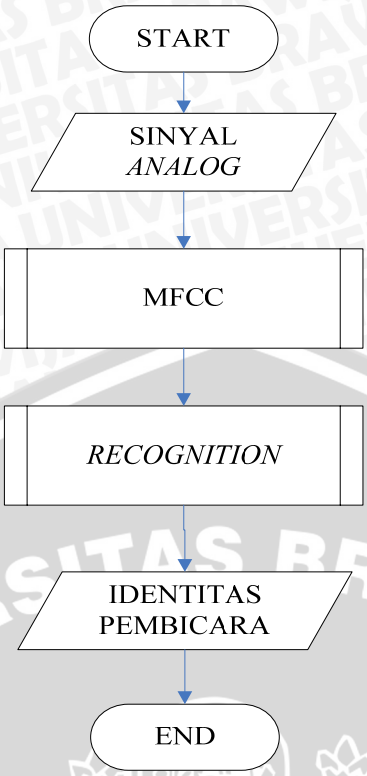
3.1.1 Deskripsi umum perangkat lunak.

Sistem pengenal suara ini merupakan sebuah sistem yang dapat melakukan pengenalan terhadap sinyal digital yang berisi frekuensi suara manusia.

Maka sistem pengenalan suara ini akan dibangun dengan menggunakan MFCC sebagai ekstrasi ciri. Sebelum melakukan pengenalan suara dilakukan terlebih dahulu proses – proses untuk memperoleh data digital yaitu proses ekstrasi gelombang yang setelah itu diproses dengan algoritma *vector quantization* (VQ), *K-mean,* yang kemudian dilakukan perbandingan pada *database* dengan rumus *Euclidean distance* pada persamaan 2.16.

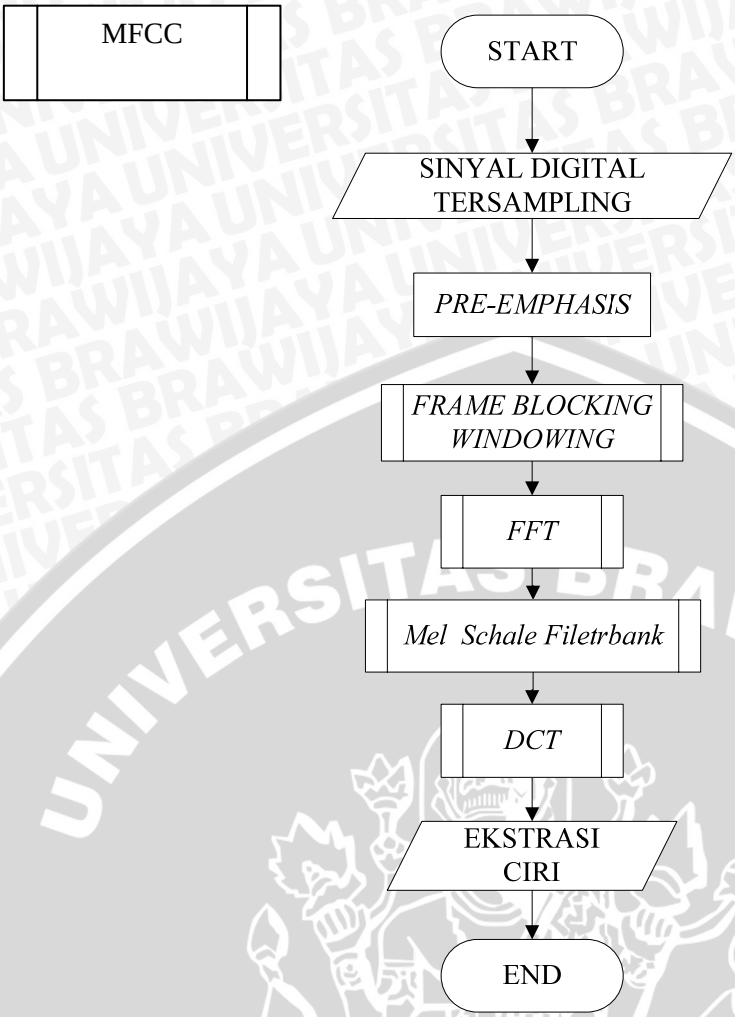
Subsistem pengenalan suara ini mempunyai fungsional sebagai mengubah pengolahan suara menjadi data digital, dan *File* yang diproses sebagai input berekstensi *Waveform Audio* (*.wav*).

Cara kerja perangkat lunak pengenalan suara ini dibagi menjadi beberapa tahap, tahap pertama adalah tahap input sinyal analog. Pada tahap ini suara dijadikan inputan sebagai pengenalan suara, langkah kedua proses ekstrasi gelombang. Pada tahap ini suara yang telah direkam dilakukan proses *sampling, frame blocking, windowing,* FFT, dan MFCC sebagai ekstrasi ciri. Langkah ketiga *vector quantisation* dimana algoritma ini merupakan pemetaan vektor dari ruang vektor yang besar menjadi wilayah yang terbatas yang disebut dengan *cluster.* Langkah keempat dari *cluster* tersebut algoritma K-mean meminimkan total varians intra-kluster. Dari beberapa langka ke-1 sampai dengan langkah ke-4 maka langkah yang terakhir dilakukan dengan pengukuran jarak distorsi dari dua kumpulan vektor yang berdasarkan jarak *Euclidean.* Setelah dari beberapa tahap selanjutnya diperoleh identitas pemilik suara. Diagram alur sistem pengenalan suara terdapat pada gambar 3.2.



Gambar 3.2 Diagram alur sistem pengenalan suara.

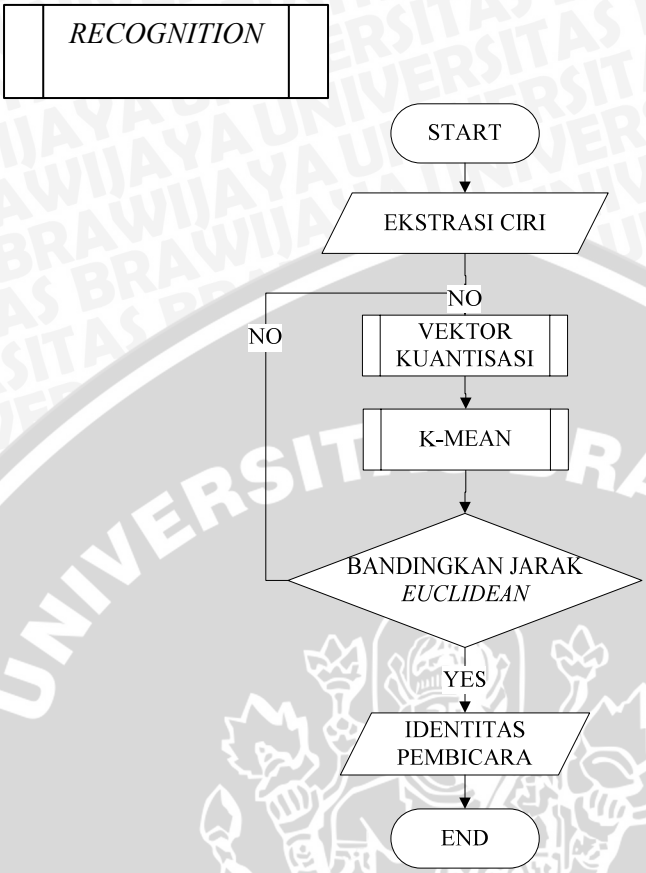
Dari gambar diagram alir sistem Gambar 3.2 dapat dijelaskan bahwa sistem pertama kali akan menerima *input* dari *user*. *Input* disini adalah sebuah sinyal analog yang berisi frekuensai pada suara. Kemudian sinyal analog yang masuk ke dalam sistem setelah direkam kemudian akan diproses dalam tahap *front end* menggunakan beberapa metode seperti *sampling*, *frame blocking*, *windowing* FFT serta ekstrasi ciri dengan MFCC. Setelah pada tahap *front end* akan di proses pada *back end* yaitu penyesuaian dengan data pada *database* dengan meenggunakan vektor kuantisasi, algoritma *K-mean*, dan *euclidean*. Jika data perbandingan telah sesuai maka hasil dari proses pengenalan berupa identitas pemilik suara akan ditampilkan.



Gambar 3.3 *Flowchart* Ekstraksi Ciri MFCC.

Dalam gambar 3.3, proses pengolahan suara dimulai dengan penyamplingan sinyal sebesar 16 KHz. Setelah itu *pre-emphasis* memperhalus bentuk sinyal yang kemudian dilakukan proses *frame blocking* dan *windowing* untuk membagi sinyal a*udio* menjadi beberapa *frame* dan mengurangi efek diskontinuitas pada masing – masing *frame* yang telah diperoleh dari proses *frame blocking.* Selanjutnya ditransformasi dengan algoritma

FFT dan *Mel schale filterbank* menormalisasi nada yang diterima menjadi nada konstan sebagai ekstraksi ciri MFCC.



Gambar 3.4. *Flowchart* dari proses pelatihan dan pengenalan

.

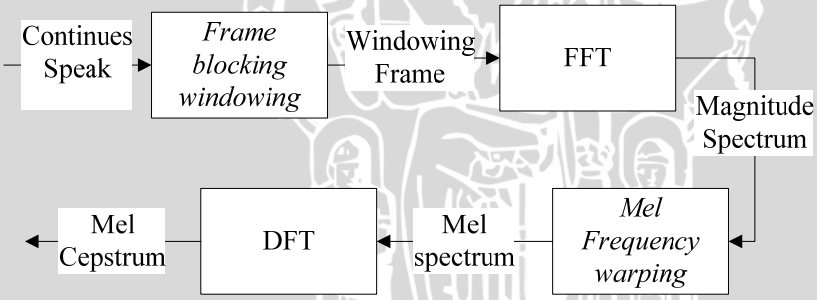
Pada gambar 3.4 dapat dijelaskan setelah proses pada gambar 3.3 yaitu ekstrasi ciri sinyal suara maka proses selanjutnya dilakukan proses vektor kuantisasi dimana proses tersebut merupakan pemetaan vektor dari ruang vektor yang besar menjadi wilayah yang terbatas yang disebut dengan *cluster*. Yang dipakai dalam proses ini adalah algoritma *clustering* untuk memperbaiki pembentukan *codebook*. Setelah dalam proses vektor kuantisasi akan dikelompokkan dengan algoritma *K-mean.* Untuk membandingkan suara dari *database* dilakukan pengukuran berdasarkan jarak *Euclidean*.

3.2 Perencangan Proses.

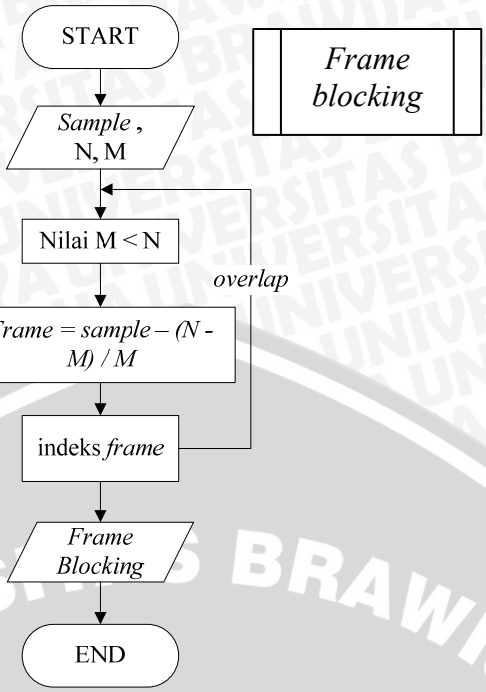
Proses yang ada dalam sistem pengenalan suara ini dibagi menjadi dua proses yaitu proses *front end* dan proses *back end*. Proses *front end* bertugas untuk mengubah sinyal analog menjadi data digital, sebelum masuk dalam tahap implementasi pengenalan suara. Kemudian proses *back end* bertujuan mengolah sinyal yang telah didapat pada proses *front end* yang kemudian akan dilakukan pengenalan suara.

3.2.1 Proses MFCC.

Dalam proses *front end* ini adalah pemrosesan ekstrasi ciri MFCC yang pertama setelah suara dilakukan perekaman, diperoleh sinyal analog yang kemudian sinyal analog tersebut disampling sebesar 16 KHz yang kemudian diperhalus pre-emphasis untuk mendapat sinyal yang akurat. Dari proses *sampling* kemudian dilakukan *frame blocking* juga *windowing* membagi – baginya beberapa *frame* setelah dari proses tersebut dilakukan transformasi dengan FFT untuk mencari data digital. Kemudian agar sistem dapat menyesuaikan pendengarannya dilakukan proses *mel filter bank* menjadikan pendengaran secara konstan. Proses log mengadaptasi sistem agar dapat menerima sinyal suara sebagai suara asli ddan selanjutnya IFFT adalah kebalikan dari proses DFT yang merupakan *invers* dari transformasi. *Front end* dapat dilihat pada gambar 3.5.



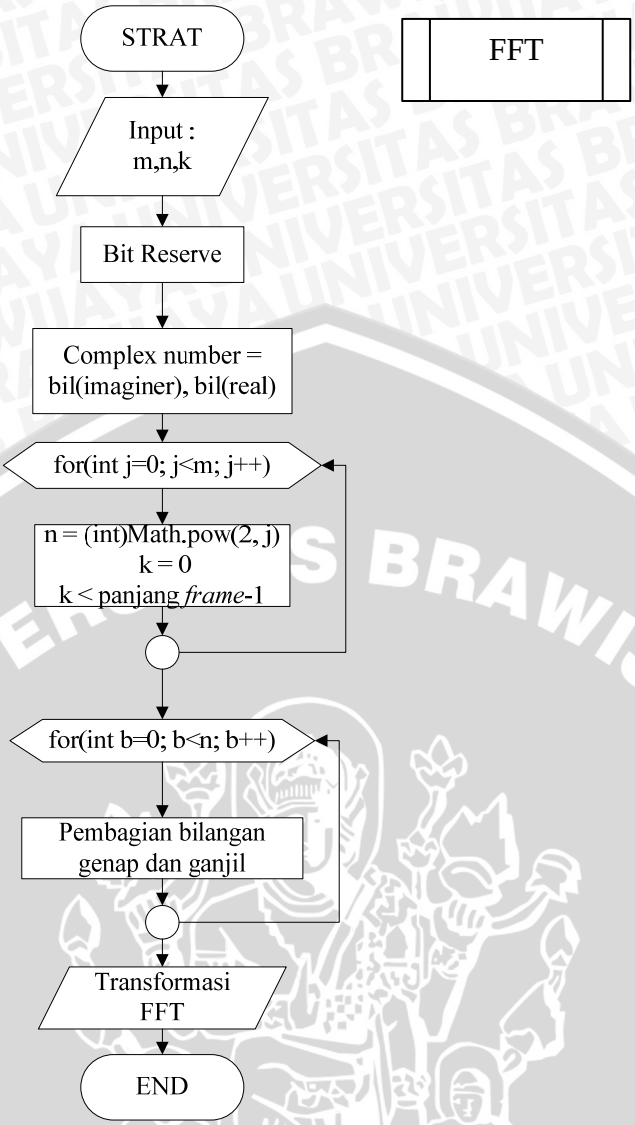
Gambar 3.5 Blok diagram proses MFCC.



Gambar 3.6 *Frame blocking.*

Sinyal keluaran *pre-emphasis* kemudian dibagi-bagi menjadi beberapa frame waktu yang sangat singkat untuk memperoleh kondisi sinyal yang stabil dalam domain waktu. Tiap *frame* berisi N sampel dengan jarak antar *frame* M sampel. Nilai M<N sehingga ada bagian *frame* yang overlapping sepanjang N-M sampel.

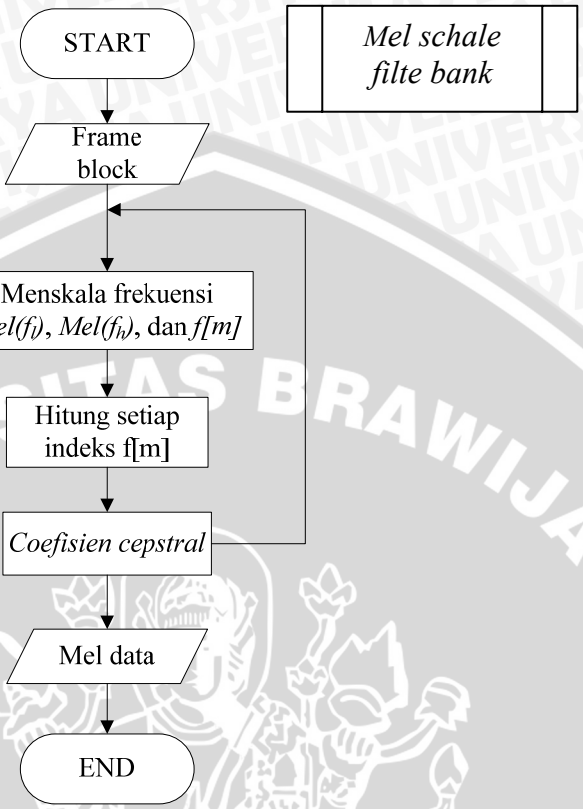
Pada proses FFT input n adalah jumlah data, m pnjang *frame*, k sinyal *sample* yaitu kelipatan dari n. Dimulai pada sinyal dengan membagi menjadi dua bagian yaitu bentuk sinyal pada waktu genap dan ganjil. Kemudian melakukan pembagian sinyal yang lebih kecil dengan 2/n point sampai N/n point dengan N kelipatan dari jumlah *sample*. Dilakukan langkah secar terus menerus sesuai dari skema *butterfly* sampai mendapat kombinasi dari satu *sample* FFT. Dapat dilihat pada gambar 3.7.



Gambar 3.7 *Fast Fourier Transform*.

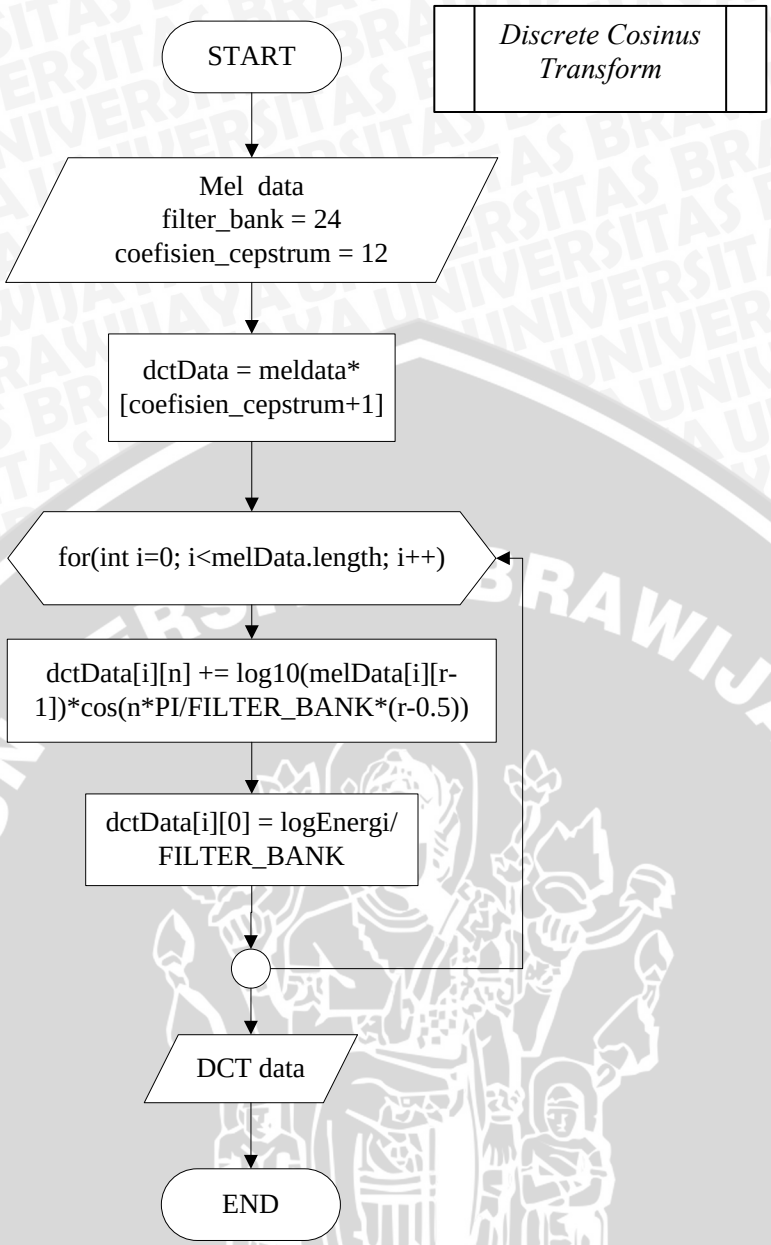
Mel-frequency filter bank adalah analisis frekuensi dengan menskalanya menggunakan mel *filter bank*. Parameter-parameter yang digunakan pada *mel-frequency filter bank* yaitu frekuensi terendah (f_l), frekuensi tertinggi (f_h), dan jumlah *filter bank* (M). Pada skripsi ini

frekuensi terendah digunakan nilai 0 Hz sedangkan frekuensi tertinggi digunakan nilai 8000 Hz. Untuk jumlah *filter bank* yang digunakan umumnya adalah 24 *filter bank* flowchart proses *filter bank* dapat ditunjukkan seperti pada gambar 3.8.



Gambar 3.8. *Mel schale filter bank.*

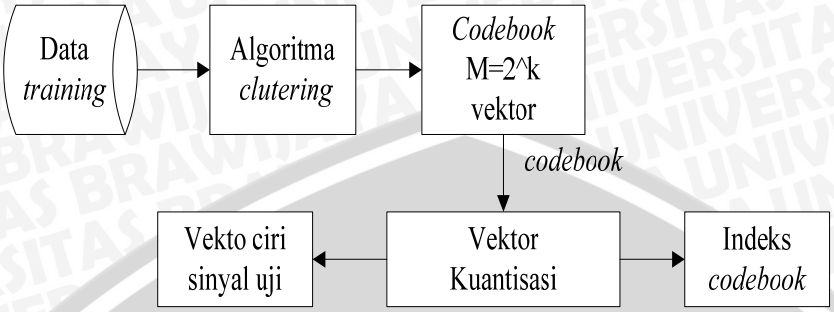
Discrete Cosinus Transform digunakan untuk mengembalikan dari domain frekuensi ke domain waktu dan menghasilkan koefisien *cepstrum*. Jumlah koefisien *cepstrum* yang dihasilkan adalah 12 *cepstrum* mulai dari C_1 sampai C_{12} sedangkan C_0 adalah nilai rata-rata log energi pada proses *mel-frequency filter bank*. Jadi jumlah koefisien *cepstrum* yang dihasilkan adalah 13 *cepstrum*. Ditunjukkan pada gambar 3.9.



Gambar 3.9. *Discrete Cosinus Transform.*

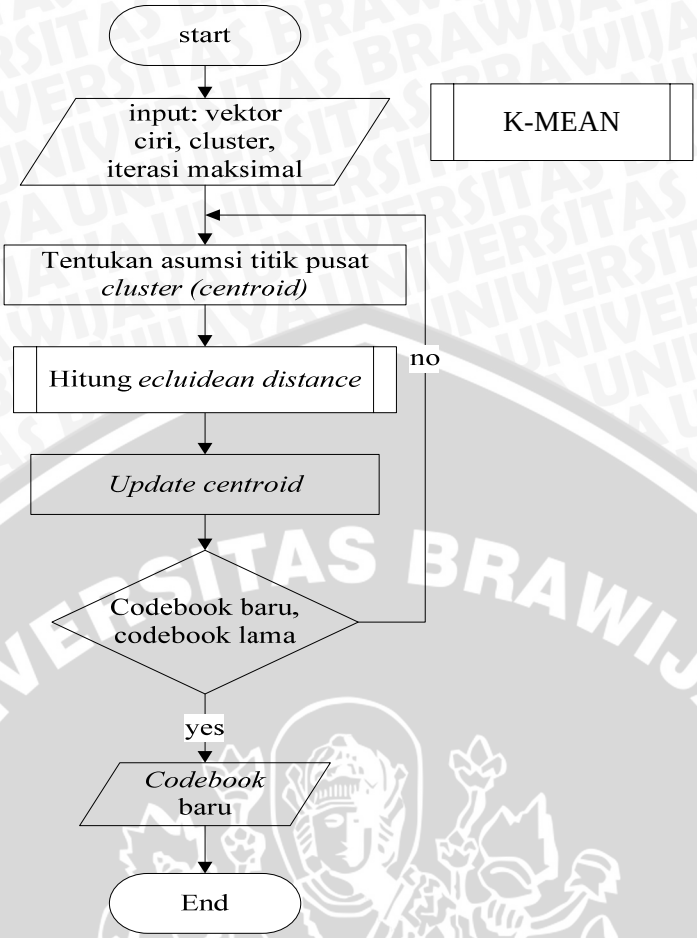
3.2.2 Perancangan *Recognition.*

Pada proses ini hasil MFCC akan dip roses dengan vektor kuantisasi dengan mempergunakan algoritma *clustering* dimana setiap *cluster* direpresentasikan dengan *centroid* yang disebut *codeword.* Diagram vector kuantisasi dapat dilihat pada gambar 3.10.



Gambar 3.10 Vektor Kuantisasi.

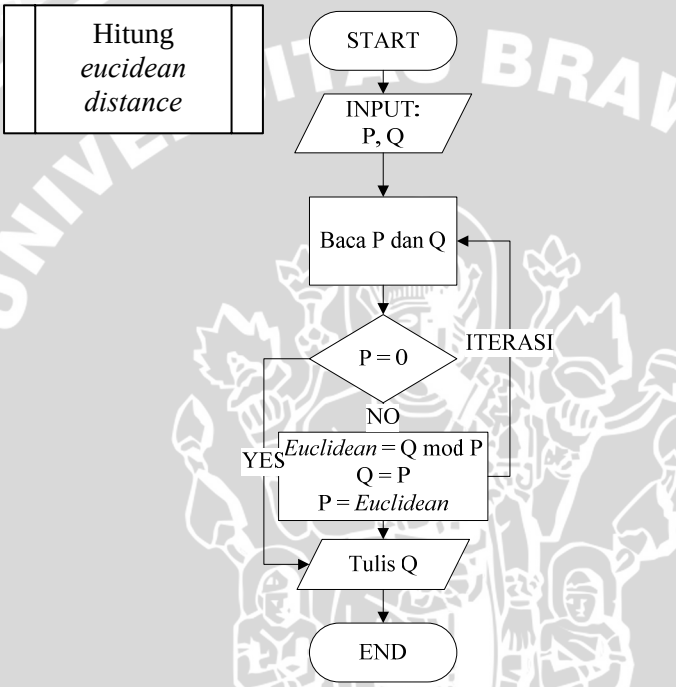
Seperti yang terlihat pada Gambar 3.6 bahwa kuantisasi vektor terpisah menjadi dua tahap yaitu, pembentukan *codebook* dan penentuan indeks *codebook*. Saat pembentukan *codebook*, masukan kuantisasi vektor adalah vektor ciri dari seluruh variasi sinyal suara yang akan dikenali. Dengan Algoritma *clustering*, vektor ciri tersebut dikelompokkan menjadi k2 cluster sehingga menghasilkan *codebook*. Algoritma *clustering* yang digunakan pada skripsi ini adalah kmeans *clustering*. Parameter algoritma kmeans yaitu,banyaknya *codebook* yang akan dibentuk k, k=1,2,...,L dimana L adalah panjang set data *training* maksimal, jumlah iterasi maksimum untuk meng-update *centroid* (pusat sebaran data pada suatu cluster), dan nilai mutlak dari selisih distorsi sebagai kondisi henti meng_*update centroid*. Setelah *codebook* terbentuk, maka kuantisasi vektor dapat dilakukan dengan mengganti suatu vektor ciri dengan salah satu vektor *codebook* yang mempunyai *euclidean distance* terkecil. Flowchart pembentukan *codebook* menggunakan algoritma *k-means clustering* ditunjukkan pada Gambar 3.11.



Gambar 3.11 Algoritma *K-mean.*

Dari gambar 3.11 Inputan berupa vektor ciri hasil ekstraksi ciri yang berupa vektor, *set centroid* secara random sebanyak *codebook*. Pada skripsi ini menggunakan ukuran *codebook* 16, 32, 64, 126, 256 untuk setiap data uji. Ukuran *codebook* ini ditetapkan pada *range* antara 2^k lebih besar dari jumlah variasi data dan data 2^{k+1} lebih kecil dari banyak variasi data. Jika ukuran *codebook* terlalu kecil daripada banyak variasi data maka ada kemungkinan untuk kelas suara yang sama dikelompokkan ke dalam satu *codebook*, dan jika ukuran *codebook* terlalu besar ada kemungkinan terdapat *codebook* yang tidak merepresentasikan kelas suara tertentu. Setelah *centroid* awal

terbentuk, kemudian menghitung *euclidean distance*. Hasil perhitungan *euclidean distance* dipilih vektor kolom yang mempunyai nilai minimum dan ditentukan indeks kolom yang memuat nilai *euclidean distance* minimum sebagai vektor ciri yang terdekat dengan *centroid* awal. *Update centroid*. Melakukan proses iterasi yang dimulai dari 1 sampai ncluster. Menghitung *error. Error* per iterasi dihitung dengan menjumlahkan vektor kolom yang memuat nilai *euclidean distance* minimum. Kemudian menyeleksi apakah *codebook* baru yang terbentuk sama dengan *codebook* lama. Jika sama, maka proses selesai dan jika tidak maka proses akan melakukan iterasi lagi sampai batas *max*_iterasi, dan berhenti apabila nilai maksimum dari selisih *codebook* baru dan *codebook* lama kurang dari 10*eps dengan nilai eps sebesar 0,01. Untuk pengukuran e*uclidean* ditunjukkan pada gambar 3.12.



Gambar 3.12 Algoritma *euclidean.*

Dari gambar 3.8 input dua kumpulan vektor P dan Q dimana P = (p_1, p_2…p_n) dan Q = (q_1, q_2...q_n). Kemudian menyeleksi apakah indeks kolom yang mempunyai nilai *euclidean distance* dengan nilai iterasi

lebih dari nol. Apabila hasilnya lebih dari nol maka *update centroid*, dan apabila hasilnya kurang dari atau sama dengan nol maka kembali melakukan proses iterasi sampai ncluster.

3.3 Perancangan Tabel.

Dari perancangan proses pola suara dan implementasi proses FFT yang telah dipaparkan pada sub bab 3.2.1 dan sub bab 3.2.2, maka diperlukan tabel 3.1 dan tabel 3.2 untuk penyimpanan pada sistem pencocokan pola suara ini.

Tabel 3.1 Tabel *centroid*

Field	Type	Keterangsan
id	Integer(11)	Berisi id dari nama pemilik
Name	varchar(50)Text	Berisi nama pemilik
c1	double	Nilai tiap *cluster*
c2	double	Nilai tiap *cluster*
. .	double	Nilai tiap *cluster*
c39	double	Nilai tiap *cluster*

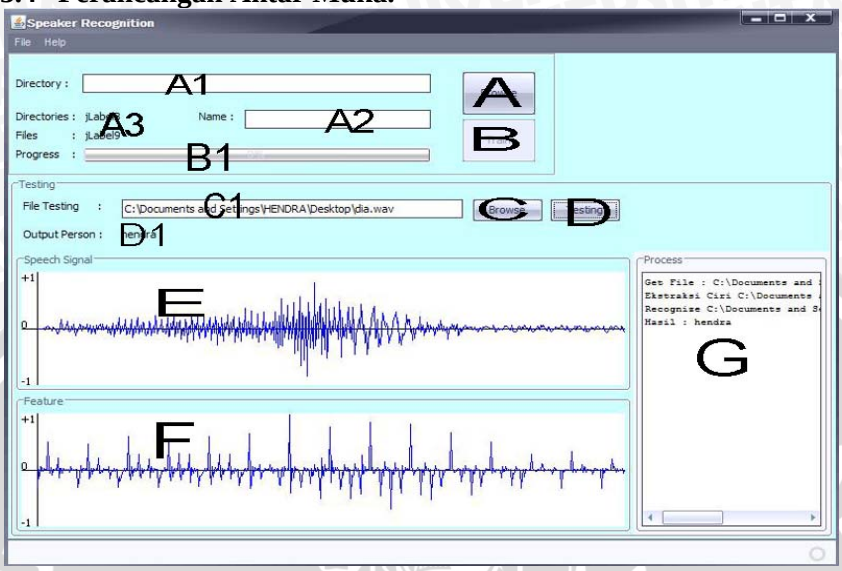
Tabel *centroid* (3.1), berisi data id, name, c1 s/d c39, *cluster* untuk penyimpanan data. Tabel *Recognition* diperlukan sebagai identitas dari pengenalan suara.

Tabel 3.2 Tabel *vector_cepstral*

Field	Type	Keterangan
id	Integer(11)	Berisi id dari nama pemilik
Name	varchar(50)Text	Berisi nama pemilik
c1	double	Nilai tiap *cluster*
c2	double	Nilai tiap *cluster*
. .	double	Nilai tiap *cluster*
c39	double	Nilai tiap *cluster*
cluster	smallint(6)	*Cluster* setiap nilai

Tabel *vector_cepstral* (3.2), berisi data id, name, c1 s/d c39, cluter. Tabel Nilai File WAV diperlukan sebagai nilai pencocokan data digital yang diproses sebagai pengenalan suara. Tabel ini terhubung oleh Tabel *centroid* jika proses pengenalan sesuai maka menampilkan kedua tabel tersebut dalam software.

3.4 Perancangan Antar Muka.



Gambar 3.13Desain *Interface.*

A : berfungsi sebagai pemanggil data *training view file open.*
A1 : *view* alamat data.
A2 : pemberi nama pemilik suara.
A3 : berfungsi jumlah data *training.*
B : berfungsi sebagai proses *training.*
B1 :.berapa persen proses data *training.*
C : *view file open* data yang akan dites.
C1 : *view* alamat *data* tes.
D : berfungsi sebagai menu data yang akan dites pada C.
E : berfungsi sebagai *graph* sinyal.
F : berfungsi sebagai *graph* ekstraksi ciri.
G : proses data.

3.5 Contoh Perhitungan

Pada sub bab ini akan dibahas tentang contoh perhitungan manual. Misalkan ada sinyal suara sebanyak 14 sampel. Diasumsikan sinyal suara tersebut sudah melalui pemrosesan awal indeks dimulai dari angka 0. Nilai tiap sampel untuk sinyal suara tersebut ditunjukkan pada tabel 3.3.

Table 3.3 Tabel sinyal suara.

0	0.88281
1	-0.13638
2	0.69531
3	-0.96094
4	0.82031
5	-0.42187
6	0.89062
7	-0.56250
8	0.55469
9	-0.89844
10	0.19531
11	-0.42969
12	0.04687
13	-0.58594

Data pada tabel 3.3 merupakan data pada sinyal setelah di_*sampling* berupa sinyal *digital.*

3.5.1 Perhitungan *Pre-emphasis.*

Tahap selanjutnya adalah preemphasis menggunakan rumus Persamaan 2.1 nilai *alpha_pre* yang digunakan adalah 0,97. Berikut contoh perhitungan untuk indeks ke-2 atau *y(2)*.

y(1) = s'(1) - 0.97*s'(0) = -0.13638 - 0.97*0.88281 = -0.99271

untuk indeks yang lainnya, perhitungannya sama dengan diatas hanya nilai tiap indeksnya diganti. Sinyal suara *pre-emphasis* ditunjukkan pada tabel 3.4

Tabel 3.4 Tabel sinyal suara *pre-emphasis*.

0	0.88281
1	-0.99271
2	0.8276
3	1.75242
4	-1.63539
5	-1.21757
6	1.29983
7	-1.4264
8	1.10032
9	-1.43649
10	1.0668
11	-0.61914
12	0.46367
13	-0.6314

Dari proses perhitungan dengan persamaan 2.1 didapat nilai setiap perhitungan *pre-emphasis* didalam tabel 3.4 pada seriap *frame*.

3.5.2 Perhitungan *Frame* Blocking.

Tahap berikutnya adalah *frame blocking*, Dari tabel 3.4 akan di ambil beberapa *frame* misalkan panjang *frame* yang digunakan adalah 8 dan *frame shift*/jarak antar *frame* sama dengan 3 pada setiap *looping frame* yang dilakukan *frame blocking*. *Frame blocking* dapat ditunjukkan pada tabel 3.5.

Tabel 3.5 Tabel *frame blocking*

frame ke-0		*frame* ke-1		*frame* ke-2	
0	0.88281	0	-1.63539	0	1.29983
1	-0.9927	1	1.75242	1	-1.4264
2	0.8276	2	-1.21757	2	1.10032
3	-1.6354	3	1.29983	3	-1.4365
4	1.75242	4	-1.4264	4	1.0668
5	-1.2176	5	1.10032	5	-0.6191
6	1.29983	6	-1.43649	6	0.46367
7	-1.4264	7	1.0668	7	-0.6314

3.5.3 Perhitungan *Hamming Window.*

Proses berikutnya adalah *hamming window*. Tiap *frame* dikalikan dengan fungsi *hamming window* $W_H(n)$. Berikut adalah perhitungan untuk *frame* ke-0 indeks ke-2 dengan persamaan 2.2.
$r_{yy}(0, 2)*W_H(2) = 0.88824*(0.54 - 0.46*\cos((2*2*3.14)/7)) = 0.57057$
untuk *frame* dan indeks lainnya prosesnya sama dengan diatas. *Frame hamming window* dapat ditunjukkan pada tabel 3.6.

Table 3.6. Tabel *hamming window.*

frame ke-0		*frame* ke-1		*frame* ke-2	
0	0.0706248	0	-0.130831	0	0.1039864
1	-0.25118647	1	0.443416704	1	-0.36092352
2	0.531279014	2	-0.78162082	2	0.706352012
3	-1.56044471	3	1.240262471	3	-1.37065973
4	1.673225117	4	-1.36193852	4	1.018589468
5	-0.78335985	5	0.707923578	5	-0.39834212
6	0.330387771	6	-0.36512369	6	0.117854564
7	-0.11411533	7	0.08534649	7	-0.05051347

3.5.4 Perhitungan *Fast Fourier Transform.*

Metode yang digunakan untuk menghitung FFT adalah "*decimation in time*". Pertama-tama dilakukan operasi *bit-reserve* pada tiap *frame* kemudian dilakukan operasi *butterfly*. Contoh perhitungan untuk *frame* ke-0 dan perubahan operasi bit-reserve dapat ditunjukkan pada tabel 3.7.

Tabel 3.7 Tabel perubahan operasi *bit*.

frame ke-0		operasi bit-reserve *frame* ke-0	
0	0.0706248	0	0.0706248
1	-0.25118647	4	1.673225117
2	0.531279014	2	0.531279014
3	-1.56044471	6	0.330387771
4	1.673225117	1	-0.25118647
5	-0.78335985	5	-0.78335985
6	0.330387771	3	-1.56044471
7	-0.11411533	7	-0.11411533

operasi *butterfly* seperti pada gambar 2.4. Pertama dilakukan operasi *butterfly* dua-dua kemudian naik menjadi empat dan naik lagi menjadi delapan atau dua kalinya seperti pada tabel 3.8.

Tabel 3.8 Tabel Operasi *Butterflies* pada FFT

indeks	nilai		nilai akhir (bil kompleks)
0	0.07682	nilai(0)+W^0*nilai(1)	-0.00147 + 0i
1	0.88246	nilai(0)+W^4*nilai(1)	0.39776 - 0.97250i
2	0.57057	nilai(2)+W^0*nilai(3)	0.05720 - 0.02969i
3	0.26704	nilai(2)+W^4*nilai(3)	-0.17567 + 0.06225i
4	-0.23726	nilai(4)+W^0*nilai(5)	-0.33982 + 0i
5	-0.60122	nilai(4)+W^4*nilai(5)	-0.17567 - 0.06225i
6	-0.89279	nilai(6)+W^0*nilai(7)	0.05720 + 0.02970i
7	-0.06979	nilai(6)+W^4*nilai(7)	0.39776 + 0.97250i

Untuk *frame* ke -1 dan seterusnya dapat diproses dengan cara yang sama seperti pada tabel 3.8.

3.5.5 Perhitungan *Mel-frequency Filter bank*.

Mel-frequency filter bank digunakan untuk menskala frekuensi menggunakan *mel-scale*. Sebelum menghitung *mel-frequency filter bank*, pertama-tama dihitung *Mel(f_l)*, *Mel(f_h)*, dan *f[m]*. Misalkan jumlah *filter bank* yang digunakan adalah 4, fs = 16000 Hz, fl = 0 Hz, dan fh = 4000 Hz.

$Mel(f_l) = Mel(0) = 2595 \log_{10}(1+0/700) = 0$
$Mel(f_h) = Mel(40000) = 2595 \log_{10}(1+4000/700) = 2146.06453$

kemudian menghitung f[m] pada *frame* ke-0 untuk m = 0,1,...,5 atau sebanyak jumlah *filter bank* ditambah 2 untuk indeks 0 dan 5. Pada tabel 3.9 perhitungan f[m] *filter bank*.

Tabel 3.9 Tabel perhitungan f[m] *filter bank*.

m	0	1	2	3	4	5
f[m]	0.00000	0.16223	0.39967	0.74715	1.25571	2.00000

Misalkan ingin menghitung *filter bank* ke-3 adalah sebagai berikut.

X'(3) = |X(0)|*H(0,3) + |X(1)|*H(1,3) + |X(2)|*H(2,3) +
|X(3)|*H(3,3) + |X(4)|H(4,3) + |X(5)|H(5,3) +
|X(6)|H(6,3) + |X(7)|H(7,3)
= |(-0.45495 + 1.00220i)|*0 + |(0.11700- 0.03255i)|*1.17475 + |(0.91177 - 1.03475i)|*0 + |(0.97044 - 0.93995i)|*0 + |(-6.93889E-18 + 0.06511i)|*0 + |(-0.97044 - 0.93995i)|*0 + |(-0.91324-.03475i)|*0 + |(-0.11847 - 0.03255i)|*0 = 0.13745 + 0.03824i

Sebelumnya yang panjang *frame* 8 menjadi 4 sesuai jumlah *filter bank* ditunjukkan pada tabel 3.10.

Table 3.10 Tabel *frame* setelah dilakukan *filter bank*

1	2	3	4
0.00000	0.00000	0.01273	0.02965

Untuk *frame* selanjutnya dapat diproses dengan cara yang sama seperti pada tabel 3.10.

3.5.6 Perhitungan *Discrete Cosinus Transform* (DCT).

Selanjutnya adalah menghitung DCT, Misalkan jumlah koefisien *cepstrum* yang dikehendaki adalah 2. Berikut adalah contoh untuk perhitungan koefisien *cepstrum* ke 1 pada *frame* ke-0 pada persamaan 2.4.

C(1) = X(0)*cos(3.14/4*0-0.5) + X(1)*cos(3.14/4*1-0.5) +
X(2)*cos(3.14/4*2-0.5) + X(3)*cos(3.14/4*3-0.5)
= 0 + 0 + 0.01273*1.1775 + 0.02965*1.9625
= 0.01499 + 0.05819 = 0.07318

proses untuk koefisien *cepstrum* berikutnya sama dengan diatas. Ditunjukkan pada tabel 3.11.

Tabel 3.11 Tabel *Frame* setelah dilakukan DCT.

1	2	3	4
0.07318	0.12963	0.09634	0.16246

untuk *frame* yang lain prosesnya sama dengan tabel 3.11.

Perbandingan data ke-1 dengan data masukan
VpR = data pembelajaran *real*
VqR = data *input real*
VpI = data pembelajaran *imaginer*
VqI = data *input imaginer*
n = jumlah data

$$Dreal = \frac{\sqrt{(VpRn - VqRn)^2 + (VpRn - VqRn)^2}}{n}$$
$$= \frac{\sqrt{0 + 16 + 25 + 1 + 25 + 16 + 0 + 16}}{8} = 6.1875$$

$$Dimaginer = \frac{\sqrt{(VpIn - VqIn)^2 + (VpIn - VqIn)^2}}{n}$$
$$= \frac{\sqrt{0 + 0 + 4 + 4 + 4 + 4 + 4 + 4}}{8} = 1.5$$

$$Fk = \sqrt{Dreal^2 + Dimaginer^2}$$

$$= \sqrt{6.1875^2 + 1.5^2} = 20.26758$$

Persentase kesamaan :

$$\% = \frac{100 - Fk}{100} x100 = \frac{100 - 20.26758}{100} x100 = 79.73\%$$

Perbandingan data ke-2 dengan data masukan

$$Dreal = \frac{\sqrt{0 + 16 + 25 + 1 + 25 + 0 + 0 + 0}}{8} = 4.875$$

$$Dimaginer = \frac{\sqrt{(0 + 0 + 4 + 4 + 4 + 0 + 4 + 0)}}{8} = 1$$

$$Fk = \sqrt{4.875^2 + 1^2} = 9.27$$

Persentase kesamaan :

$$\% = \frac{100 - 9.27}{100} x100 = 90.73\%$$

Perbandingan data ke-3 dengan data masukan

$$Dreal = \frac{\sqrt{(0+16+25+0+0+0+25+16)}}{8} = 5.125$$

$$Dimaginer = \frac{\sqrt{0+0+4+0+4+4+4+4}}{8} = 1.25$$

$$Fk = \sqrt{5.125^2 + 1.25^2} = 13.91$$

Persentase kesamaan :

$$\% = \frac{100 - 13.91}{100} x100 = 86.09\%$$

Sehingga input tersebut termasuk pada data ke-2 karena persentase terbesar pada perbandingan data ke-2 dengan data *inputan* sebesar 90.73%

3.6 Perancangan Uji Coba

Proses perancangan uji coba merupakan tahapan akhir setelah perangkat lunak selesai dikembangkan. Perancangan uji coba yang nantinya akan dilakukan meliputi uji coba terhadap sistem yang telah dibangun dengan melakukan proses pembelajaran dan proses pengenalan pengenalan suara. Dari 5 macam suara dari 6 orang yang berbeda, akan digunakan untuk data pelatihan dan data pengujian pengenalan suara.

Ada beberapa parameter yang digunakan untuk mengetahui seberapa besar persentase keberhasilan sistem terhadap pengenalan suara. Hasil pengujian tersebut kemudian dihitung keakuratannya dengan menggunakan *Word Error Rate* (WER). Tabel hasil uji pembelajaran ditunjukkan pada tabel 3.12 sedangkan untuk belum dilakukan pembelajaran pada tabel 3.13.

Tabel 3.12 Tabel Hasil Uji Pembelajaran

nama	kata	benar	salah
	halo1		
	halo2		
	halo3		
	halo4		
	halo5		
WER %			

Tabel 3.13 Tabel Hasil Uji Belum Dilakukan Pembelajaran

nama	kata	benar	salah
	halo1		
	halo2		
	halo3		
	halo4		
	halo5		
WER %			

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai implementasi dari perancangan pada bab 3. Berikut ini uraian spesifikasi perangkat keras, perangkat lunak yang digunakan dalam sistem pengenalan suara dan implementasi.

4.1. Perangkat Keras.

Perangkat keras yang digunakan dalam menerapkan system pengenalan suara dengan Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) sebagai ekstraksi ciri adalah
sebagai berikut :

1. Processor Intel (R) Pentium(R) Dual CPU T2330 1.60 GHz.
2. Memori 1 GB.
3. HardDisk 120 GB.

4.2. Perangkat Lunak.

Perangkat lunak yang digunakan dalam menerapkan sistem pengenalan suara dengan Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) sebagai ekstraksi ciri adalah
sebagai berikut :

1. Sistem operasi Microsoft Windows XP.
2. Bahasa pemrograman yang digunakan dalam menerapkan sistem pengenalan suara adalah JAVA.
3. Jet Audio *version* 6.2.1.6701 *Plus* VX.
4. XAMPP.

4.3. Implementasi Program.

Berdasarkan analisa dan perancangan proses yang terdapat dalam sub bab 3.4, maka pada sub bab ini akan dijelaskan implementasi proses-proses tersebut.

4.3.1. Deskripsi Program.

Struktur data yang digunakan dalam pemrosesan sinyal tampak pada Gambar 4.1.

```
private final double ALPHA_PRE = 0.97;
private final int FRAME_WIDTH = 512;
private final int FRAME_SHIFT = 160;
private final int FILTER_BANK = 24;
private final int COEF_CEPSTRUM = 12;
```

Gambar 4.1 *Source code* struktur data proses suara.

ALPHA_PRE adalah nilai *coefisien pre-emphasis* digunakan untuk mengambil *envelope* dari sinyal tersampel untuk memperhalus bentuk sinyal. *FRAME_WIDTH* adalah lebar *frame* digunakan untuk panjang *frame* yang ditentukan. *FRAME_SHIFT* perubahan suatu *frame* yaitu merubah *frame* asli dengan membaginya dengan yang dibutuhkan. *FILTER_BANK* nilai *mel* digunakan untuk penyesuaian frekuansi yang dterima. *COEF_CEPSTRUM* nilai koefisien *cepstral* digunakan untuk menangkap karakterisrik dari suatu sinyal suara.

4.3.2. Pemrosesan Awal.

Tahap pemrosesan awal dilakukan untuk menyetarakan semua sinyal suara dan mengubah sinyal suara dari analog menjadi digital. Pada tahap ini, terdiri dari 2 proses yaitu *Sampling* normalisasi, *Pre-Emphasis*.

4.3.2.1. *Sampling.*

Sampling merupakan proses awal dalam pengenalan suara yaitu merubah sinyal *analog* menjadi sinyal *digital* hal ini bertujuan untuk membaca *file* dari *database* dengan bentuk *digital*. Dalam proses ini digunakan *import* pada java berupa AudioFormat sebagai mengambil *file* format (WAV), AudioInputStream sebagai pengambilan data, AudioSystem sebagai *system audio. Source code sampling* ini digambarkan pada Gambar 4.2.

```
public double[] bacaData(File fileName){
        AudioInputStream audioInputStream = null;
        try {
            audioInputStream =
AudioSystem.getAudioInputStream(fileName);
        } catch (UnsupportedAudioFileException e) {
            JOptionPane.showMessageDialog(null,
"File Unsupported!", "Error Message",
                JOptionPane.ERROR_MESSAGE);
```

```
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null,

"I/O Error!", "Error Message",
                    JOptionPane.ERROR_MESSAGE);
        }
AudioFormat audioFormat =
audioInputStream.getFormat();
        double[] audioData = new double[(int)
audioInputStream.getFrameLength()];
        byte[] buffer = new
byte[FRAME_WIDTH*2*10];
        int sampleMax = 0;
        int indeks = 0;
        int nByteRead = 0;

        do {
            try {
                nByteRead =
audioInputStream.read(buffer);
            } catch (IOException e) {

JOptionPane.showMessageDialog(null, "Cannot Read
Audio Stream!",
                        "Error Message",
JOptionPane.ERROR_MESSAGE);
            }
            for(int i=0; i<nByteRead/2; i++){
                int LSB = (int)buffer[2*i];
                int MSB = (int)buffer[2*i+1];
                int result = MSB << 8 | (255 &
LSB);
                if(Math.abs(result) > sampleMax)
sampleMax = Math.abs(result);
                audioData[indeks++] = result;
            }
        } while(nByteRead != -1);
        normalisasi(audioData, sampleMax);
        return audioData;
    }
```

Gambar 4.2 *Source code sampling.*

4.3.2.2. Normalisasi.

Pada tahap selanjutnya dilakukan proses normalisasi bertujuan untuk menyamakannya amplitudo maksimum semua sinyal agar suara yang masuk mempunyai nilai sama. Dapat dilihat pada gambar 4.3.

```
public void normalisasi(double[] audioData, int
sampleMax) {
        for (int i = 0; i < audioData.length;
i++) {
            audioData[i] /= sampleMax;
        }
    }
    public void normalisasi(double[] audioData,
double sampleMax) {
        for (int i = 0; i < audioData.length;
i++) {
            audioData[i] /= sampleMax;
        }
    }
```

Gambar 4.3 *Source code* normalisasi.

4.3.2.3. *Pre-Emphasis.*

Untuk proses *pre-emphasis* untuk meratakan suara yang maasuk dengan nilai berkisar antara 0.96-0.99 dinyatakan dalam ALPHA_PRE dengan nilai yang diaambil 0.97. Digambarkan pada gambar 4.4.

```
public void preemphasis(double[] audioData) {
        for (int i = audioData.length - 1; i > 0;
i--) {
            audioData[i] -= ALPHA_PRE *
audioData[i - 1];
        }
    }
```

Gambar 4.4 *Source code Pre-Emphasis*.

4.3.3. Ekstraksi Ciri (MFCC).

Seteleh dilakukan proses awal merubah sinyal analog menjadi digital selanjutnya dilakukan proses ekstrasi ciri dengan metode

MFCC. *Frame Blockin. Windowing. Fast Fourier Transform* (FFT), *Mel Frequency filterbank, Discrete Fourier Transform* (DFT).

4.3.3.1. *Frame Blocking.*

Frame blocking adalah mengelompokkan sampel sinyal suara dengan *FRAME*_WIDTH sebesar 2^9=512 dengan *FRAME*_SHIFT sebesar 160 dimana *frame* ke-0 dan ke-1 terjadi *overlap*/tumpang tindih sebanyak *FRAME*_WIDTH - *FRAME*_SHIFT sampel, begitu seterusnya. *Source code frame blocking* gambar 4.5.

```
public double[][] frameBlocking(double[]
audioData){
        int frameCount = (audioData.length-
(FRAME_WIDTH-FRAME_SHIFT))/FRAME_SHIFT;
        int sampleLack = (audioData.length-
(FRAME_WIDTH-FRAME_SHIFT))%FRAME_SHIFT;
double[][]frameBlock=newdouble[frameCount][FRAME_W
IDTH];
        if(sampleLack > 0){
            frameCount++;
frameBlock=new double[frameCount][FRAME_WIDTH];
            for(int i=(frameCount-1)*FRAME_SHIFT;
i<audioData.length; i++){frameBlock[frameCount-
1][i-((frameCount-1)*FRAME_SHIFT)] = audioData[i];
            }
            for(int i=0; i<sampleLack; i++){
                    frameBlock[frameCount-
1][FRAME_WIDTH-
sampleLack+i]=audioData[audioData.length-
16+(i%16)];
            }
            frameCount--;
        }
        for(int i=0; i<frameCount; i++){
            for(int j=i*FRAME_SHIFT;
j<(i*FRAME_SHIFT)+512; j++){frameBlock[i][j-
i*FRAME_SHIFT)] = audioData[j];
            }
        }
        return frameBlock;
    }
```

gambar 4.5. *Source code frame blocking*

4.3.3.2. *Windowing.*

Pada windowing (Gambar 4.6) ini metode yang digunakan adalah *hammingWindow* dengan persamaan seperti pada 2.2, sinyal tersebut harus dikalikan dengan fungsi window yang tepat.

```
public void hammingWindow(double[][] frameBlock){
        double h = 2*Math.PI/(FRAME_WIDTH-1);
        for(int i=0; i<frameBlock.length; i++){
            for(int j=0; j<FRAME_WIDTH; j++){
     frameBlock[i][j] *= 0.54-0.46*Math.cos(h*j);
            }
        }
    }
```

Gambar 4.6. *Source codeWindowing.*

4.3.3.3. *Fast Fourier Transform* (FFT).

Pada gambar 4.7 sebelum pada tahap ini dilakukan pembuatan tabel dan bitReserve yang bertujuan untuk membalik *bit* sebelum dilakukan proses FFT *butterfly*.

```
public int[] createTable(int num){
   int[] table = new int[num];
   int bitSift = Integer.numberOfLeadingZeros(num-
1);
for(int i=0; i<num; i++{int
tmp=Integer.reverse(i);
            tmp = tmp >>> bitSift;
            table[i] = tmp;
        }
        return table;
    }
    public void bitReserve(double[][] frameBlock,
ComplexNumber[][] fftData){
        int[] table = createTable(FRAME_WIDTH);
        for(int i=0; i<frameBlock.length; i++){
            for(int j=0; j<FRAME_WIDTH;
j++){fftData [i][j]=new
ComplexNumber(frameBlock[i][table[j]]);
            }
        }    }
```

Gambar 4.7. *Source code* Craeate tabel dan bitReserve.

Pada proses FFT *butterfly* (gambar 4.8) tranformasi dilakukan menentukan bilangan ganjil dan genap seperti pada persamaan 2.8 dengan bilangan komplek. Dengan k lebih kecil dari *FRAME*_WIDTH-1 Transformasi Fourier ini dilakukan untuk mentransformaikan sinyal dari domain waktu ke domain frekuensi. Hal ini bertujuan agar sinyal dapat diproses dalam spectral substraksi. Operasi *butterfly* dua-dua kemudian naik menjadi empat dan naik lagi menjadi delapan atau dua kalinya.

```
public void FFT(double[][] frameBlock){
        ComplexNumber[][] fftData = new
ComplexNumber[frameBlock.length][FRAME_WIDTH];
        bitReserve(frameBlock, fftData);
        for(int i=0; i<fftData.length; i++){
            int m =
Integer.bitCount(FRAME_WIDTH-1);
            int n,k,x;
            double c,s;
            ComplexNumber tmp = new
ComplexNumber();
            for(int j=0; j<m; j++){
                n = (int)Math.pow(2, j);
                k = 0;
                while(k < FRAME_WIDTH-1){
                    for(int b=0; b<n; b++){
                        c =
Math.cos(b*Math.PI/n);
                        s = Math.sin(-
b*Math.PI/n);
                        x = k+b;
tmp = fftData[i][x+n].multiply(new
ComplexNumber(c, s));
fftData[i][x+n] = fftData[i][x].subtract(tmp);
fftData[i][x] = fftData[i][x].add(tmp);
                    }
                    k += 2*n;
                }
            }
            for(int j=0; j<FRAME_WIDTH; j++){
                frameBlock[i][j] =
Math.pow(fftData[i][j].divide(Math.sqrt(FRAME_WI
DTH)).magnitude(), 2);            }        }   }
```

Gambar 4.8. S*ource code Fast Fourier Transform.*

4.3.3.4. *Mel Frequency filterbank.*

Mel Frequency filterbank analisis frekuensi menggunakan *filter bank* dan hasilnya adalah frekuensi akan diskala menggunakan Mel-*filter bank H(k,m)* dengan perhitungan *mel* pada persamaan 2.9.

$Mel(f_l) = Mel(0) = 2595 \log_{10}(1+0/700) = 0$

$Mel(f_h) = Mel(40000) = 2595 \log_{10}(1+4000/700) = 2146.06453$

untuk ukuran FILTER_BANK sebesar 24 dalam perhitungan Mel-*filter bank H(k,m).* Ditunjukkan pada gambar 4.9.

```
public double[][]
melFrequencyFilterBank(double[][] frameBlock){
        double fh =
2595*Math.log10(1+(8000.0/700.0));
        double fl =
2595*Math.log10(1+(0.0/700.0));
        double fm[] = new double[FILTER_BANK+2];
        for(int i=0; i<FILTER_BANK+2; i++){
            fm[i] =
(double)FRAME_WIDTH/16000.0*700.0*(Math.pow(10,(i
*fh/(FILTER_BANK+1))/2595.0)-1);
        }
        double[][] melData = new
double[frameBlock.length][FILTER_BANK];
        for(int i=0; i<frameBlock.length; i++){
for(int m=1; m<=FILTER_BANK; m++){melData[i][m-1]
= 0;
for(int k=0; k<FRAME_WIDTH; k++){double
filterBank = 0;
if((k >= fm[m-1]) && (k < fm[m])){
filterBank = (2.0*(k-fm[m-1]))/((fm[m+1]-fm[m-
1])* (fm[m]-fm[m-1]));
} else if((k >= fm[m]) && (k < fm[m+1])){
filterBank = (2.0*(fm[m+1]-k))/((fm[m+1]-fm[m-
1])* (fm[m+1]-fm[m]));
                    }
 melData[i][m-1] += frameBlock[i][k]*filterBank;
                }           }        }
       return melData;    }
```

Gambar 4.9. *Source code Mel Frequency filterbank.*

4.3.3.5. *Discrete Fourier Transform* (DFT).

Pada gambar 2.10 DFT merubah kembali domain frekuensi menjadi waktu, selain itu juga menghasilkan koefisien-koefisien *cepstrum* yang akan digunakan untuk pengenalan suara. Dengan COEF_*CEPSTRUM* sebesar 12.

```
public double[][] DCT(double[][] melData){
       double[][]       dctData       =       new
double[melData.length]
[COEF_CEPSTRUM+1];
        for(int i=0; i<melData.length; i++){
            for(int n=1; n<=COEF_CEPSTRUM; n++){
                double logEnergi = 0;
                dctData[i][n] = 0;
                for(int r=1; r<=FILTER_BANK; r++){
dctData[i][n]     +=     Math.log10(melData[i][r-
1])*Math.cos
(n*Math.PI/FILTER_BANK*(r-0.5));
    logEnergi += Math.log10(melData[i][r-1]);
                }
                dctData[i][0]                    =
logEnergi/FILTER_BANK;
            }       }
         return dctData;    }
```

Gambar 4.10. *Source code Discrete Fourier Transform.*

4.3.4. Recognizing.

Dalam pengenalan suara ini terdapat 2 proses yaitu vektor kuantisasi dan penglasteran. Berikut struktur data yang dipakai dalam pengenalan suara ditunjukkan pada gambar 2.11.

```
private final String JDBC_DRIVER="com.mysql.jdbc.
Driver";
private final String
DATABASE_URL="jdbc:mysql://localhost/speaker_
recognition";
private Connection con;
private Statement stat;
private final int codebook = 16;
```

Gambar 4.11. *Source code* sturktur data pengenalan.

JDBC_DRIVER digunakan untuk koneksi pada *database*. DATABASE_URL berfungsi untuk menentukan alamat pada

database speaker_recognition. con dan stat sebagai fariabel penghubung *database. Codebook* penyimpan vektor dari data latih kemudian menyimpannya sebagai vektor kode.

4.3.4.1. *Vector Quantization.*

Kuantisasi vektor (gambar 2.12) adalah metode kuantisasi yang dimana spektral sinyal suara tidak dikuantisasi satu per satu melainkan vektor spektral dikuantisasi menjadi satu kode nilai yang mewakili vektor tersebut.

```
public VectorQuantization(){
        try {
            Class.forName(JDBC_DRIVER);
            con =
DriverManager.getConnection(DATABASE_URL, "root",
"");
            stat = con.createStatement();
        } catch (ClassNotFoundException e) {
            JOptionPane.showMessageDialog( null,
e.getMessage(), "JDBC DRIVER Error",
JOptionPane.ERROR_MESSAGE );
        } catch (SQLException e){
            JOptionPane.showMessageDialog( null,
e.getMessage(), "SQL Error",
JOptionPane.ERROR_MESSAGE );
        }
    }
```

Gambar 4.12. *Source code Vector Quantization.*

Setelah nilai kuantisasi diperoleh maka akan dilakukan inputan ke dalam *database source codex* ditunjukkan pada gambar2.13.

```
public void insertVectorCepstral(String name,
double[][] data){
        for(int i=0; i<data.length; i++){
            try {
stat.addBatch("INSERT INTO vector_cepstral VALUES
(NULL,'"+name+"',"+data[i][0]+","+data[i][1]+","+da
ta[i][2]+","+data[i][3]+","+data[i][4]+","+data[i][
5]+","+data[i][6]+","+data[i][7]+","+data[i][8]+","
+data[i][9]+","+data[i][10]+","+data[i][11]+","+dat
a[i][12]+","+data[i][13]+","+data[i][14]+","+data[i
```

```
][15]+","+data[i][16]+","+data[i][17]+","+data[i][1
8]+","+data[i][19]+","+data[i][20]+","+data[i][21]+
","+data[i][22]+","+data[i][23]+","+data[i][24]+","
+data[i][25]+","+data[i][26]+","+data[i][27]+","+da
ta[i][28]+","+data[i][29]+","+data[i][30
]+","+data[i][31]+","+data[i][32]+","+data[i][33]+"
,"+data[i][34]+","+data[i][35]+","+data[i][36]+","+
data[i][37]+","+data[i][38]+",NULL);");
} catch (SQLException e) {
JOptionPane.showMessageDialog( null,
e.getMessage(), "SQL Error",
JOptionPane.ERROR_MESSAGE );
            }
        }
        try {
            stat.executeBatch();
            stat.clearBatch();
        } catch (SQLException e) {
 JOptionPane.showMessageDialog( null,
e.getMessage(), "SQL Error",
JOptionPane.ERROR_MESSAGE );       }    }
```

Gambar 4.13. *Source code* insertVectorCepstral.

Untuk vektor *centroid* ditunjukkan pada gambar 4.14.

```
public void insertVector Centroid(String name,
double[][] data){
        for(int i=0; i<data.length; i++){
            int idx = i;
            try {
stat.addBatch("INSERT INTO centroid VALUES
("+idx+",'"+name+"','"+data[i][0]+","+data[i][1]+","
+data[i][2]+","+data[i][3]+","+data[i][4]+","+data[
i][5]+","+data[i][6]+","+data[i][7]+","+data[i][8]+
","+data[i][9]+","+data[i][10]+","+data[i][11]+","+
data[i][12]+","+data[i][13]+","+data[i][14]+","+dat
a[i][15]+","+data[i][16]+","+data[i][17]+","+data[i
][18]+","+data[i][19]+","+data[i][20]+","+data[i][2
1]+","+data[i][22]+","+data[i][23]+","+data[i][24]+
","+data[i][25]+","+data[i][26]+","+data[i][27]+","
+data[i][28]+","+data[i][29]+","+data[i][30]+","+
data[i][31]+","+data[i][32]+","+data[i][33]+","+
data[i][34]+","+data[i][35]+","+data[i][36]+","+dat
```

```
a[i][37]+","+data[i][38]+");");
} catch (SQLException e) {
 JOptionPane.showMessageDialog( null,
e.getMessage(), "SQL Error",
JOptionPane.ERROR_MESSAGE );
            }
        }       try {
stat.executeBatch();
            stat.clearBatch();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog( null,
e.getMessage(), "SQL Error",
JOptionPane.ERROR_MESSAGE );
        }
    }

} catch (SQLException e) {
 JOptionPane.showMessageDialog( null,
e.getMessage(), "SQL Error",
JOptionPane.ERROR_MESSAGE );
            }
        }       try {
stat.executeBatch();
            stat.clearBatch();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog( null,
e.getMessage(), "SQL Error",
JOptionPane.ERROR_MESSAGE );
        }
    }
```

Gambar 4.14. *Source code* insertVector*Centroid*.

4.3.4.2. *Clustering.*

Metode *clustering-based* biasanya menggunakan algoritma *k-means*, *k-means* adalah algoritma klustering yang akan membagi data menjadi *k* kategori yang setiap kategori terdapat *centroid* atau titik tengah kluster. Jika ada data baru maka data baru tersebut akan dibandingkan dengan semua *centroid.*

```
public void klustering1(String name){
        double[][] centroid = null;
        int cluster = 1;
        double split = 0.01;

        try {
PreparedStatement
updateVector=con.prepareStatement
("UPDATE vector_cepstral SET cluster = ? WHERE id
= ?;");
PreparedStatement
selectVector=con.prepareStatement
("SELECT * FROM vector_cepstral where
name='"+name+"';");
PreparedStatement avgVector =
con.prepareStatement ("SELECT
avg(c1),avg(c2),avg(c3),avg(c4),avg(c5),avg(c6),"
+"avg(c7),avg(c8),avg(c9),avg(c10),avg(c11),avg(c
12),avg(c13),avg(c14),avg(c15),"+"avg(c16),avg(c1
7),avg(c18),avg(c19),avg(c20),avg(c21),avg(c22),a
vg(c23),avg(c24),"+"avg(c25),avg(c26),avg(c27),av
g(c28),avg(c29),avg(c30),avg(c31),avg(c32),avg(c3
3),"+"avg(c34),avg(c35),avg(c36),avg(c37),avg(c38
),avg(c39) FROM vector_cepstral "+"WHERE name =
'"+name+"' and cluster = ?;");
while(cluster <= codebook){
        double[][] tempCentroid = null;
if(cluster == 1){tempCentroid = new
double[cluster][39];
rs = stat.executeQuery("SELECT * FROM
vector_cepstral where name='"+name+"' LIMIT
0,1;");
        for(int i=0; i<cluster; i++){
             rs.next();
for(int j=3; j<=41; j++){tempCentroid[i][j-3] =
rs.getDouble(j);
         }        }
} else{tempCentroid=new
double[centroid.length*2][39];
for(int i=0; i<centroid.length; i++){
for(int j=0; j<39; j++){tempCentroid[i*2][j] =
centroid[i][j]-split;
tempCentroid[i*2+1][j] = centroid[i][j]+split;
```

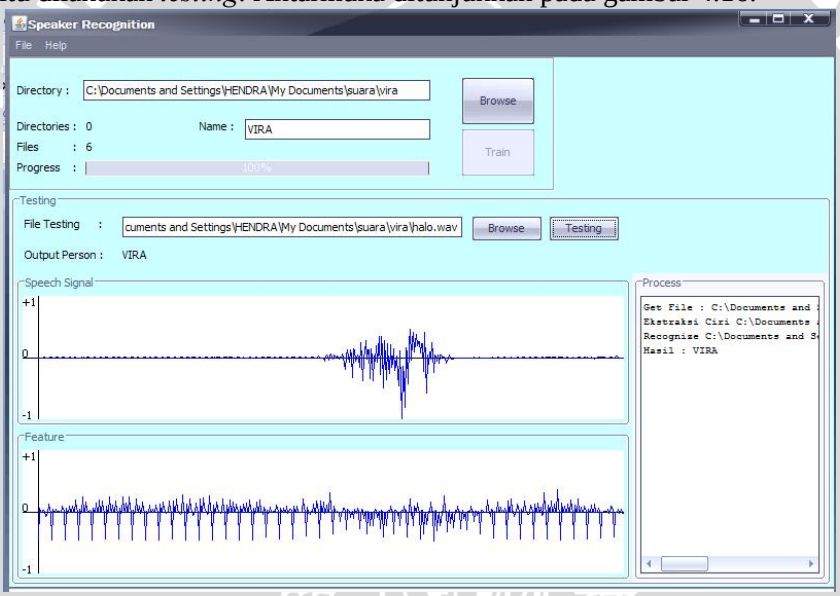
```
}                   }              }
boolean repeat;
int iteration = 0;
do{repeat = false;
rs = selectVector.executeQuery();
while(rs.next()){double[] distance=new
double[cluster];
for(int i=3; i<=41; i++){double
ceps=rs.getDouble(i);
for(int j=0; j<cluster; j++){distance[j] +=
((tempCentroid[j][i-3]-ceps)*(tempCentroid[j][i-3]-
ceps));
}                 }
double min = Double.MAX_VALUE;
int code = -1;
for(int i=0; i<cluster; i++){distance[i] =
Math.sqrt(distance[i]);
if(min > distance[i]){ min = distance[i];
code = i;
}     }
updateVector.setInt(1, code);
updateVector.setInt(2, rs.getInt(1));
updateVector.executeUpdate();}
for(int i=0; i<cluster; i++){
for(int j=1; j<=39; j++)
avgVector.setInt(1, i);
rs = avgVector.executeQuery();
rs.next();
if(rs.getDouble(j) != tempCentroid[i][j-1]){
tempCentroid[i][j-1] = rs.getDouble(j);
repeat = true;}}}
iteration++;
System.out.println("cluster-"+cluster+" : iterasi
ke-"+iteration);
} while(repeat);
centroid = tempCentroid.clone();
cluster *= 2; }
insertVectorCentroid(name, centroid);
} catch (SQLException e) {
JOptionPane.showMessageDialog( null, e.getMessage(),
"SQL Error", JOptionPane.ERROR_MESSAGE );
        }    }
```

Gambar 4.15 *Source code clustering.*

4.4. Implementasi Antarmuka.

Berdasarkan rancangan antarmuka pada sub bab 3.4 maka dihasilkan antar muka program utama yang ditunjukkan pada Gambar 4.16. Pada pojok kanan atas terdapat tombol *browse* mengambil data yang akan di_*training*, jLabel2 atau *directories* adalah jumlah folder pada pemanggilan data sedangkan jLabel3 atau *Files* adalah jumlah *file* yang akan di_*training*. Untuk tombol *training* merupaan proses ekstraksi ciri digunakan sebagai inputan data *learning* yang akan diuji disimpan pada *database*. Sedangkan tombol *browse* pada *file testing* digunakan sebagai *file* yang akan diuji, tombol testing merupakan proses pengenalan data *input*. *Graph speech signal* jPanel3 adalah sinyal analog *file* wav kemudian di bawah jPanel4 adalah sinyal setelah diekstraksi ciri. Untuk jTextArea1 *input* proses dari *file* ekstraksi ciri juga pengenalan saat pengenalan.

Cara kerja program pengenalan suara ini adalah pertama – tama *user* melakukan proses pengambilan data kemudian akan di_*training* yang akan dimasukkan pada *database* sebagai ekstraksi ciri. Kedua, pengambilan data yang akan diuji sebagai pengenalan suara, setelah itu dilakukan *testing*. Antarmuka ditunjukkan pada gambar 4.16.



Gambar 4.16. Antarmuka pengenalan suara.

4.5. Pengujian dan Analisa.

Pada subbab ini akan menjelaskan tentang pengujian dan analisa terhadap kinerja perangkat lunak yang telah dibuat. Hasil pengujian akan digunakan untuk mengamati pengaruh parameter yang digunakan dalam Algoritma FFT pada MFCC dan vektor kuantisai. Dalam pengujian ini di ambil dari 6 orang 3 laki – laki dan 3 perempuan. Masing – masing orang diambil 5 sampel data traning dan non traning jadi jumlah data 30 sampel data traning da 30 sampel data nontraning.

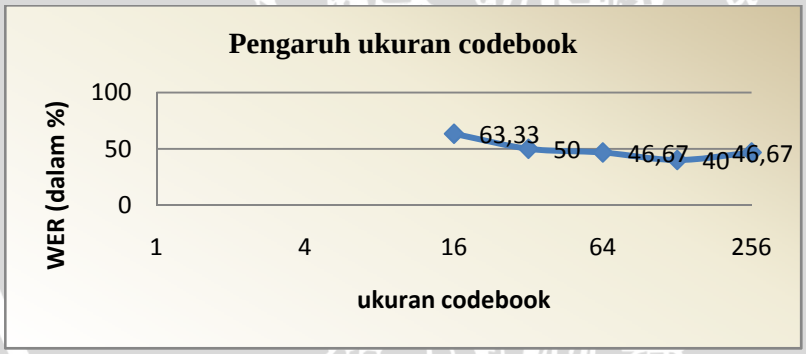
4.5.1. Hasil Uji Coba Pengaruh Ukuran *Codebook*.

Pengujian digunakan untuk mengetahui pengaruh ukuran *codebook* terhadap akurasi kata. Berdasarkan uji coba yang dilakukan selama lima kali, tidak terdapat perubahan terhadap hasil pengujian setiap perlakuan. Tabel 4.1 nilai pada setiap *codebook* dengan persentase WER pada data *learning*.

Tabel 4.1 Tabel pengaruh *codebook* pada data *learning*.

Ukuran	WER(dalam%)
16	63.33
32	50
64	46.67
128	40
256	46.67

Pengaruh pengujian ukuran *codebook* terhadap akurasi kata ditunjukkan pada gambar 4.17.



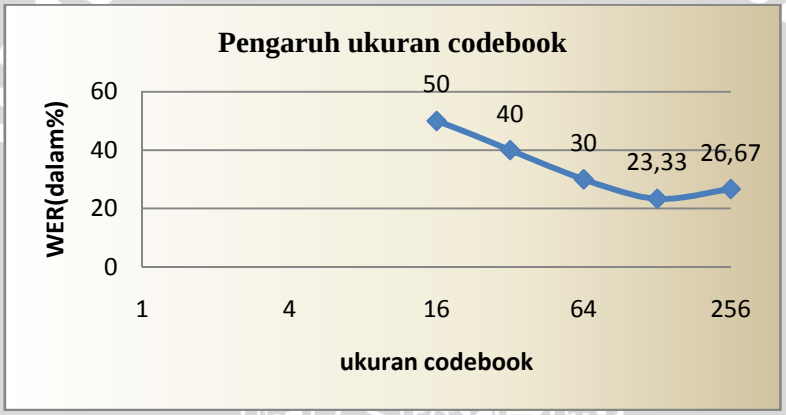
Gambar 4.17 Diagram pengaruh ukuran *codebook learning*.

Dari Gambar 4.17, dapat diketahui bahwa pada *codebook* 16 nilai WER adalah 63.33%, *codebook* 32 nilai WER 50%, *codebook* 64 nilai WER 46.67%, *codebook* 128 nilai WER 40% dan pada *codebook* 256 didapat nilai WER 47%. Pada *codebook* 16 sampai 128 nilai WER semakin kecil, sedangkan pada *codebook* 256 nilai WER mengalami peningkatan. Tabel 4.2 nilai pada setiap *codebook* dengan persentase WER pada data *non-learning*.

Tabel 4.2 Tabel pengaruh *codebook* pada data *non-learning*.

Ukuran	WER(dalam%)
16	50
32	40
64	30
128	23.33
256	26.67

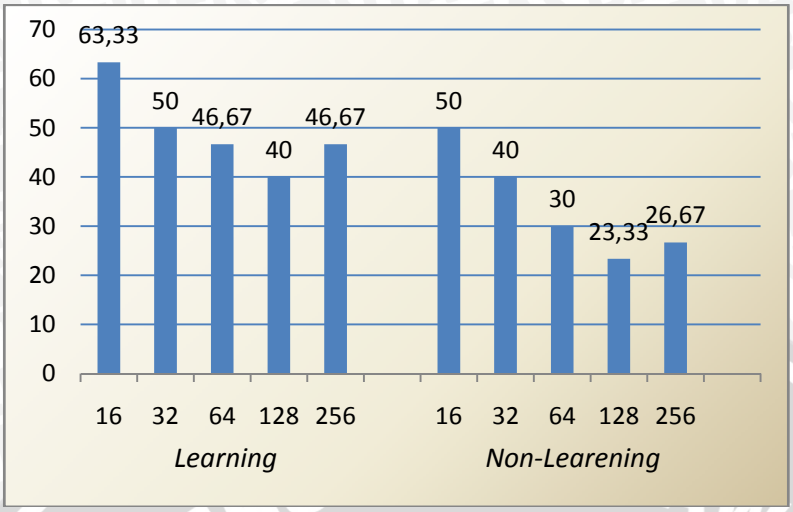
Pada pengujian *non-learning* dapat dilihat pada diagram gambar 4.18.



Gambar 4.18 Diagram pengaruh ukuran *codebook non-learning*.

Dari Gambar 4.18, dapat diketahui bahwa pada *codebook* 16 nilai WER adalah 50%, *codebook* 32 nilai WER 40%, *codebook* 64 nilai WER 30%, *codebook* 128 nilai WER 23.33% dan pada *codebook* 256 didapat nilai WER 26.67%.

Dengan demikian pada data *learning* dan *non-learning* data persentase keberhasilan dapat di gambarkan pada gambar 2.19.



Gambar 4.19. Persentase data *learning* dan *non-learning*.

Pada pengaruh *codebook* maka setiap *codebook* diperbesar akan berpengaruh pada setiap pengenalan suara, tetapi pada *codebook* yang lebih besar masih bisa mengenali identitas seseorang tetapi hanya pada perorangan tertentu dengan frekuensi sinyal suara yang besar. Jika ukuran *codebook* terlalu kecil daripada banyak variasi data maka ada kemungkinan untuk kelas suara yang sama dikelompokkan ke dalam satu *codebook*, dan jika ukuran *codebook* terlalu besar ada kemungkinan terdapat *codebook* yang tidak merepresentasikan kelas suara tertentu. Dengan demikian dapat disimpulkan bahwa hasil untuk pengenalan suara dilakukan dengan ukuran *codebook* 16 dengan WER sebesar 63.33%.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan.

Berdasarkan hasil uji coba yang telah dilakukan, maka dapat diambil kesimpulan antara lain :

1. Algoritma *Fast Fourier Transform* (FFT) Dengan Algoritma *Mel Frequency Cepstrum Coeffisient* (MFCC) Sebagai Ekstrasi Ciri dengan *vector quantization* sebagai pengenalan untuk menyelesaikan permasalahan sistem pengenalan suara dapat diimplementasikan pada aplikasi dengan berpengaruh pada besar *codebook*.
2. Hasil uji coba pengaruh *codebook* terhadap keakurasian sistem dihasilkan untuk data *learning* bahwa pada *codebook* 16 nilai WER adalah 63.33%, *codebook* 32 nilai WER 50%, *codebook* 64 nilai WER 46.67%, *codebook* 128 nilai WER 40% dan pada *codebook* 256 didapat nilai WER 47%. Pada *codebook* 16 sampai 128 nilai WER semakin kecil, sedangkan pada *codebook* 256 nilai WER mengalami peningkatan. Sedangkan untuk data *non-learning* diketahui bahwa pada *codebook* 16 nilai WER adalah 50%, *codebook* 32 nilai WER 40%, *codebook* 64 nilai WER 30%, *codebook* 128 nilai WER 23.33% dan pada *codebook* 256 didapat nilai WER 26.67%. Sehingga dapat disimpulkan bahwa semakin besar jumlah *codebook* tidak menjamin dapat meningkatkan keakurasian sistem.

5.2. Saran.

Saran untuk pengembangan lebih lanjut adalah:

1. Perekaman suara, khususnya untuk *database training* sebaiknya di dalam ruangan yang bebas *noise*.
2. Untuk penelitian selanjutnya tidak hanya untuk file berformat WAV tetapi juga file berformat yang lain seperti MP3, MIDI, AMR, dll.

DAFTAR PUSTAKA

Agustina Trifena Dame Saragih, Achmad Rizal, Rita Magdalena, 2009. PENENTUAN AKOR GITAR DENGAN MENGGUNAKAN ALGORITMA SHORT TIME FOURIER TRANSFORM, Departemen Teknik Elektro – Institut Teknologi Telkom, Bandung, Indonesia.

Ali Mustofa. 2007. Sistem Pengenalan Penutur dengan Metode Mel-frequency Wrapping, FT UB Malang, Indonesia.

Ario Muhammad Fanie. 2008. Pengenalan jenis ikan dengan metode HMM menggunakan DSK TMS320C6713, FT UI Jakarta, Indonesia.

Budi Setiawan, 2006. MODEL SINUSOIDA SECARA SEGMENTAL UNTUK PENGKODEAN SINYAL SUARA, Teknik Elektro, Universitas Katolik Soegijapranata , Semarang 50236, Indonesia.

Dessy Irmawati. 2006. Pendekatan analisis pola untuk mengetahui pengaruh karawitan campursari pada vokalisnya, dalam sistem skala *pentatonic* dan *diatonis*. Teknik Elektro, Universitas Wangsamanggala, Yogyakarta, Indonesia.

Elliot, D.F. and Rao, K.R. 1982. DFT (*Discrete Fourier Transform*) FFT (*Fast Fourier Transform*) Academic Press, New York.

Gresia Melisa. 2008. Pencocokan Pola Suara (*Speech Recognition*) Dengan Algoritma *Fast Fourier Transform* (FFT) Dan *Divide And Conquer* (DC), Institut Teknologi Bandung, Bandung.

Nurlaily, 2009. Pencocokan Pola Suara (*Speech Recognition*) Dengan Algoritma *Fast Fourier Transform* (FFT) Dan *Divide And Conquer* (DC), Univeritas Sumatra Utara.

Petra Christian. 2003
http://digilib.petra.ac.id/viewer.php?page=1&submit.x=19&submit.y=18&submit=next&qual=high&submitval=next&fname=%2Fjiunkpe%2Fs1%2Felkt%2F2003%2Fjiunkpe-ns-s1-2003-23499127-1392-digital_signal-chapter3.pdf
Diakses 15 Nofember 2010

LAMPIRAN 1

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3	v	
	halo4		v
	halo5		v
hendra	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
istianah	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
wahyu	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5		v
WER %		63.33	

Learning
Codebook 16

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
hendra	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
istianah	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2		v
	halo3	v	
	halo4		v
	halo5		v
wahyu	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5		v
WER %		50	

Learning
Codebook 32

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
hendra	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
istianah	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
wahyu	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5		v
WER %		46.67	

Learning
Codebook 64

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
hendra	halo1		v
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
istianah	halo1	v	
	halo2	v	
	halo3		v
	halo4	v	
	halo5	v	
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
wahyu	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5		v
WER %		40	

Learning
Codebook 128

Learning
Codebook 256

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
hendra	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
istianah	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
wahyu	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5		v
WER %		46.67	

Non Learning
Codebook 16

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
hendra	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
istianah	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2		v
	halo3	v	
	halo4		v
	halo5		v
wahyu	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5		v
WER %		63.33	

Non Learning
Codebook 32

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
hendra	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5	v	
istianah	halo1	v	
	halo2	v	
	halo3		v
	halo4		v
	halo5		v
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2		v
	halo3	v	
	halo4		v
	halo5		v
wahyu	halo1	v	
	halo2	v	
	halo3	v	
	halo4	v	
	halo5		v
WER %		40	

Non Learning
Codebook 64

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
hendra	halo1	v	
	halo2	v	
	halo3		v
	halo4	v	
	halo5	v	
istianah	halo1	v	
	halo2	v	
	halo3		v
	halo4		v
	halo5		v
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
wahyu	halo1	v	
	halo2	v	
	halo3		v
	halo4	v	
	halo5		v
WER %		30	

Non Learning
Codebook 128

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
hendra	halo1	v	
	halo2	v	
	halo3		v
	halo4	v	
	halo5		v
istianah	halo1	v	
	halo2	v	
	halo3		v
	halo4		v
	halo5		v
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
wahyu	halo1	v	
	halo2		v
	halo3		v
	halo4		v
	halo5		v
WER %		23.33	

Non Learning
Codebook 128

nama	kata	benar	salah
hadi	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
hendra	halo1	v	
	halo2	v	
	halo3		v
	halo4	v	
	halo5		v
istianah	halo1	v	
	halo2	v	
	halo3		v
	halo4		v
	halo5		v
lia	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
vira	halo1		v
	halo2		v
	halo3		v
	halo4		v
	halo5		v
wahyu	halo1	v	
	halo2	v	
	halo3		v
	halo4		v
	halo5		v
WER %		26.67	

DAFTAR PUSTAKA

Agustina Trifena Dame Saragih, Achmad Rizal, Rita Magdalena, 2009. PENENTUAN AKOR GITAR DENGAN MENGGUNAKAN ALGORITMA SHORT TIME FOURIER TRANSFORM, Departemen Teknik Elektro – Institut Teknologi Telkom, Bandung, Indonesia.

Ali Mustofa. 2007. Sistem Pengenalan Penutur dengan Metode Mel-frequency Wrapping, FT UB Malang, Indonesia.

Ario Muhammad Fanie. 2008. Pengenalan jenis ikan dengan metode HMM menggunakan DSK TMS320C6713, FT UI Jakarta, Indonesia.

Budi Setiawan, 2006. MODEL SINUSOIDA SECARA SEGMENTAL UNTUK PENGKODEAN SINYAL SUARA, Teknik Elektro, Universitas Katolik Soegijapranata , Semarang 50236, Indonesia.

Dessy Irmawati. 2006. Pendekatan analisis pola untuk mengetahui pengaruh karawitan campursari pada vokalisnya, dalam sistem skala *pentatonic* dan *diatonis*. Teknik Elektro, Universitas Wangsamanggala, Yogyakarta, Indonesia.

Elliot, D.F. and Rao, K.R. 1982. DFT (*Discrete Fourier Transform*) FFT (*Fast Fourier Transform*) Academic Press, New York.

Gresia Melisa. 2008. Pencocokan Pola Suara (*Speech Recognition*) Dengan Algoritma *Fast Fourier Transform* (FFT) Dan *Divide And Conquer* (DC), Institut Teknologi Bandung, Bandung.

Nurlaily, 2009. Pencocokan Pola Suara (*Speech Recognition*) Dengan Algoritma *Fast Fourier Transform* (FFT) Dan *Divide And Conquer* (DC), Univeritas Sumatra Utara.

Petra Christian. 2003
http://digilib.petra.ac.id/viewer.php?page=1&submit.x=19&su

bmit.y=18&submit=next&qual=high&submitval=next&fname
=%2Fjiunkpe%2Fs1%2Felkt%2F2003%2Fjiunkpe-ns-s1-
2003-23499127-1392-digital_signal-chapter3.pdf

Diakses 15 Nofember 2010

DAFTAR LAMPIRAN

Halaman