IDENTIFIKASI PEMBICARA *INDEPENDENT TEXT* PADA DATA *CLOSE-SET* DENGAN MENGGUNAKAN *MEL FREQUENCY CEPSTRAL COEFFICIENTS*

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

oleh :

INGGIH PERMANA
10751000018

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2011

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

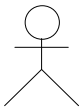
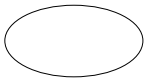
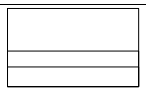
Gambar Halaman

DAFTAR ALGORITMA

DAFTAR RUMUS

DAFTAR LAMPIRAN

DAFTAR SIMBOL

SIMBOL	KETERANGAN SIMBOL
	Pengguna Sistem (Aktor)
	Proses (*Use case)*
	Aktivitas (*Activity*)
	Keputusan (*decision*)
	Status mulai (*start state*)
	Status selesai (*end state*)
	Kelas (*class*)

DAFTAR ISTILAH

A

Activity diagram : memodelkan aliran tindakan prosedural yang merupakan bagian dari kegiatan yang lebih besar pada pengembangan perangkat lunak.

C

Cepstrum : mengembalikan dari domain frekuensi ke domain waktu.

Class : konsep UML yang terdiri dari identitas *(identity), state,* prilaku *(behaviour),* serta relasi dengan kelas yang lainnya *(relationship).*

Close-set speaker : kemampuan untuk mengenali pembicara selama suara pembicara tersebut telah dipelajari oleh sistem.

Cluster : pembagian kelompok berdasarkan kemiripan.

Collaboration diagram : diagram interaksi yang menunjukkan penyampaian pesan-pesan atau interaksi (komunikasi) yang terjadi antar operasi-operasi atau transaksi-transaksi dalam kurun waktu tertentu pada sebuah sistem.

E

Euclidean distance : metode untuk mengukur jumlah kuadrat perbedaan nilai masing-masing variable

F

Fast fourier transform : metode untuk mengubah frame dari domain waktu ke domain frekuensi.

Frame blocking : membagi sinyal yang masuk kedalam beberapa frame.

M

Mel Frequency Wrapping : metode untuk untuk mendapatkan skala mel.

MFCC : Cara ekstrasi ciri suara yang paling sering digunakan, karena cara kerjanya didasarkan pada perbedaan frekuensi yang dapat ditangkap oleh telinga manusia.

O

Open-set speaker : pengenalan pembicara yang dapat dimulai dengan pembicara yang belum dipelajari sebelumnya.

R

Reinforced Learning : metode jaringan saraf dengan target tersedia, tetapi bukan sebagai penentu jawaban, hanya sebagai pengindikasi *output* yang dihasilkan benar atau salah.

S

Sequence diagram : gambaran interaksi sebagai diagram dua dimensi (matra) yang terdiri dari matra vertikal dan horizontal.

SOM : salah satu tipe dari metode jaringan saraf tiruan yang mana sistem pembelajarannya menggunakan *unsupervised learning* untuk menghasilkan dimensi rendah (pada umumnya dua dimensi).

Speaker recognition : adalah kemampuan komputer untuk mengidentifikasi pembicara dengan menggunakan karakteristik suara yang terdapat pada pembicara.

State chart diagram : perjalanan pengklasifikasi dalam perjalanan waktu.

Supervised Learning : proses belajar terawasi.

T

Text dependent : pengenalan suara yang dibatasi pada kata-kata yang identik pada waktu pelatihan saja

Text independent : pengenalan pembicara yang tidak terpengaruh terhadap kandungan kata-kata yang akan dikenalinya.

U

UML : bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek.

Unified Software Development Process (USDP) : salah satu metode pemodelan objek dalam pengembangan sistem/ perangkat lunak berbasis OOP yang menggunakan UML *(Unified Modeling Language)* sebagai *tool* utamanya.

Unsupervised Learning : proses belajar tak terawasi

Use cases : model fungsionalitas sistem/ perangkat lunak yang dilihat dari sisi pengguna yang ada diluar sistem, namun memiliki koherensi terhadap sistem/ perangkat lunak tersebut yang diekspresikan sebagai transaksi-transaksi yang terjadi antara aktor dan sistem.

W

Windowing : menghaluskan masing-masing *frame* untuk menimalkan sinyal tidak kontinu pada awal dan akhir masing-masing *frame*.

TEXT INDEPENDENT SPEAKER IDENTIFICATION ON CLOSE-SET DATA USING MEL FREQUENCY CEPSTRAL COEFFICIENTS

INGGIH PERMANA
10751000018

Date of Final Exam : July 1, 2011
Graduation Ceremony Period : October , 2011

Informatics Departement
Faculty of Sciences and Technology
State Islamic University of Sultan Sarif Kasim Riau

ABSTRACT

Speakers identification is one part of digital signal processing in the field of voice processing. This topic raised because limitations of the people to recognizing human sound that very various and many voice very similar among one people with another people. Speaker recognition that discussed in this research is the indenpendent text on close-set data. In this research uses MFCCs as feature extraction and SOMs as feature matching. This research used 10 people as a source of voice. Test results show the number of MFCC coefficients and the number of iteration can improve the accuracy of speaker identification with the speaker detection result by 80% and average accuracy of 44%.

Keywords: *Speaker Identification, MFCCs, SOMs.*

IDENTIFIKASI PEMBICARA *INDEPENDENT TEXT* PADA DATA *CLOSE-SET* DENGAN MENGGUNAKAN *MEL FREQUENCY CEPSTRAL COEFFICIENTS*

INGGIH PERMANA
10751000018

Tanggal Sidang : 1 Juli 2011
Periode Wisuda : Oktober 2011

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Sarif Kasim Riau

ABSTRAK

Identifikasi pembicara adalah salah satu bagian pemrosesan sinyal *digital* dibidang pengolahan suara. Topik ini diangkat mengingat keterbatasan manusia dalam mengenali suara manusia yang begitu banyak ragamnya serta banyak suara yang hampir sama antara manusia satu dengan manusia lainnya. Pengenalan pembicara yang dibahas dalam penelitian ini adalah pengenalan pembicara *indenpendent text* pada data *close-set*. Dalam penelitian ini menggunakan MFCC sebagai ekstraksi ciri dan SOM sebagai pengenalan pola. Penelitian ini menggunakan 10 orang sebagai sumber suaranya. Dalam penelitian ini telah dilakukan pengujian terhadap beberapa pengaruh parameter terhadap ketepatan identifikasi pembicara, yaitu jumlah iterasi, jumlah koefesien MFCC dan jumlah data latih. Hasil pengujian memperlihatkan jumlah koefesien MFCC dan jumlah iterasi dapat meningkatkan ketepatan identifikasi pembicara dengan hasil pendeteksian pembicara tertinggi sebanyak 80% dan rata-rata akurasi 44%.

Kata Kunci : Identifikasi Pembicara, MFCC, SOM.

DAFTAR ISI

DAFTAR ISTILAH

A

Activity diagram : memodelkan aliran tindakan prosedural yang merupakan bagian dari kegiatan yang lebih besar pada pengembangan perangkat lunak.

C

Cepstrum : mengembalikan dari domain frekuensi ke domain waktu.

Class : konsep UML yang terdiri dari identitas *(identity), state,* prilaku *(behaviour),* serta relasi dengan kelas yang lainnya *(relationship).*

Close-set speaker : kemampuan untuk mengenali pembicara selama suara pembicara tersebut telah dipelajari oleh sistem.

Cluster : pembagian kelompok berdasarkan kemiripan.

Collaboration diagram : diagram interaksi yang menunjukkan penyampaian pesan-pesan atau interaksi (komunikasi) yang terjadi antar operasi-operasi atau transaksi-transaksi dalam kurun waktu tertentu pada sebuah sistem.

E

Euclidean distance : metode untuk mengukur jumlah kuadrat perbedaan nilai masing-masing variable

F

Fast fourier transform : metode untuk mengubah frame dari domain waktu ke domain frekuensi.

Frame blocking : membagi sinyal yang masuk kedalam beberapa frame.

M

Mel Frequency Wrapping : metode untuk untuk mendapatkan skala mel.

MFCC : Cara ekstrasi ciri suara yang paling sering digunakan, karena cara kerjanya didasarkan pada perbedaan frekuensi yang dapat ditangkap oleh telinga manusia.

O

Open-set speaker : pengenalan pembicara yang dapat dimulai dengan pembicara yang belum dipelajari sebelumnya.

R

Reinforced Learning : metode jaringan saraf dengan target tersedia, tetapi bukan sebagai penentu jawaban, hanya sebagai pengindikasi *output* yang dihasilkan benar atau salah.

S

Sequence diagram : gambaran interaksi sebagai diagram dua dimensi (matra) yang terdiri dari matra vertikal dan horizontal.

SOM : salah satu tipe dari metode jaringan saraf tiruan yang mana sistem pembelajarannya menggunakan *unsupervised learning* untuk menghasilkan dimensi rendah (pada umumnya dua dimensi).

Speaker recognition : adalah kemampuan komputer untuk mengidentifikasi pembicara dengan menggunakan karakteristik suara yang terdapat pada pembicara.

State chart diagram : perjalanan pengklasifikasi dalam perjalanan waktu.

Supervised Learning : proses belajar terawasi.

T

Text dependent : pengenalan suara yang dibatasi pada kata-kata yang identik pada waktu pelatihan saja

Text independent : pengenalan pembicara yang tidak terpengaruh terhadap kandungan kata-kata yang akan dikenalinya.

U

UML : bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek.

Unified Software Development Process (USDP) : salah satu metode pemodelan objek dalam pengembangan sistem/ perangkat lunak berbasis OOP yang menggunakan UML *(Unified Modeling Language)* sebagai *tool* utamanya.

Unsupervised Learning : proses belajar tak terawasi

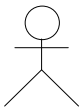
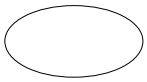
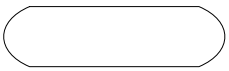
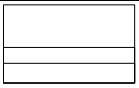
Use cases : model fungsionalitas sistem/ perangkat lunak yang dilihat dari sisi pengguna yang ada diluar sistem, namun memiliki koherensi terhadap sistem/ perangkat lunak tersebut yang diekspresikan sebagai transaksi-transaksi yang terjadi antara aktor dan sistem.

W

Windowing : menghaluskan masing-masing *frame* untuk menimalkan sinyal tidak kontinu pada awal dan akhir masing-masing *frame*.

DAFTAR RUMUS

DAFTAR SIMBOL

SIMBOL	KETERANGAN SIMBOL
	Pengguna Sistem (Aktor)
	Proses (*Use case)*
	Aktivitas (*Activity*)
	Keputusan (*decision*)
	Status mulai (*start state*)
	Status selesai (*end state*)
	Kelas (*class*)

DAFTAR GAMBAR

Gambar Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Identifikasi pembicara adalah cabang dari pengenalan pembicara yang merupakan salah satu bagian pemrosesan sinyal *digital* dibidang pengolahan suara. Penelitian ini akan membahas identifikasi pembicara pada *independent text*. Identifikasi pembicara adalah menentukan pembicara dari data-data suara yang telah terdaftar sebelumnya (Park, 2002).

Topik pengenalan pembicara ini diangkat mengingat keterbatasan manusia dalam mengenali suara manusia yang begitu banyak ragamnya serta banyak suara yang hampir sama antara manusia satu dengan manusia lainnya. Dalam kehidupan sehari-hari pengenalan pembicara juga sangat penting, contohnya dalam mengidentifikasi siapa yang berbicara pada rekaman percakapan yang dijadikan bukti pada suatu kasus di persidangan.

Sebuah penelitian tentang pengenalan suara telah dilakukan oleh Muda, Begam dan Elamvazuthi pada tahun 2010. Mereka menguji keakurasian pengenalan suara dengan menggunakan kombinasi MFCC (*Mel Frequency Cepstral Coeffecients*) dan DTW (*Dynamic Time Warping*). Dalam penelitian itu didapatkan bahwa kombinasi antara kedua metode tersebut dapat memberikan hasil yang efektif. Mereka menyarankan bahwa metode lainnya seperti jaringan saraf tiruan juga perlu diselidiki. Penelitian ini merupakan tindak lanjut dari saran tersebut. Penelitian ini mencoba menguji ketepatan dari kombinasi metode MFCC dengan salah satu metode jaringan saraf tiruan yaitu SOM.

Seperti yang telah dijelaskan pada paragraf sebelumnya, tugas akhir ini akan menguji ketepatan identifikasi pembicara dengan menggunakan kombinasi metode MFCC (*Mel Frequency Cepstral Coefficients*) untuk ekstraksi ciri dan SOM (*Self Organizing Maps*) sebagai pecocokan ciri. MFCC merupakan cara yang paling sering digunakan, karena cara kerjanya didasarkan pada perbedaan frekuensi yang dapat ditangkap oleh telinga manusia (Ganchev, 2005). SOM

sendiri merupakan suatu lapisan yang berisi neuron-neuron akan menyusun dirinya sendiri berdasarkan *input* nilai tertentu dalam suatu kelompok (Kesumadewi, 2003).

Penelitian ini akan menggunakan 10 orang pembicara yang terdiri dari 5 orang pria dan 5 orang wanita. Masing-masing orang akan diambil *sample* suaranya selama 20 detik sebanyak 2 kali. Satu *sample* suara akan dijadikan data pelatihan sedangkan satunya lagi akan dijadikan data pengujian.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas dapat diambil suatu rumusan masalah yang akan dibahas yaitu: bagaimana mengidentifikasi pembicara *independent text* dengan mengkombinasikan metode MFCC dan SOM pada data *close-set*.

1.3 Batasan Masalah

Agar pembahasan sesuai dengan tujuan penulisan, maka ruang lingkup batasan masalah yang disajikan adalah sebagai berikut:

1. Pembicara yang digunakan dalam penelitian ini adalah 10 orang yang terdiri dari 5 orang pria dan 5 orang wanita.
2. Bahasa pemograman yang digunakan adalah bahasa pemograman Java.
3. Metode jaringan saraf yang digunakan untuk pengenalan pola adalah dengan menggunakan SOM.
4. Suara pelatihan dan pengujian telah direkam sebelumnya

1.4 Tujuan

Tujuan dalam tugas akhir ini sebagai berikut:

1. Mempelajari dan menerapkan metode MFCC untuk ektrasi ciri pada suara.
2. Mempelajari dan menerapkan model jaringan syaraf tiruan SOM untuk identifikasi suara manusia.

3. Menguji akurasi kombinasi metode MFCC dan SOM untuk identifikasi pembicara.

1.5 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari enam bab, dengan sistematika penulisan sebagai berikut:

Bab I Pendahuluan

Bab ini berisi tentang deskripsi umum dari Tugas Akhir ini, yang meliputi latar belakang permasalahan, rumusan masalah, batasan masalah, tujuan dari pembahasan dan sistematika penulisan Tugas Akhir.

Bab II Landasan Teori

Bab ini akan membahas teori-teori yang berhubungan dengan pembahasan tugas akhir ini. Teori yang diangkat yaitu mengenai teori-teori tentang pengenalan pembicara,metode MFCC, jaringan saraf tiruan serta metode *Unified Software Development Process*.

Bab III Metodologi Penelitian

Bab ini akan membahas tentang metodologi penelitian yang digunakan dalam penyusunan tugas akhir ini. Pada bab ini akan dibahas tahapan-tahapan penelitian yang digunakan dalam tugas akhir ini yang terdiri dari penelitian pendahuluan, data *requirements*, analisa, implementasi dan pengujian serta analisa hasil.

Bab IV Analisis dan Perancangan

Berisikan tentang analisa penyelesaian masalah, perancangan aplikasi, cara menggunakan MFCC dan SOM dalam tugas akhir ini serta perancangan struktur menu dan perancangan antar muka.

Bab V Implementasi dan Pengujian

Bab ini berisi penjelasan mengenai batasan implementasi, lingkungan implementasi dan hasil dari pengujian.

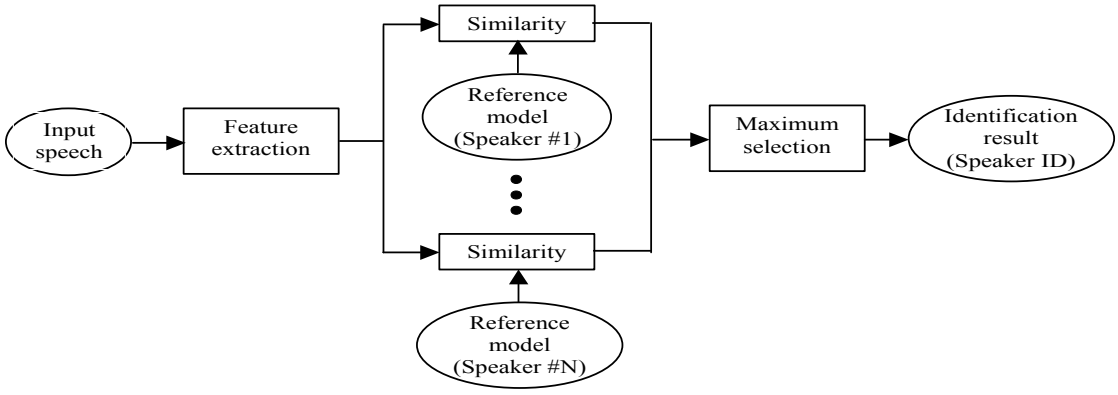
Bab VI Penutup

Bab ini berisikan kesimpulan dari Tugas Akhir yang dibuat dan menjelaskan saran-saran penulis kepada pembaca.

BAB II
LANDASAN TEORI

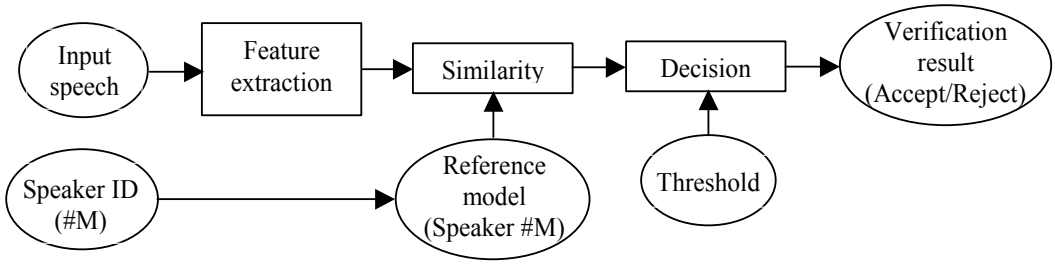
2.1 Pengenalan Pembicara

Pengenalan pembicara (*speaker recognition*) adalah kemampuan komputer untuk mengidentifikasi pembicara dengan menggunakan karakteristik suara yang terdapat pada pembicara (www.ehow.com/list_7690647_speaker-recognition-methods.html). Sinyal suara sendiri merupakan gelombang yang tercipta dari tekanan udara yang berasal dari paru-paru yang berjalan melewati lintasan suara menuju mulut dan rongga hidung, sehingga memiliki frekuensi yang berbeda (Al-Akaidi, 2007). Menurut Alex S. Park pengenalan pembicara mempunyai dua tujuan, yaitu untuk identifikasi dan verifikasi. Identifikasi pembicara adalah menentukan pembicara dari data-data suara yang telah terdaftar sebelumnya sedangkan verifikasi pembicara adalah pencocokan suara yang dimasukkan dengan suara yang suara yang dijadikan referensi.



Gambar 2.1. Identifikasi pembicara

(www.ifp.illinois.edu/~minhdo/teaching/speaker_recognition/speaker_recognition.html)



Gambar 2.2. Verifikasi pembicara

(www.ifp.illinois.edu/~minhdo/teaching/speaker_recognition/speaker_recognition.html)

Suara yang akan dikenali berdasarkan pembicaranya dapat berupa *close set speaker* atau *open set speaker*. Jika sistem disajikan dengan kemampuan untuk mengenali pembicara selama suara pembicara tersebut telah dipelajari oleh sistem maka hal tersebut dinamakan *close set speaker* sedangkan jika pengenalan pembicara dapat dimulai dengan pembicara yang belum dipelajari sebelumnya hal tersebut dinamakan *open set speaker* (www.speaker-recognition.org/).

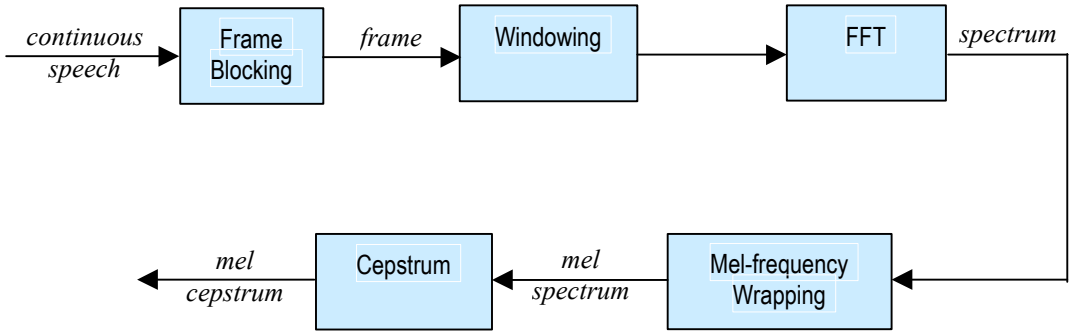
Pengenalan pembicara berdasarkan ketergantungannya terhadap variasi teks dibagi menjadi dua yaitu *text independent* dan *text dependent*. *Text independent* adalah pengenalan pembicara yang tidak terpengaruh terhadap kandungan kata-kata yang akan dikenali sedangkan *text dependent* adalah pengenalan suara yang dibatasi pada kata-kata yang identik pada waktu pelatihan saja (http://www.speaker-recognition.org/).

2.2 *Mel Frequency Cepstral Coefficients*

Bagian yang penting dari pengenalan suara adalah ektrasi ciri (*feature extraction*). Kegunaan dari ekstrasi ciri adalah mengurangi informasi-informasi yang tidak dibutuhkan dari sensor dan mengkonversi informasi-informasi yang penting dari signal untuk pengenalan pola agar menghasilkan format yang lebih sederhana dengan kelas-kelas yang jelas (http://www.speaker-recognition.org/navAlg.html). Berdasarkan penelitian Magnus Rossel, Mel Frequency Cepstral Coefficients

(MFCC) adalah metode ekstrasi mempunyai performa yang lebih bagus dari metode lainnya seperti dalam hal pengurangan *error rate*. MFCC merupakan cara yang paling sering digunakan, karena cara kerjanya didasarkan pada perbedaan frekuensi yang dapat ditangkap oleh telinga manusia (Ganchev , 2005).

MFCC memiliki beberapa tahap untu ekstrasi suara, berikut tahapannya:



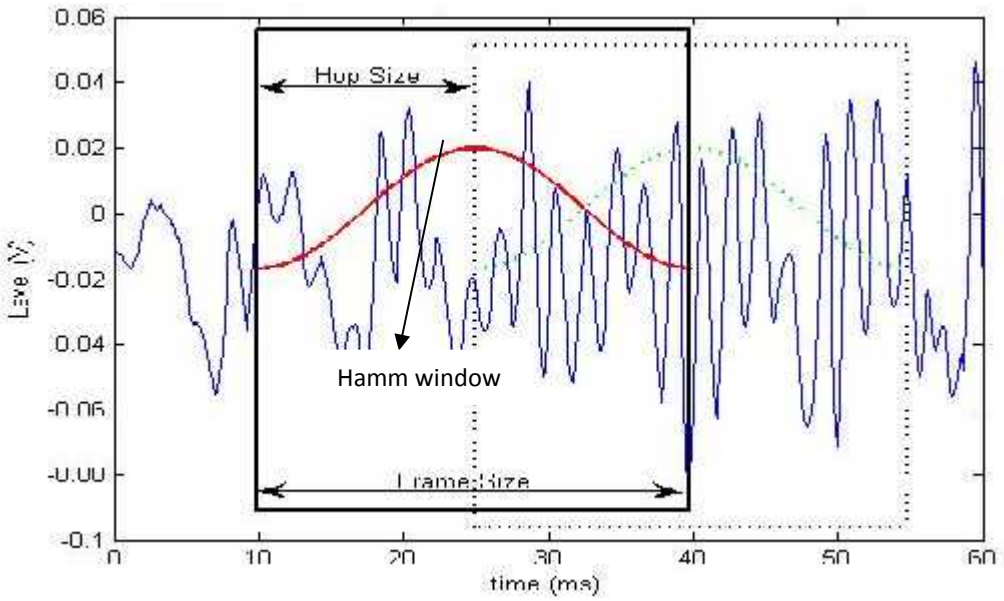
Gambar 2.3. Blok diagram tahapan MFCC

(www.ifp.illinois.edu/~minhdo/teaching/speaker_recognition/speaker_recognition.html)

2.2.1 *Frame Blocking*

Sinyal suara merupakan sinyal yang bervariasi dalam fungsi waktu, dalam hal ini ketika diamati pada durasi yang sangat pendek kurang lebih 5 sampai 100 ms mempunyai karakteristiknya masih stasioner. Dalam implementasi dalam program sinyal suara dibaca dalam *byte-byte* yang masuk yang mana 512 *byte* mewakili 32 ms.

Frame blocking adalah membagi sinyal yang masuk kedalam beberapa frame. (www.ifp.illinois.edu/~minhdo/teaching/speaker_recognition/speaker_recognition.html). Pada tahap ini kita melakukan *overlopping*. Biasanya *overlopping* dimulai dari ukuran *frame* dibagi dengan dua yang disebut dengan istilah *hopsize*. Untuk lebih jelasnya perhatikan gambar 2.4.



Gambar 2.4. *Frame blocking*

Gambar 2.4 memperlihatkan contoh *frame blocking*. Sumbu horizontal Menerangkan suara yang masuk berdasarkan satuan waktu (milli secon). Sedangkan Sumbu vertikal menunjukan level (volt) dari suara yang masuk. Pada gambar tersebut terlihat *frame size* adalah 30 ms dan *hop size* nya adalah 15 ms. Pada gambar tersebut juga terlihat garis melengkung yang disebut *Hamming Window*. *Hamming Window* akan dibahasa pada pada sub bab 2.2.2.

2.2.2 *Windowing*

Langkah selanjutnya adalah menghaluskan masing-masing *frame* untuk menimalkan sinyal tidak kontinu pada awal dan akhir masing-masing *frame*. Hal ini dilakukan dengan *Hamming Window*, berikut rumusnya. (Lundsgaard, 2006)

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1 \quad \text{.....(2.1)}$$

Dimana :

w(n) = *window*

N = jumlah *sample* pada masing-masing *frame*.

2.2.3 *Fast Fourier Transform*

Langkah selanjutnya adalah *fast fourier transform* (FFT). Hal ini dilakukan untuk mengubah frame dari domain waktu ke domain frekuensi. Hasil dari langkah ini disebut spectrum. Berikut rumus untuk FFT. (Lundsgaard, 2006)

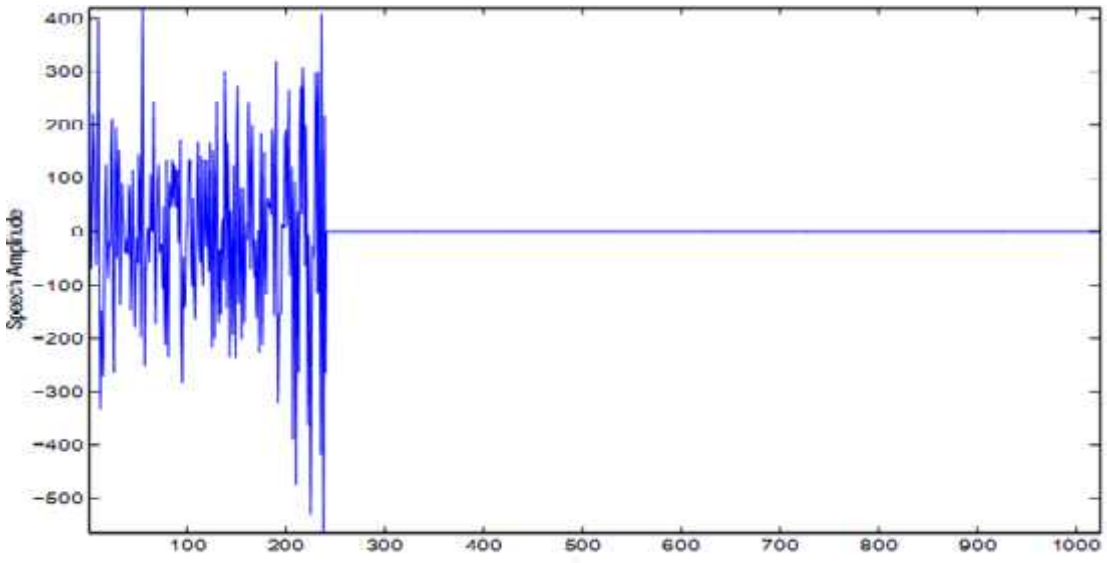
$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N}, \qquad k = 0,1,2,...,N-1 \quad \text{.....(2.2)}$$

Dimana :

Xk = frekuensi

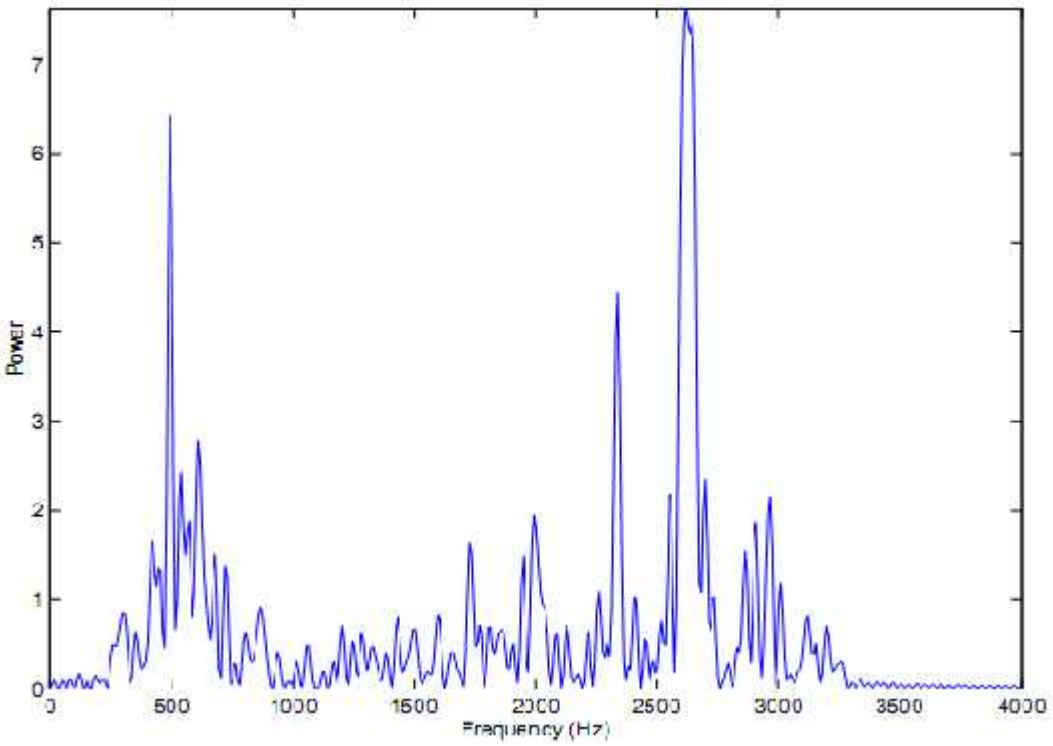
N = jumlah *sample* pada masing-masing *frame*.

Untuk lebih jelasnya perhatikan gambar 2.5 dan gambar 2.6. Gambar 2.5 adalah gambar sinyal suara yang belum dilakukan FFT sedangkan gambar 2.6 adalah gambar sinyal suara yang sudah dilakukan proses FFT.



Gambar 2.5. ***Sinyal suara sebelum proses FFT***

Sinyal suara pada gambar 2.5. masih dalam bentuk domain waktu yang mana sumbu horizontal mewakili suara yang masuk berdasarkan satuan waktu (milisecond) sedangkan sumbu vertikal adalah perwakilan level amplitudonya.



Gambar 2.6. ***Sinyal suara setelah proses FFT***

Pada gambar 2.6. adalah sinyal suara yang telah dirubah ke domain frekuansi melalui proses FFT. Sumbu horizontal mewakili frekuensi suara sedangkan sumbu vertikal mewakili *power* (db) suatu sinyal suara. Semua nilai db setelah proses FFT bernilai positif.

2.2.4 *Mel Frequency Wrapping*

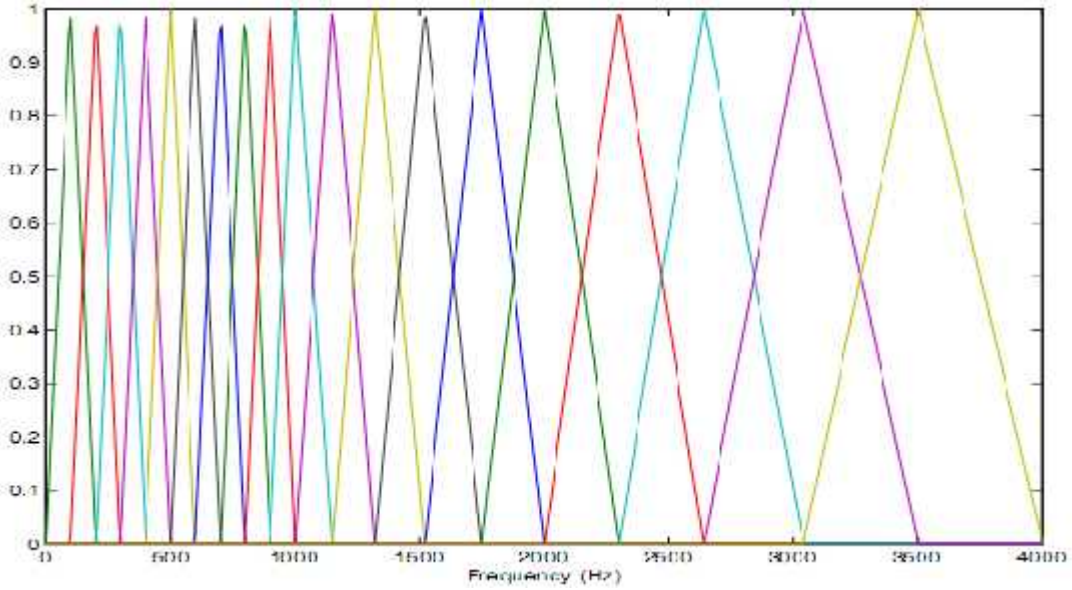
Untuk menggunakan metode ini frekuensi dejadikan dalam skala Mel. (Lundsgaard, 2006)

$$Mel(f) = 2595 \log 10(1 + \frac{f}{700}) \quad \text{..... (2.3)}$$

$$Freq(m) = 700(10^{\frac{m}{2595}} - 1) \quad \text{..... (2.4)}$$

$$\tilde{S}(l) - \sum_{k=0}^{N/2} S(k) M_l(k) \quad \text{..... (2.5)}$$

Persamaan 2.3 dan 2.4 adalah untuk mendapatkan *mel filter bank*, untuk mendapatkan *spectrum* dengan persamaan 2.5. *Mel filter bank* diperlihatkan pada gambar 2.7.



Gambar 2.7. *Mel filter bank*

Terlihat pada gambar 2.7 *mel filter bank* merupakan segitiga tumpang tindih. Pada gambar tersebut sumbu horizontal adalah mewakili frekuensi dan sumbu vertikal mewakili db sinyal suara.

2.2.5 *Cepstrum*

Langkah terkahir dari proses ini adalah mengembalikan dari domain frekuensi ke domain waktu. Hasil dari langkah ini disebut MFCC. Berikut rumus untuk mencarinya. (Lundsgaard, 2006)

$$\tilde{c}_n = \sum_{k=1}^{K} (\log \tilde{S}_k) \cos\left[n\left(k - \frac{1}{2} \right) \frac{\pi}{K} \right], \qquad n = 0,1,\ldots, K\text{-}1$$

..... (2.6)

Dimana

$\tilde{C}_n$ = MFCC

$\tilde{S}_n$ = mel frekuensi

2.3 Jaringan Saraf Tiruan

Pada tahun 1943 oleh seorang ahli saraf Warren Mc Culloch dan seorang ahli logika Walter Pits merancang model formal yang pertama kali sebagai perhitungan dasar *neuron*. (Hermawan, 2006). Jaringan Saraf Tiruan adalah sebuah kelompok pengolahan elemen dalam suatu kelompok yang khusus membuat perhitungan sendiri dan memberikan hasilnya kepada kelompok kedua atau berikutnya. Setiap sub-kelompok menurut gilirannya harus membuat perhitungan sendiri dan memberikan hasilnya untuk *subgroup* atau kelompok yang belum melakukan perhitungan. Pada akhirnya sebuah kelompok dari satu atau beberapa pengolahan elemen tersebut menghasilkan keluaran (*output*) dari jaringan (Rao dan Rao, 1993).

Jaringan saraf tiruan (JST) (*artificial neural network (ANN)*, atau juga disebut *simulated neural network (SNN)*, atau umumnya hanya disebut *neural network (NN)*), adalah sekelompok jaringan saraf *(neuron)* buatan yang menggunakan model matematis atau komputasi untuk pemrosesan informasi berdasarkan pendekatan terhubung pada komputasi. Pada kebanyakan kasus, JST merupakan sistem adaptif yang merubah strukturnya berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. JST dapat digunakan untuk memodelkan hubungan yang kompleks antara *input* dan *output* untuk menemukan pola-pola data. (http://id.wikipedia.org/wiki/Jaringan_saraf_tiruan).

2.3.1 Algoritma Belajar pada Jaringan Saraf Tiruan

Ada tiga jenis algoritma belajar, yaitu (Jha, 2007) :

1. *Supervised Learning* (proses belajar terawasi), pada pembelajaran ini target tersedia (diberikan oleh penyelia/ guru). Contohnya antara lain *Delta Rule* (Widrow dan Hoff, 1960), algoritma propagasi balik (Rumelhart dan McClelland, 1986).
2. *Unsupervised Learning* (proses belajar tak terawasi). Algoritma ini sama sekali tidak menggunakan data target (tanpa target). Kohonen (1984)

mengembangkan pelatihan *unsupervised* (*unsupervised learning*). Pada algoritma belajar ini, tidak membutuhkan target untuk keluarannya.

3. *Reinforced Learning,* pada metode ini target tersedia, tetapi bukan sebagai penentu jawaban, hanya sebagai pengindikasi *output* yang dihasilkan benar atau salah. Metode tidak termasuk pembelajaran yang popular digunakan.

2.3.2 *Euclidean Distance*

Euclidean distance, yaitu mengukur jumlah kuadrat perbedaan nilai masing-masing variable (Budhi, 2006).

$$d = \sqrt{\sum_{i}^{n} (Wi - Xi)^2} \quad \text{..... (2.7)}$$

Dimana:

d = jarak *Euclidian*

Wi= vektor bobot ke-i

Xi= vektor *input* ke Xi

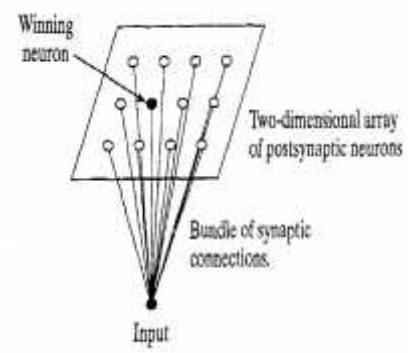
Semakin kecil nilai d, semakin besar kesamaan antara kedua obyek atau kasus tersebut, dan sebaliknya, semakin besar nilai d, semakin kecil kesamaan antara dua obyek.

2.3.3 *Self Organizing Maps*

Self Organizing Maps (SOM), atau *Kohonen*, merupakan salah satu tipe dari *artifcial neural network* yang mana sistem pembelajarannya menggunakan *unsupervised learning* untuk menghasilkan dimensi rendah (pada umumnya dua dimensi). *Unsupervised learning* adalah *input* disini tidak terdefinisikan. model ini tidak membutuhkan *input*an yang selalu ada, sehingga tidak mempengaruhi *output*. Jadi jika *input* dari model ini hilang, tidak akan memepengaruhi variabel karena tidak memiliki target. Tipe ini berlawanan dengan *supervised learning* yang konsepnya adalah *input* diasumsikan diawal sedangkan *output* diakhir. Antara *input* dan *output*

saling berhubungan sehingga kita pasti dapat mepredisikan *output* yang akan keluar (memiliki target). SOM ini ditemukan oleh Teuvo Kohonen (http://www.cis.hut._/harri/thesis/ valpola_thesis/node34.html).

Jaringan SOM Kohonen terdiri dari dua lapisan (*layer*), yaitu lapisan *input* dan lapisan *output*. Setiap neuron dalam lapisan *input* terhubung dengan setiap neuron pada lapisan *output*. Setiap neuron dalam lapisan *output* merepresentasikan kelas dari *input* yang diberikan (Madarum, 2003).



Gambar 2.8. Kohonen Model (Haykin, 2005)

Setiap neuron *output* mempunyai bobot untuk masing-masing neuron *input*. Proses pembelajaran dilakukan dengan melakukan penyesuaian terhadap setiap bobot pada neuron *output*. Setiap *input* yang diberikan dihitung jarak *Euclidian*-nya dengan setiap neuron *output*, kemudian cari neuron *output* yang mempunyai jarak minimum. Neuron yang mempunyai jarak yang paling kecil disebut neuron pemenang atau neuron yang paling sesuai dengan *input* yang diberikan.

Algoritma pengelompokan pola jaringan SOM adalah sebagai berikut (Siang, 2005) :

0. Inisialisasi
    a. Bobot w_{ji} (acak)
    b. Laju pemahaman awal dan faktor penurunannya
1. Selama kondisi penghentian bernilai salah, lakukan langkah 2-7
2. Untuk setiap vektor masukan x, lakukan langkah 3-5

3. Hitung $d = \sqrt{\sum_{i}^{n} (Wi - Xi)^2}$ untuk semua j
4. Tenntukan indeks J sedemikian hingga D(J) minimum
5. Untuk setiap unit j disekitar J modifikasi bobot :
   $w_{ji}^{baru} = w_{ji}^{lama} + \alpha(x_i - w_{ji}^{lama})$
6. Modifikasi laju pemahaman
7. Uji kondisi penghentian

Apabila semua w_{ij} hanya berubah sedikit saja, berarti iterasi sudah mencapai konvergensi sehingga dapat dihentikan.

2.4 *Unified Software Development Process*

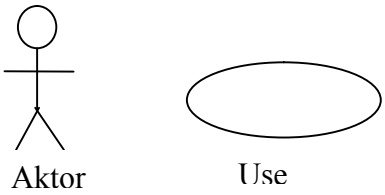
Unified Software Development Process (USDP) merupakan salah satu metode pemodelan objek dalam pengembangan sistem/ perangkat lunak berbasis OOP yang menggunakan UML *(Unified Modeling Language)* sebagai *tool* utamanya. UML adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek *(Nugroho, 2010).*

2.4.1 *Use Case Diagram*

Use cases dan aktor merupakan model fungsionalitas sistem/ perangkat lunak yang dilihat dari sisi pengguna yang ada diluar sistem, namun memiliki koherensi terhadap sistem/ perangkat lunak tersebut yang diekspresikan sebagai transaksi-transaksi yang terjadi antara aktor dan sistem *(Nugroho, 2010).* Dalam buku lain yang berjudul *Object-Oriented Software Engineering (A Use Case Driven Approach)* dijelaskan bahwa aktor dan *use cases* merupakan transformasi utama yang dibuat dari sisi spesifikasi kebutuhan sebuah model sistem/ perangkat lunak yang terdiri dari *(Jacobson, 1992)*:

1. Sebuah model *use case*
2. Deskripsi terhadap *interface*

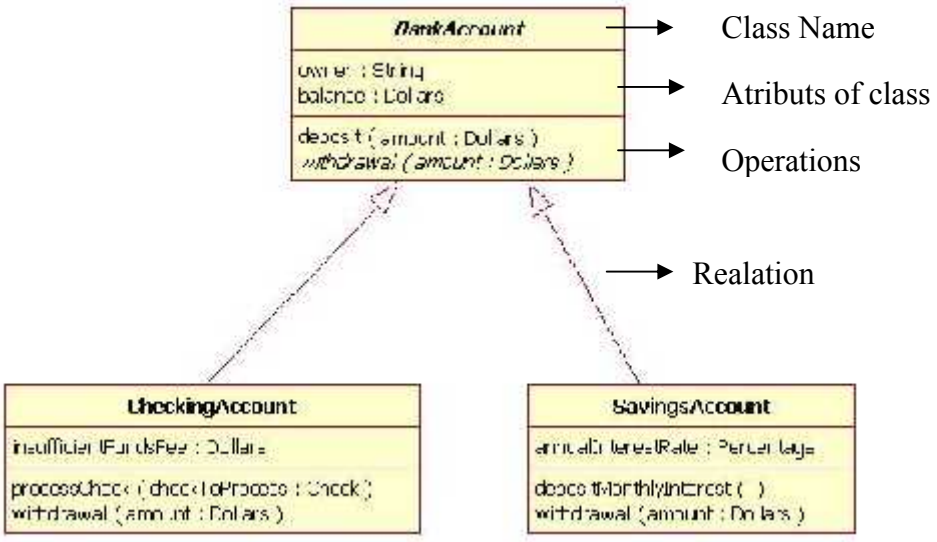
3. Domain masalah model



Gambar 2.9. *Use case model* (Jacobson, 1992)

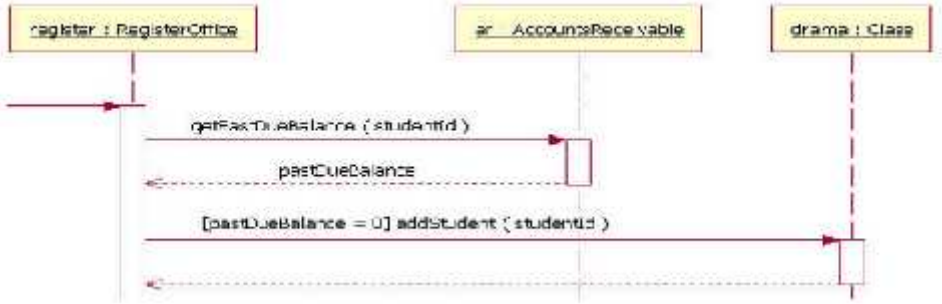
2.4.2 *Class Diagram*

Sebuah kelas *(class)* merupakan konsep diskret dalam model yang terdiri dari identitas *(identity), state,* prilaku *(behaviour),* serta relasi dengan kelas yang lainnya *(relationship).* Beberapa jenis pengelasan mencakup di dalamnya kelas, *interface,* serta tipe data. Sementara jenis yang lainnya mencakup konsep-konsep prilaku, sesuatu dalam lingkungan saat aplikasi dijalankan *(runtime environment),* atau struktur-struktur yang berkait dengan tahapan implementasi atau dalam kalimat lain sebuah kelas terdiri dari *use case,* aktor, komponen, simpul *(node)*, serta subsistem *(Nugroho, 2010).*



Gambar 2.10. *Class diagram* (www.ibm.com)

2.4.3 *Sequence Diagram*



Gambar 2.11. *Sequence diagram* (www.ibm.com)

Secara umum *sequence diagram* merupakan gambaran interaksi sebagai diagram dua dimensi (matra) yang terdiri dari matra vertikal dan horizontal. Matra vertikal merupakan sumbuh waktu yang berubah dan bertambah dari atas ke bawah, sementara matra horizontal merupakan peran pengklasifikasi yang merepresentasikan objek-objek mandiri yang terlibat dalam sebuah kolaborasi. Masing-masing pengklasifikasi direpresentasikan sebagai kolom-kolom vertikal dalam *sequence diagram* dan sering disebut sebagai garis waktu *(lifeline)*.

BAB III
METODOLOGI PENELITIAN

Metodologi penelitian dilakan dengan menggunakan metode kuantitatif untuk mengukur akurasi kombinasi MFCC dan SOM dalam identifikasi pembicara.

3.1 Data Penelitian

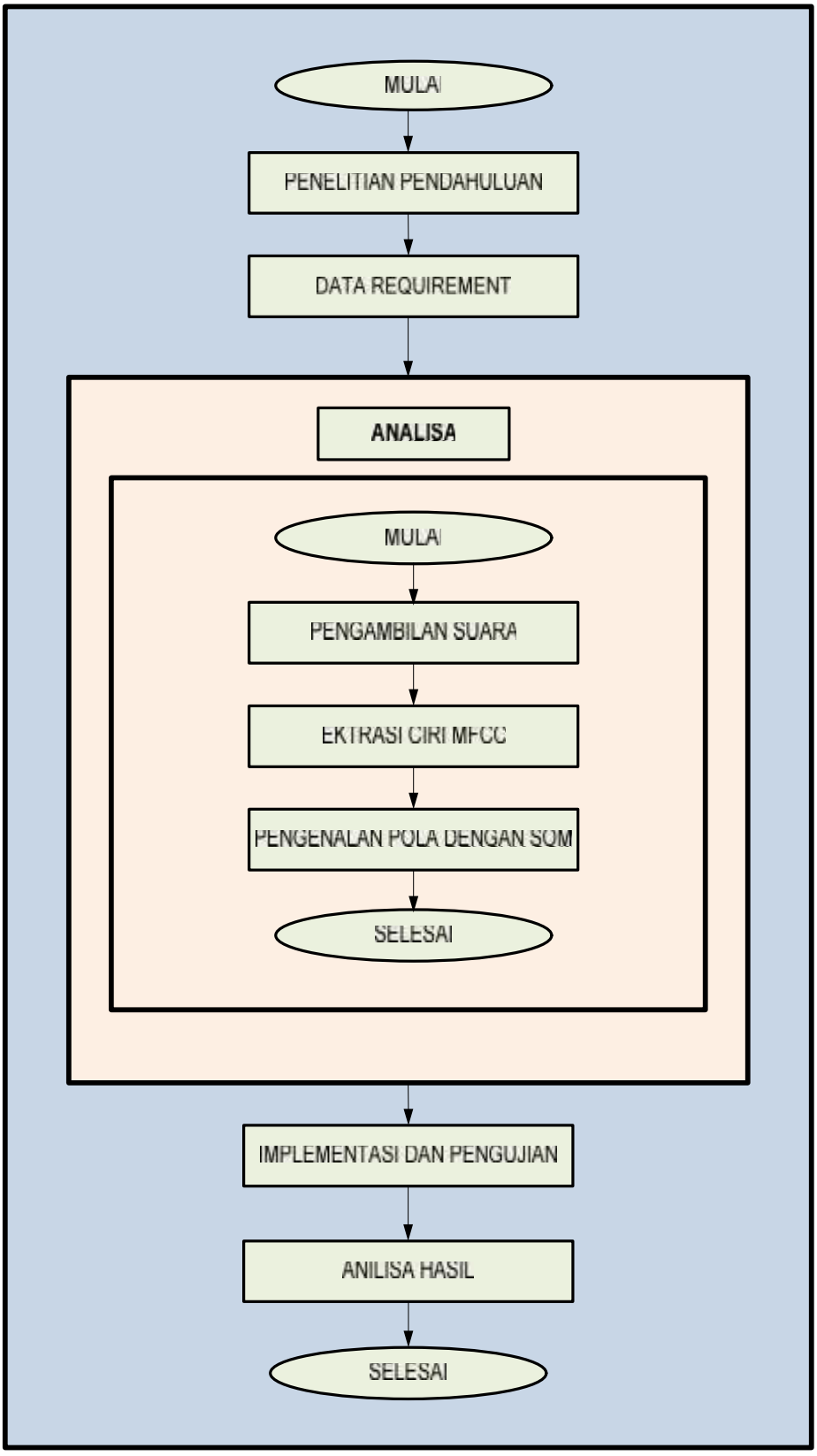
Bahan suara yang digunakan berasal dari 10 orang yang terdiri dari 5 orang pria dan 5 orang wanita. Masing-masing mereka mengucapkan sebuah kalimat selama 20 detik sebanyak dua kali. *Sample* suara pertama akan dijadikan data pelatihan dan *sample* kedua akan dijadikan data pengujian.

3.2 Tahapan Penelitian

Metodologi penelitian digunakan sebagai pedoman dalam pelaksanaan penelitian agar hasil yang dicapai tidak menyimpang dari tujuan yang telah dilakukan sebelumnya. Metodologi penelitian yang digunakan dalam penyusunan Tugas Akhir ini akan melalui beberapa tahapan yang membentuk sebuah alur yang sistematis.

3.2.1 Penelitian Pendahuluan

Pada tahapan ini, peneliti mengidentifikasi penelitian yang akan dilakukan dengan melakukan studi pustaka. Hal ini untuk mengetahui metode apa yang akan digunakan untuk menyelesaikan permasalahan yang akan diteliti, serta mendapatkan dasar-dasar referensi yang kuat dalam menerapkan suatu metode yang akan digunakan dalam Tugas Akhir ini. Studi kepustakaan juga melihat serta membandingkan penelitian-penelitian yang sudah ada, sehingga peneliti mendapatkan tema penelitian mengenai pengenalan pembicara dengan metode MFCC sebagai ekstrasi ciri dan SOM sebagai pencocokan ciri.



Gambar 3.1. *Flowchart* tahapan penelitian

3.2.2 Data *Requirements*

Untuk mendapatkan data-data yang dibutuhkan, maka dilakukan studi pustaka untuk mendapatkan dasar-dasar referensi yang kuat dalam menerapkan suatu metode yang akan digunakan dalam Tugas Akhir ini, yaitu dengan mempelajari *ebook* dari Mikael Lundsgaard berjudul *JOPSPEECH* serta buku-buku, artikel-artikel dan jurnal-jurnal yang berhubungan dengan pengenalan pembicara, MFCC dan SOM.

3.2.3 Analisa

Setelah melakukan penelitian pendahuluan, identifikasi masalah dan data *requirement* maka beberapa analisa untuk penelitian ini. Berikut rincian dari analisa-analisa tersebut:

1. Analisa Model Permasalahan

   Permasalahan yang akan diselesaikan adalah bagaimana akurasi identifikasi pembicara dengan menggunakan metode MFCC pada ekstrasi ciri dan SOM untuk pencocokan ciri.

2. Kebutuhan Data

   Penelitian ini membutuhkan *sample* untuk pelatihan dan pengujian. Pada penelitian ini s*ample* didapat dari 10 orang dengan setiap orang mengucapkan kalimat selama 20 detik sebanyak 2 kali. *Sample* pertama akan dijadikan data pelatihan dan *sample* kedua akan dijadikan data pengujian.

3. Analisa Penyelesaian Masalah

   Menganalisa data masukan, proses dan hasil dari penelitian ini.

4. Analisa MFCC

   Menganalisa metode MFCC untuk ekstrasi ciri dari sample suara yang ada.

5. Analisa SOM

   Menganalisa metode SOM untuk permasalahan pengelompokkan pembicara dengan data masukan berasal dari hasil ekstrasi ciri MFCC.

3.2.4 Implementasi dan Pengujian

Pada tahap ini akan diimplementasikan identifikasi pembicara dengan menggunakan bahasa pemrograman Java. Mekanisme pembelajaran yang digunakan dalam penelitian ini adalah jaringan syaraf tiruan SOM. Kemudian akan dilakukan pengujian terhadap implementasi tersebut dan peninjauan kembali hasil dari akurasi yang telah dikembangkan.

Lingkungan kerja:

a. Perangkat lunak dan sistem operasi yang digunakan dalam penelitian adalah Java dan *Microsoft Window*s *Vista.*
b. Perangkat keras yang digunakan dalam penelitian ini adalah komputer dengan spesifikasi:
    1. Prosesor Intel Core 2 Duo 2.0 GHz
    2. Memori 1 GB

Tujuan dilakukan pengujian terhadap pelatihan yang telah dilakukan adalah agar bisa menentukan akurasi yang didapat dari kombinasi metode ekstrasi ciri MFCC dan pengenalan pola SOM dalam pengidentifikasian pembicara pada data *close-set*.

3.2.5 Analisa Hasil

Menganalisa hasil pengujian yang telah dilakukan untuk mengetahui apakah kombinasi metode MFCC dan SOM dapat mengidentifikasi pembicara dengan baik atau tidak. Tahap ini juga menentukan berapa persen ketepatan identifikasi pembicara yang didapat dari hasil pengujian. Pada saat pengujian akan diuji pengaruh beberapa parameter terhadap ketepatan identifikasi pembicara. Parameter-parameter tersebut adalah jumlah koefesien MFCC, jumlah data latih, jumlah iterasi serta kombinasi antara koefesien MFCC dan jumlah iterasi.

BAB IV
ANALISA DAN PERANCANGAN

Analisa dan perancangan yang akan dilakukan dalam penelitian ini adalah analisa untuk membuat aplikasi pengenalan pembicara yang bisa menguji berapa akurasi ketepatan kombinasi antara metode MFCC dan SOM dalam mendeteksi pembicara. Bagian analisa akan berfokus pada parameter-parameter apa saja yang akan menjadi *input*, proses apa saja yang akan dilakukan serta bagaimana *output* yang dihasilkan. Sedangkan perancangan menggunakan UML-USDP sebagai pemodelannya.

4.1 Analisa Penyelesaian Masalah

Sub ini akan membahas tentang *input*, proses, *output* dan *flowchart* yang yang dimiliki aplikasi.

4.1.1 Analisa Data Masukan (*Input*)

Aplikasi ini memiliki beberapa parameter masukan yang akan dianalisa dalam pendeteksian pembicara. Berikut parameter-parameternya:

1. *Frame size*

   Byte yang terdapat pada sample suara dibagi dalam beberapa *frame* dengan ukuran adalah 2^n. Setiap *frame* akan dihitung nilai MFCC nya. Jumlah *frame size* akan dimasukan oleh *user*.
2. Jumlah *coefficient* MFCC

   Jumlah *coefficient* MFCC adalah jumlah nilai yang diambil dari setiap *frame* yang dimasukkan. Nilai-nilai tersebut adalah karakteristik suara yang didapat. Jumlah *coefficient* MFCC akan dimasukan oleh *user*.
3. Jumlah data pelatihan yang diambil per pembicara

   Jumlah ini adalah jumlah hasil ektrasi ciri yang diambil untuk data pelatihan. Jumlah data pelatihan akan dimasukan oleh *user*.
4. Inisialisasi vektor bobot

Inisialisasi vektor bobot adalah untuk menetukan nilai awal vektor bobot. Inisialisasi disini dilakukan secara *random*.

5. Jumlah iterasi

   Jumlah iterasi disini adalah jumlah maksimum pengulangan algortima SOM. Jumlah iterasi akan dimasukan oleh *user*.

4.1.2 Analisa Proses

Langkah-langkah yang terjadi dalam proses pengenalan pola suara masing-masing pembicara secara garis besarnya adalah sebagai berikut:

Langkah I *User* menginputkan parameter-parameter masukan yang dibutuhkan.

Langkah II Aplikasi melakukan ekstrasi ciri terhadap suara-suara pembicara berdasarkan parameter-parameter yang dimasukan dengan menggunakan metode MFCC.

Langkah III Setelah dilakukan ektrasi ciri lalu dilakukan pengenalan pola pada masing-masing suara pembicara dengan menggunakan metode SOM.

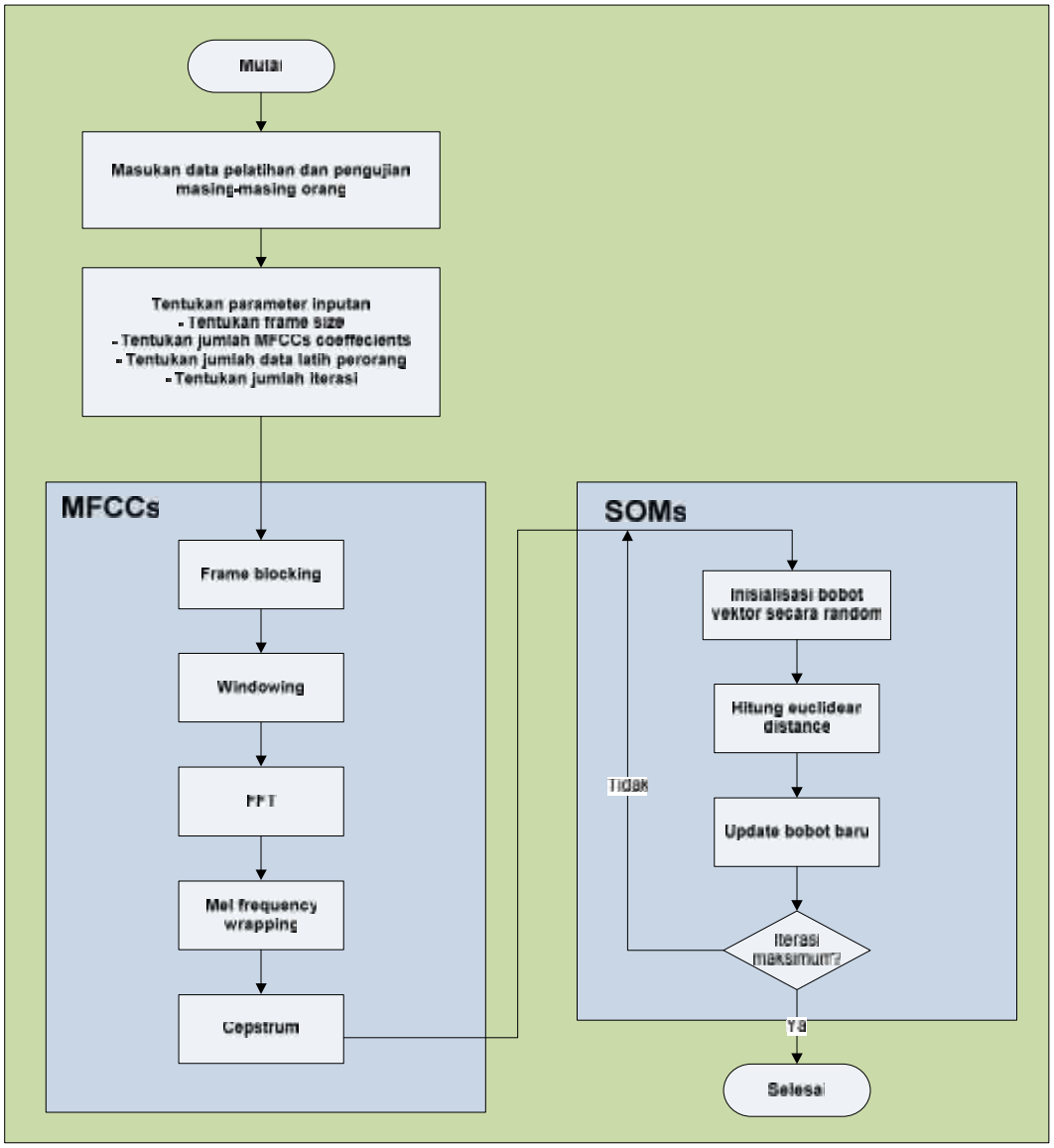
Langkah IV Hasil pengenalan pola tersebut diuji dengan suara pengujian dan hasil pengujian ditampilkan dalam tabel yang berisi persen kemiripan.

4.1.3 Analisa Data Keluaran (*Output*)

Tujuan akhir dari aplikasi ini adalah menguji berapa persen ketepatan kombinasi metode MFCC dan SOM dalam pengenalah pembicara. Hasil pengujian ditampilkan dalam bentuk tabel yang berisi persen kemiripan data pengujian yang dihitung berdasarkan pola yang telah didapat dalam pelatihan sebelumnya.

4.1.4 *Flowchart*

Langkah-langkah penyelesaian aplikasi pengenalan pembicara dengan metode MFCC dan SOM ini dapat digambarkan melalui *flowchart* pada gambar 4.1.



Gambar 4.1. *Flowchart* pengenalan pembicara

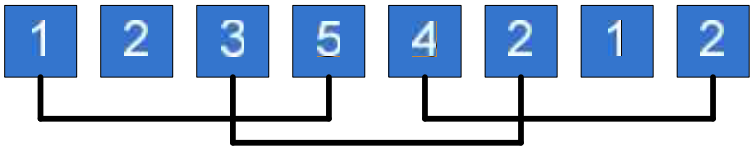
Dari gambar *flowchart* diatas dapat dijelaskan bahwa langkah pertama adalah memasukan data pelatihan dan pengujian masing-masing orang. Kemudian tentukan parameter-parameter yang dibutuhkan, yaitu jumlah *frame size*, MFCC *coefficient*, data latih dan iterasi. Setelah itu lalukan ekstrasi ciri dengan MFCC, yang terdiri dari *frame blocking*, *windowing*, FFT, *mel frequency wrapping* dan *cepstrum*. Setelah ekstrasi ciri dilakukan maka dilakukan pengenalan pola dengan SOM yang secara garis besar terdiri dari peng-inisialisasian bobot, menghitung jarak *euclidean* dan *update* bobot.

4.1.4.1 Contoh Persoalan Menggunakan MFCC

Pada contoh ini dimisalkan *sample* yang masuk adalah x = {1, 2, 3, 5, 4, 2, 1, 2}. *Frame size* yang digunakan adalah 4. Frekuensi minimal = 0 dan frekuansi maksimal = 8. Jumlah MFCC *coefficient*-nya 2. Filter yang digunakan adalah 2.

Langkah 1 – *Framing*

Hal pertama yang kita lakukan adalah membagi *array* yang masuk tersebut dalam beberapa *frame*. Cara pembagian diilustrasikan seperti Gambar 4.2.



Gambar 4.2. Ilustrasi *framing*.

Meskipun menggunakan *frame size* 4 pada 8 buah data tidak dihasilkan dua buah *frame* tetapi tiga buah. Ini karena dalam pembagian *frame*-nya kita menggunakan *hop size* sebagai *overlapping*-nya yaitu *frame size* dibagi 2.

Frame - *frame* yang dihasilkan adalah sebagai berikut:

Frame 1 : {1, 2, 3, 5}

Frame 2 : {3, 5, 4, 2}

Frame 3 : {4, 2, 1, 2}

Langkah 2 – *Windowing*

Setelah itu kita lakukan *windowing* untuk menghaluskan data pada setiap *frame*-nya. Pertama hitunglah nila W nya dengan persamaan (2.2) sehingga menghasilkan data berikut:

W(0) = `0.7699999999999999`

W(1) = `0.7700000000000002`

W(2) = `0.08000000000000002`

W(3) = `0.7699999999999997`

Lalu kalikan nilai W(i..n) dengan nilai item pada masing-masing *frame* sehingga dihasilkan sebagai berikut :

Frame 1 : {1, 2, 3, 5} → {0.76, 1.54, 0.24, 3.84}

Frame 2 : {3, 5, 4, 2} → {2.30, 3.85, 0.32, 1.53}

Frame 3 : {4, 2, 1, 2} → {3.07, 1.54, 0.08, 1.53}

Langkah 3 – FFT

Setelah itu kita lakukan *windowing* untuk menghaluskan data pada setiap *frame*-nya dengan persamaan (2.3) sehingga menghasilkan data berikut:

Frame 1 : {0.76, 1.54, 0.24, 3.84} → {40.19, 9.43, 0.43, 9.43}

Frame 2 : {2.30, 3.85, 0.32, 1.53} → {55.05, 14.08, 0.17, 14.08}

Frame 3 : {3.07, 1.54, 0.08, 1.53} → {10.49, 0.36, 0.57, 0.36}

Langkah 4 – *Mel Frequency Wrapping*

Langkah pertama untuk mendapatkan *mel frequency wrapping* adalah dengan mendapat *mel filter bank boundary*. Langkah mendapat *mel filter bank boundary* dijelaskan pada tahapan berikut.

1. Jadikan frekuensi minimal dan maksimal menjadi *mel frequency* dengan persamaan 2.2.

   Min = 0 → 0.0

   Max = 8 → 12.80

2. Hitung delta

   Delta = (max-min)/(filter+1)

   Delta = 4.26896328931801

3. Hitung *mel frequency* selanjutnya dengan menjadikan delta sebagai *increment*-nya sampai batas akhir *filter*+2.

   M(0) = 0.0

   M(1) = 4.26896328931801

   M(2) = 8.53792657863602

   M(3) = 12.80688986795403

4. Ubah dari *mel frequency* ke frekuensi normal dengan persamaan 2.3.

   b[0] = 0.0

   b[1] = 2.6565719490367012

   b[2] = 5.3232258616739525

   b[3] = 7.9999999999999405

Langkah selanjutnya adalah mengisi *row* masing filter. Panjang *row* adalah *frame size/2* + 1. Lalu hitung dengan (j – b(i-1)) * (2(b(i+1)-b(i-1)))/(b(i) – b(i-1))). Sehingga didapat hasil seperti dibawah ini:

Mfb(0) = {0.0, 0.14142738770674632, 0.28285477541349263}

Mfb(1) = {0.0, 0.0, 0.0}

Langkah terakhir adalah mengalikan *mel spectrum*-nya dengan cara mengalikan *mel filter blank* dengan frekuensi hasil langkah 3 sehingga menjadi seperti dibawah ini:

S(0) = {1.4572112319752313 0.0}

S(1) = {2.041702339889672 0.0}

S(2) = {0.21429077785326212 0.0}

Langkah 5 – *Cepstrum*

Pada langkah terkahir ini kita akan mendapat *mel frequency cepstrum coefficients*-nya. Langkah ini mengacu pada persamaan 2.6. Hal pertama yang kita lakukan adalah melakukan *log* pada semua *spectrum* yang dihasilkan dilangkah 4. Sehingga menghasilkan data seperti dibawah ini.

S(0) = {0.6612328416197294 0.0}

S(1) = {3.099924265039553 0.0}

S(2) = {-6.689965187090812 0.0}

Hasil tersebut dijadikan *discrit cosines transform* hasilnya seperti di bawah ini:

C(0) = {0.46756222625256105 0.0}

C(1) = {2.1919774689741924 0.0}

C(2) = {0.0 0.0}

Hasil diatas adalah hasil dari ekstrasi ciri MFCC. Hasil ini yang akan dikenali polanya dengan SOM.

4.1.4.2 Contoh Persoalan Menggunakan SOM

Misalkan kita ingin mengelompokkan data-data berikut menjadi 2 kelompok:

X1	X2
0,10	0,10
0,20	0,20
0,30	0,10

Tingkat pembelajaran (α = 0,6) dengan setiap kenaikan iterasi akan diset 0,5x(α). Maksimum iterasi ditetapkan sebesar 100.

Penyelesaian :

Langkah 1 – Inisialisasi

α = 0,60

Pengurangan α = 0,5

MaxIterasi = 100

Random bobot awal , W_{ij} :

0,50 0,50

0,50 0,50

Langkah 2 – Hitung similaritas dengan dengan menggunakan jarak Euclidian dan *update* bobot neuron pemenang :

Iterasi ke - 1

Data ke - 1

Rumus : $d = \sqrt{\sum_{i}^{n} (Wi - Xi)^2}$

Bobot ke-1 = $\sqrt{(0,5 - 0,1)2 + (0,5 - 0,1)2} = \sqrt{0.32}$

Bobot ke-2 = $\sqrt{(0,5 - 0,1)2 + (0,5 - 0,1)2} = \sqrt{0.32}$

Jarak terkecil pada bobot ke-1

Bobot ke-1 baru :

Rumus : $W_{ij}(t+1) = W_{ij}(t) + \alpha(t) * [X_i(t) - W_{ij}(t)]$

W_{11} = 0,5 + 0,6 (0,1 – 0,5) = 0,26

W_{12} = 0,5 + 0,6 (0,1 – 0,5) = 0,26

Data ke – 2

Rumus : $d = \sqrt{\sum_{i}^{n} (Wi - Xi)^2}$

Bobot ke-1 = $\sqrt{(0,26 - 0,2)2 + (0,26 - 0,2)2} = \sqrt{0,0072}$

Bobot ke-2 = $\sqrt{(0,5 - 0,2)2 + (0,5 - 0,2)2} = \sqrt{0,1800}$

Jarak terkecil pada bobot ke-1

Bobot ke-1 baru :

Rumus : $W_{ij}(t+1) = W_{ij}(t) + \alpha(t) * [X_i(t) - W_{ij}(t)]$

W_{11} = 0,26 + 0,6 (0,2 – 0,26) = 0,224

W_{12} = 0,26+ 0,6 (0,2 – 0,26) = 0,224

Data ke - 3

Rumus : $d = \sqrt{\sum_{i}^{n} (Wi - Xi)^2}$

Bobot ke-1 = $\sqrt{(0,224 - 0,3)2 + (0,224 - 0,1)2} = \sqrt{0,0212}$

Bobot ke-2 = $\sqrt{(0,5 - 0,3)2 + (0,5 - 0,1)2} = \sqrt{0,2000}$

Jarak terkecil pada bobot ke-1

Bobot ke-1 baru :

Rumus : $W_{ij}(t+1) = W_{ij}(t) + \alpha(t)h(t) * [X_i(t) - W_{ij}(t)]$

W_{11} = 0,224+ 0,6 (0,3 – 0,224) = 0,2696

W_{12} = 0,224+ 0,6 (0,1 – 0,224) = 0,1496

Langkah 3 – *Update* tingkat pembelajaran dan fungsi tetangga.

Tingkat pembelajaran : α = 0,5 * 0,6 = 0,3

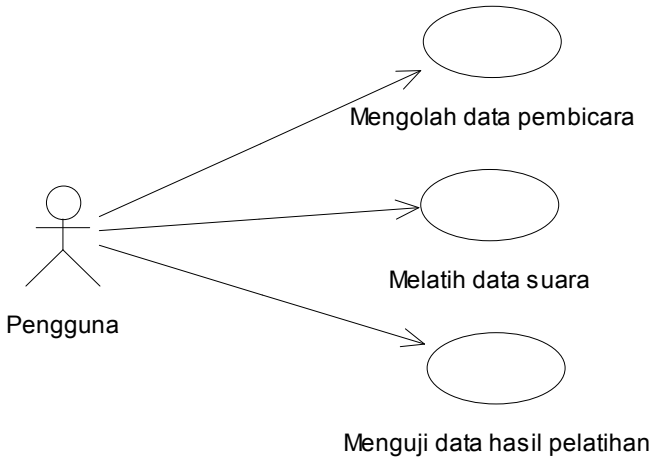
Langkah 4 – Lakukan langkah 2 sampai 4 sampai iterasi maksimum.

4.2 Perancangan Aplikasi

Setelah analisis sistem selesai dilakukan, maka akan dilakukan perancangan terhadap model aplikasi yang akan dibuat. Model perancangan yang akan digunakan dalam aplikasi ini adalah UML meliputi *use case diagram*, *class diagram*, *sequence diagram*, *collaboration diagram*, *activity diagram*, *statechart diagram* dan *deployment diagram*.

4.2.1. *Use Case Diagram*

Use Case Diagram menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada di luar sistem (*actor*). Diagram ini menunjukkan fungsionalitas sistem dan bagaimana cara sistem berinterakasi dengan perangkat luar dari sistem atau aplikasi.



Gambar 4.3. *Use case diagram* pengenalan pembicara

Tabel 4.1. Spesifikasi *use case diagram* mengolah data pembicara

Aktor utama	Pengguna
Kondisi awal	Aplikasi telah dijalankan sebelumnya
Kondisi Akhir	Data pembicara berhasil diolah
Main success scenario	1. Pengguna mengklik menu pembicara. 2. Aplikasi menampilkan *form* untuk mengatur pembicara yang berisi ojek *combo* box nama pembicara dan jenis data suara (pelatihan atau pengujian) suara, *button* rekam, hapus dan tambah pembicara. 3. Jika pengguna mengklik tombol tambah pembicara maka aplikasi akan menampilkan *dialog input* nama. Pengguna memasukkan nama dan mengklik tombol *yes*, maka pembicara baru akan disimpan 4. Jika pengguna mengklik tombol rekam maka suara pembicara akan direkam berdasarkan jenis yang dipilih di *combo* jenis. 5. Jika pengguna mengklik tombol hapus, maka pembicara akan dihapus.

Tabel 4.2. Spesifikasi *use case diagram* melatih data suara

Aktor utama	Pengguna
Kondisi awal	Data suara telah tersedia sebelumnya
Kondisi Akhir	Data suara berhasil dikenali polanya.
Main success scenario	1. Pengguna mengklik menu pelatihan. 2. Aplikasi menampilkan *form* yang berisi inputan-inputan yang dibutuhkan (*frame size*, jumlah MFCC *coefficient*, jumlah data pelatihan dan jumlah iterasi) serta *button* latih dan batal. 3. Jika pengguna mengklik tombol latih maka data suara akan dilatih dengan cara mengekstrasi cirinya dengan MFCC lalu mengenali polanya dengan menggunakan SOM. 4. Jika pengguna mengklik tombol batal, maka aplikasi akan membatalkan pelatihan.

Tabel 4.3. Spesifikasi *use case diagram* menguji data hasil pelatihan

Aktor utama	Pengguna
Kondisi awal	Data suara telah dilatih sebelumnya
Kondisi Akhir	Hasil pengujian ditampilkan dalam *list*
Main success scenario	1. Pengguna mengklik menu pengujian. 2. Data hasil pelatihan tadi diuji dengan data suara pengujian. Pengujian dilakukan dengan menghitung jarak dengan rumus *Euclidean*. 3. Aplikasi menampilkan form hasil pelatihan yang berisi persen kemiripan suara.

4.2.2. *Class Diagram*

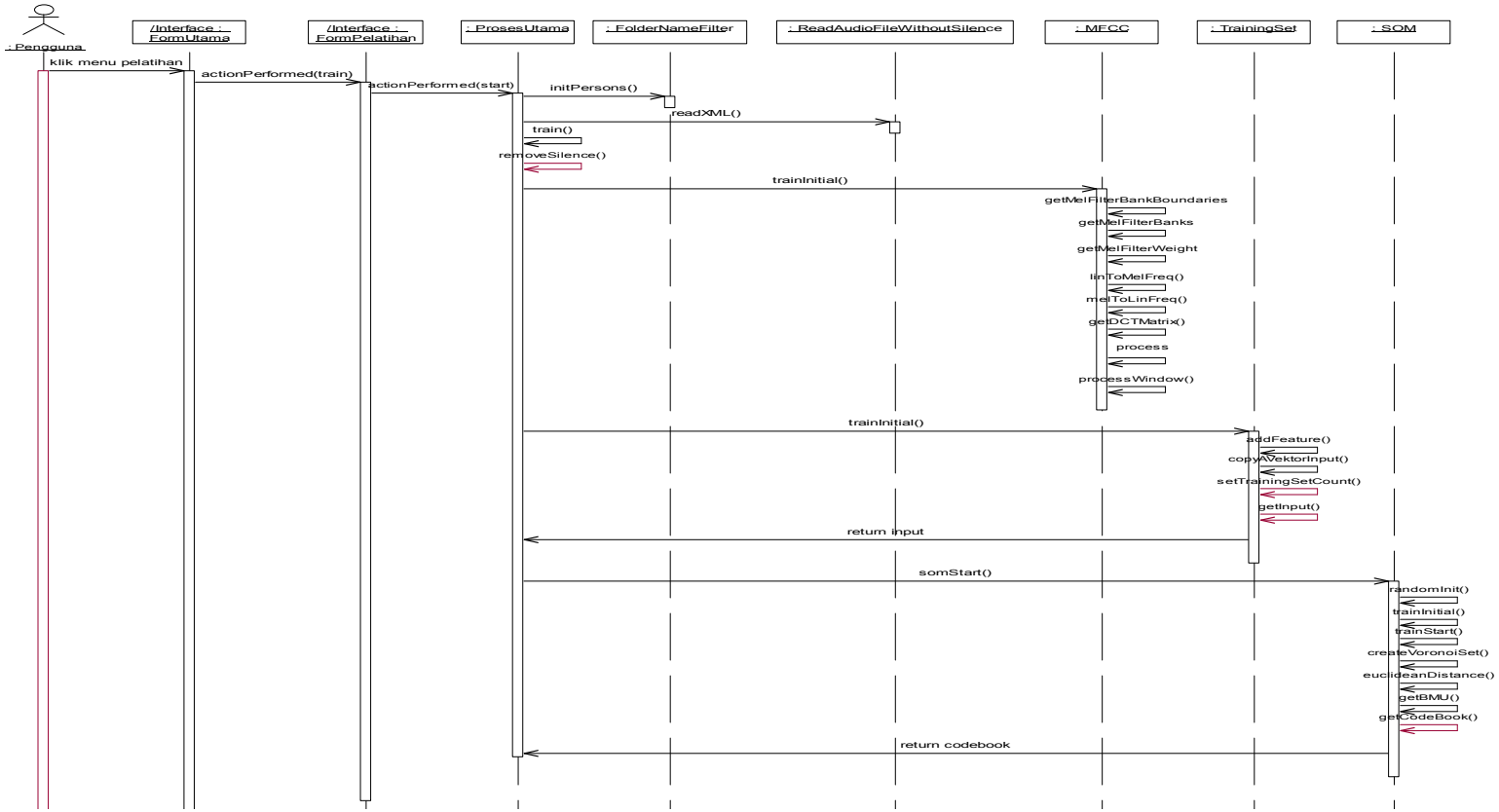
Gambar 4.4. *Class diagram* pengenalan pembicara

Tabel 4.4. Spesifikasi *class diagram diagram*

No.	Nama Kelas atau Objek	Deskripsi
1.	*AudioRecord*	*Class* ini berfungsi untuk merekam suara.
2.	*ReadAudioFileWithoutSilence*	*Class* ini berfungsi membaca *file wav*.
3.	MFCC	*Class* ini berfungsi melalukan algoritma MFCC.
4.	FormPelatihan	*Class* yang menentukan parameter untuk pelatihan.
5.	FormPembicara	*Class* ini berfungsi mengolah data pembicara.
6.	FormUtama	*Class* ini adalah form utama dari aplikasi pengenalan pembicara.
7.	Mulai	*Class* ini untuk memulai menjalankan aplikasi pengenalan pembicara.
8.	*MetricModel*	*Class* ini *interface* dari *class MetricModel*.
9.	*EuclidesMetric*	*Class* ini adalah untuk menghitung jarak *Euclidean*.
10.	SOM	*Class* ini untuk melatih data pelatihan dengan menggunakan SOM berdasarkan parameter yang dimasukkan.
11.	*TrainingSet*	*Class* ini untuk mengumpulkan data-data pelatihan dalam bentuk matrik.
12.	*FolderNameFilter*	*Class* ini untuk memfilter folder yang berisi file.
13.	*Person*	*Class* ini untuk menyimpan informasi setiap pembicara.
14.	ProsesUtama	*Class* ini berisi proses utama yang ada di aplikasi pengenalan pembicara.

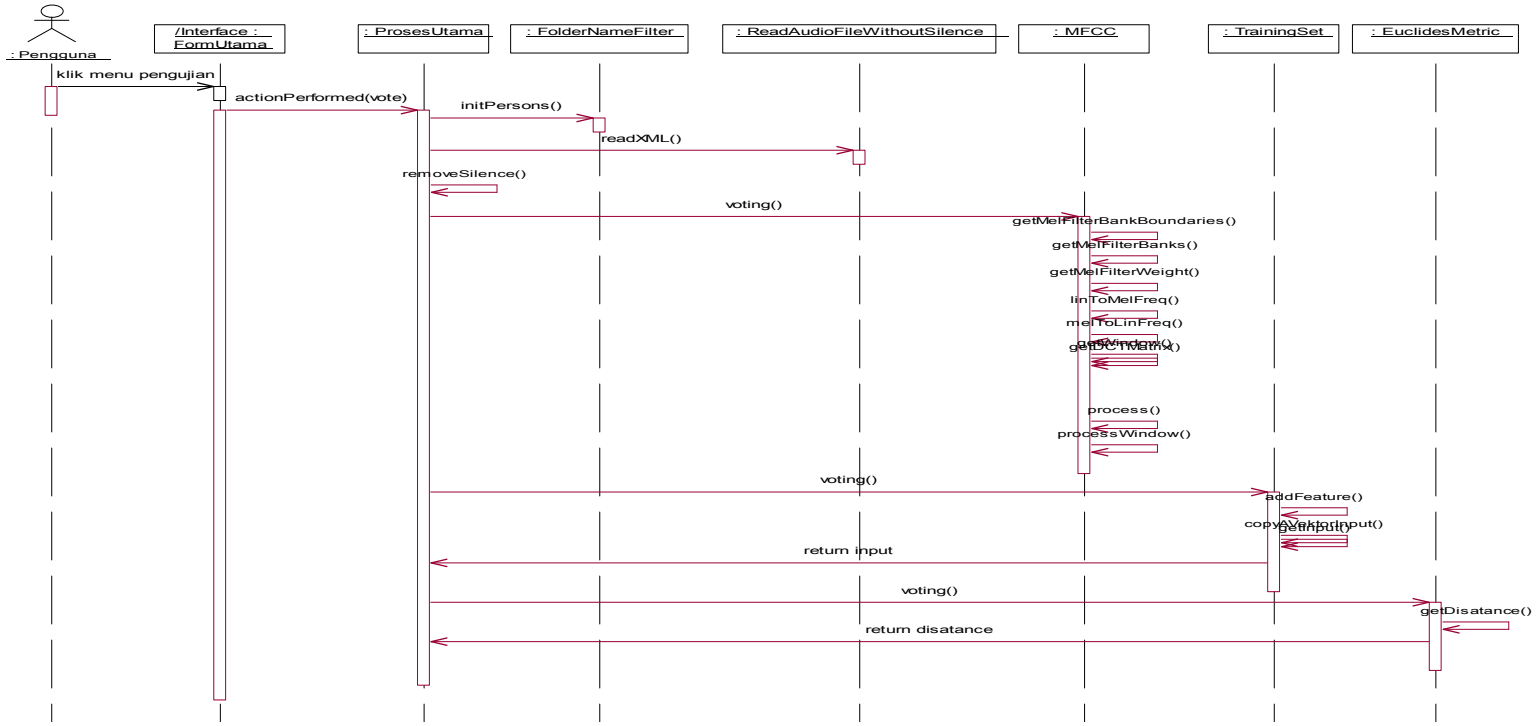
4.2.3. *Sequence Diagram*

4.2.3.1. *Sequence Diagram* Melatih Data Suara



Gambar 4.5. *Sequence diagram* melatih data suara

4.2.3.2. *Sequence Diagram* Menguji Data Hasil Pelatihan

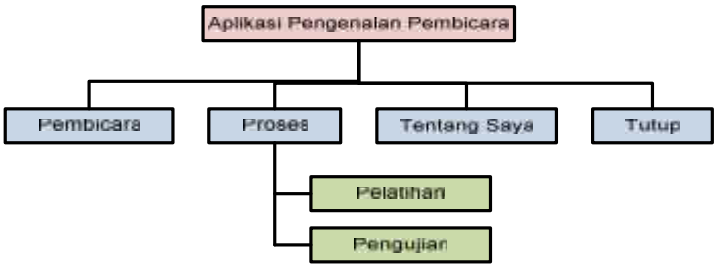


Gambar 4.6. *Sequence diagram* menguji data hasil pelatihan

Perancangan *sequence diagram* yang lain terlampir pada lampiran B

4.3 Perancangan Struktur Menu

Berikut menu-menu yang disediakan aplikasi pengenalan pembicara ini.



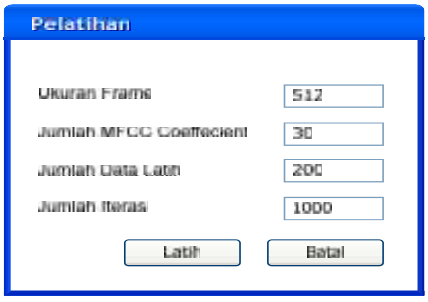
Gambar 4.7. Rancangan menu

Gambar 4.7. diatas teridiri dari empat buah menu yaitu menu **pembicara**, menu **proses**, menu **tentang saya**, menu **tutup**. Menu **pembicara** adalah menu untuk mengolah data pembicara. Menu **proses** terdiri dari dua buah submenu yaitu submenu **pelatihan** dan submenu **pengujian**. Submenu **pelatihan** pada menu **proses** adalah untuk melatih data-data suara sedangkan submenu pengujian untuk menguji data hasil pelatihan. Menu **tentang saya** adalah untuk menampilkan informasi pembuat aplikasi. Menu terakhir yaitu menu **tutup** adalah untuk mengakhiri aplikasi.

4.4 Perancangan Antar Muka

Bagian ini akan menerangkan rancangan antar muka untuk form utama, *form* pelatihan dan form hasil pengujian. Berikut rancangan antar muka ketiga *form* tersebut.

4.4.1 Perancangan Antar Muka *Form* Pelatihan



Gambar 4.8. Rancangan antar muka *form* pelatihan

Tabel 4.5. Spesifikasi antar muka *form* pelatihan

Nama objek	Jenis	Keterangan
Text Ukuran *Frame*	*Textbox*	Berfungsi untuk memasukan ukuran *frame*
Text MFCC	*Textbox*	Berfungsi untuk memasukan memasukan jumlah MFCC *coeffecient*.
Text Data Latih	*Textbox*	Berfungsi untuk memasukan jumlah data latih per pembicara
Text Iterasi	*Textbox*	Berfungsi untuk memasukan jumlah iterasi
Button Latih	*Button*	Berfungsi untuk memulai pelatihan
Button Batal	*Button*	Berfungsi untuk membatalkan pelatihan

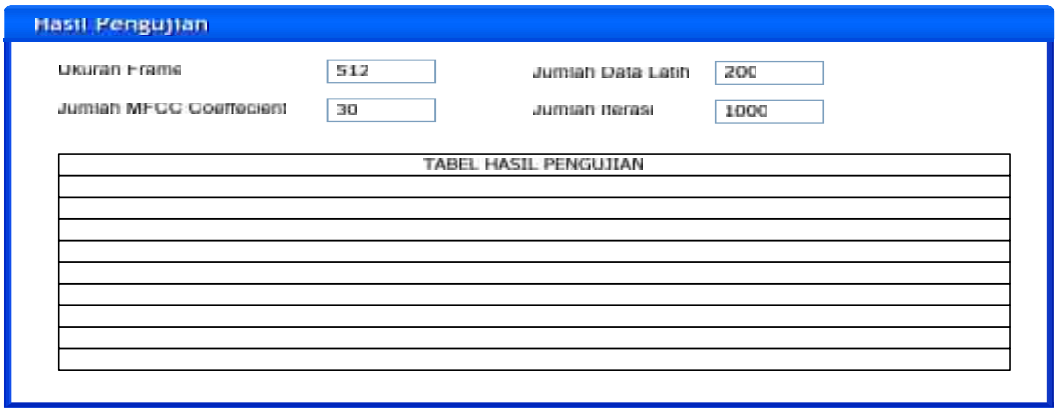
4.4.2 Perancangan Antar Muka *Form* Utama

Gambar 4.9. Rancangan antar muka *form* utama

Tabel 4.6. Spesifikasi antar muka *form* utama

Nama objek	Jenis	Keterangan
Menu Pembicara	*Menu*	Berfungsi untuk mengelolah data pembicara.
Menu Pelatihan	*MenuItem*	Berfungsi untuk memasukkan paramater-parameter inputan serta memulai pelatihan.
Menu Pengujian	*MenuItem*	Berfungsi menampilkan *list* hasil pengujian.
Menu Tentang Saya	*Menu*	Memberi informasi tentang pembuat aplikasi.
Menu Tutup	*Menu*	Berfungsi untuk menutup aplikasi.
Button refresh	*Button*	Berfungsi merefresh tabel pembicara

4.4.3 Perancangan Antar Muka *Form* Hasil Pengujian



Gambar 4.10. Rancangan antar muka *form* pengujian

Tabel 4.7. Spesifikasi antar muka *form* pengujian

Nama objek	Jenis	Keterangan
Ukuran *Frame*	*Label*	Berfungsi untuk menampilkan ukuran frame
MFCC	*Label*	Berfungsi untuk menampilkan memasukan jumlah MFCC *coeffecient*.
Data Latih	*Label*	Berfungsi untuk menampilkan jumlah data latih per pembicara
Iterasi	*Label*	Berfungsi untuk menampilkan jumlah iterasi

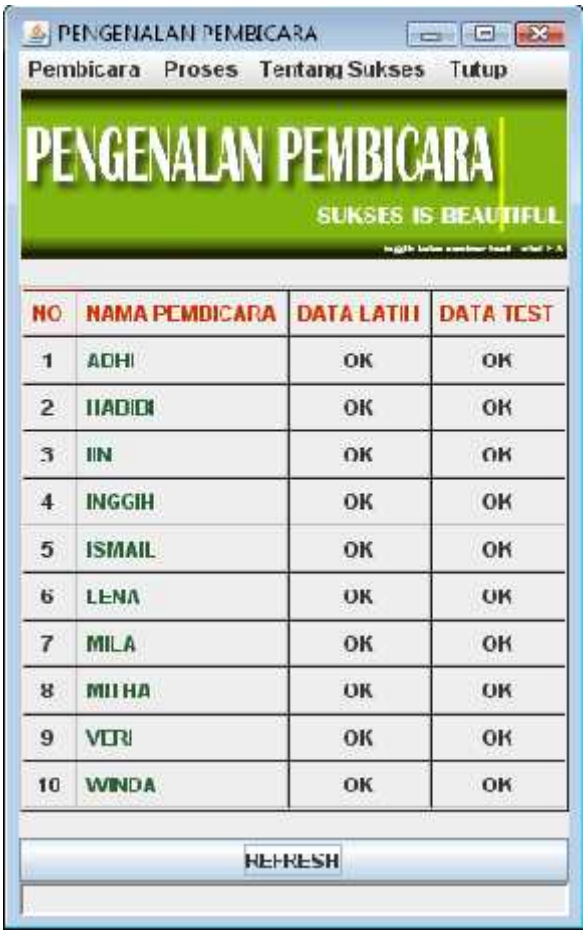
BAB V
IMPLEMENTASI DAN PENGUJIAN

Implementasi dan pengujian yang akan dilakukan dalam bagian ini adalah implementasi dari analisa aplikasi pengenalan pembicara yang bisa menguji berapa akurasi ketepatan kombinasi antara metode MFCC dan SOM dalam mendeteksi pembicara.

5.1 Implementasi

Sub ini akan membahas tentang implementasi *form* utama, *form* pelatihan dan *form* hasil pengujian.

5.1.1 Implementasi *Form* Utama



Gambar 5.1. Implementasi *form* utama

Pada tabel tersebut terdapat tabel yang berisi daftar para pembicara serta keberadaan data latih dan data tesnya. Jika data ada maka di selnya akan tertulis OK tetapi jika tidak maka akan tertulis kosong. Rincian tentang objek-ojek yang ada pada *form* utama dapat dilihat pada tabel 4.5.

5.1.2 Implementasi *Form* Pelatihan

Gambar 5.2. Implementasi *form* pelatihan

Form ini memiliki beberapa objek yang penjelasannya dapat dilihat pada tabel 4.6. Jika tombol latih ditekan maka akan dilakukan dua buah proses yaitu ektrasi ciri dengan MFCC dan pengenalan pola dengan SOM. *Pseudo code* untuk aksi yang untuk tombol latih akan dijelaskan pada sub bab setelah ini.

5.1.2.1 *Pseudo Code* MFCC

Pseudo code pada MFCC memiliki 8 buah *function* utama yang terdiri dari 6 buah *function* untuk perhitungan matematis dan 2 buah untuk proses utama. Berikut *function-function* tersebut.

1. *Function* untuk perhitungan matematis

a. linToMelFreq

Function ini untuk mengubah frekuensi yang masuk menjadi frekuensi *mel*. *Pseudo code*-nya dapat dilihat di Algoritma 5.1.

```
double linToMelFreq(parameter : double inputFreq)
begin
      return (2595.0*(log(1.0 + inputFreq/700.0)/log(10.0)))
end
```

Algoritma 5.1. Frekuensi normal ke frekuensi *mel*

b. melToLinFreq

Function ini untuk mengubah dari frekuensi *mel* ke frekuensi normal. *Pseudo code*-nya dapat dilihat di Algoritma 5.2.

```
double melToLinFreq(input double inputFreq)
begin
      return (700.0*(pow(10.0,(inputFreq/2595.0))-1.0))
end
```

Algoritma 5.2. Frekuensi *mel* ke frekuensi normal

c. getDCTMatrix

Function ini untuk mempersiapkan matrik untuk melakukan *discreate cosines transform*. *Pseudo code*-nya dapat dilihat di Algoritma 5.3.

```
Matrix getDCTMatrix()
begin
    double k → phi/numberFilters
    double w1 → 1.0/(sqrt(numberFilters))
    double w2 → sqrt(2.0/numberFilters)
      Matrix matrix → Matrix(numberCoefficients, numberFilters)
    for(int i → 0; i < numberCoefficients; i++)
    begin for(int j → 0; j < numberFilters; j++)
      begin  if(i = 0)  matrix.set(i, j, w1 * cos(k*i*(j + 0.5)))
             else  matrix.set(i, j, w2 * cos(k*i*(j + 0.5)))
    end  end
      matrix → matrix.get(1,numberCoefficients-1,0,
numberFilters-1)
    return matrix
end
```

Algoritma 5.3. DCT matrik

d. getMelFilterBankBoundaries

Function ini berfungsi membuat batas tepi untuk *mel filter bank*. *Pseudo code*-nya dapat dilihat di Algoritma 5.4.

```
double[] getMelFilterBankBoundaries(parameter : double minFreq,
double maxFreq, int numberFilters)
begin
    double[] centers → double[numberFilters + 2]
    double maxFreqMel, minFreqMel, deltaFreqMel, nextCenterMel,
nextCenter

    maxFreqMel → call linToMelFreq(maxFreq)
    minFreqMel → call linToMelFreq(minFreq)
    deltaFreqMel → (maxFreqMel - minFreqMel)/(numberFilters + 1)

    nextCenterMel → minFreqMel
    for(int i → 0; i < centers.length; i++)
    begin
      centers[i] → call melToLinFreq(nextCenterMel)
      nextCenterMel → call nextCenterMel+deltaFreqMel
    end

    centers[0] → minFreq
    centers[numberFilters + 1] → maxFreq

    return centers
end
```

Algoritma 5.4. ***Mel filter bank boundaries***

e. getMelFilterWeight

Function ini berfungsi mencari berat *mel filter*. *Pseudo code*-nya dapat dilihat di Algoritma 5.5.

```
double getMelFilterWeight(parameter : int filterBank, double freq,
double[] boundaries)
begin
    double result → 0
```

```
    double start → boundaries[filterBank - 1]
    double center → boundaries[filterBank]
    double end → boundaries[filterBank + 1]
    double height → 2.0/(end - start)
    if(freq >= start And freq <= end)
    begin
      if(freq < center)
      begin
        result → (freq - start) * (height/(center - start))
      end
      else
      begin
        result → height+((freq-center)*(-height/(end - center)))
      end
    end
    return result
end
```

Algoritma 5.5. ***Mel filter weight***

f. getMelFilterBank

Function ini berfungsi untuk mencari *mel filter bank*. *Pseudo code*-nya dapat dilihat di Algoritma 5.6.

```
Matrix getMelFilterBanks()
Begin
    double[] boundaries → call
getMelFilterBankBoundaries(minFreq, maxFreq, numberFilters)
    double[][] matrix → double[numberFilters][]

    for(int i → 1; i <= numberFilters; i++)
    begin
         double[] filter → new double[(windowSize/2)+1]
         for(int j → 0; j < filter.length; j++)
         begin
           double freq → baseFreq * j
        filter[j] → call getMelFilterWeight(i, freq, boundaries)
```

```
      end     matrix[i-1] → filter    end    return
Matrix(matrix, numberFilters, (windowSize/2)+1)
end
```

Algoritma 5.6. ***Mel filter bank***

2. *Function* untuk proses-proses utama

a. *process*

Function ini berfungsi mendapatkan seluruh nilai MFCC berdasarkan input yang dimasukkan. *Pseudo code*-nya dapat dilihat di Algoritma 5.7.

```
double[][] process(parameter : double[] input)
begin

    double[][] mfcc → double[(input.length/hopSize)-
1][numberCoefficients]

    for(int i → 0, pos → 0; pos < input.length - hopSize; i++,
pos+=hopSize)
    begin
      mfcc[i] → call processWindow(input, pos)
    end
    return mfcc
end
```

Algoritma 5.7. Proses utama MFCC

b. *processWindow*

Function ini berfungsi mendapatkan nilai MFCC per *frame*. *Pseudo code*-nya dapat dilihat di Algoritma 5.8.

```
double[] processWindow(parameter : double[] window, int start)
begin
    for (int j → 0; j < windowSize; j++)
    begin
      buffer[j] → window[j + start]
    end
```

```
    FFT(buffer)
    Matrix x → new Matrix(buffer, windowSize)
    x → (call getMelFilterBanks) * x
    x → log(x)
    x → (call getDctMatrix) * x
    return x

end
```

Algoritma 5.8. MFCC per *frame*

5.1.2.2 *Pseudo Code* SOM

Pseudo code pada SOM terdiri dari 3 *function* utama. Berikut *pseudo code*-nya.

1. euclideanDistance

Function ini berfungsi menghitung jarak antara satu vektor ke vektor lain. *Pseudo code*-nya dapat dilihat di Algoritma 5.9.

```
double euclideanDistance(parameter : Vector<Double> item1,
Vector<Double> item2)
begin
      double sum → 0
      for (int i → 0;i<item1.size();i++)
      begin

      Double item1Value → item1.elementAt(i)
            Double item2Value → item2.elementAt(i)
            sum → sum + ((item2Value.doubleValue() -
item1Value.doubleValue()) * (item2Value.doubleValue() -
item1Value.doubleValue()))
      end

      return sqrt(sum)
end
```

Algoritma 5.9. *Euclidean Distance*

2. getBMU

Dari perhitungan-perhitungan jarak *Euclidean* antara satu vektor yang satu ke vektor yang lain, diambil bobot terkecil sebagai vektor pemenang. *Function* ini berfungsi mengambil indeks vektor pemenang tersebut. *Pseudo code*-nya dapat dilihat di Algoritma 5.10.

```
int getBMU(Vector<Double> dataItem)
begin
      double minDist → MAX_VALUE
      int indexMinDist → -1
      double value
      for (int i → 0; i< codebook.getNumberOfRows(); i++)
      begin
            value → call euclideanDistance(dataItem,
codebook.getRow(i))
            if (value < minDist)
            begin
                  minDist → value
                  indexMinDist → i
            end
      end
      return indexMinDist
end
```

Algoritma 5.10. ***Best matching unit***

3. train

Function ini berfungsi untuk melakukan pelatihan. *Pseudo code*-nya dapat dilihat di Algoritma 5.11.

```
train() {
      int iterations → trainingLength*data.getNumberOfRows()
      double alpha → 1
      double alpha_decrease → 1/iterations
      double radius → 1
      Vector<Double> x
      Vector<Double> m
```

```
      Random randItem → Random()                          for (int i
→ 0; i<iterations; i++) {          x →
data.getRow(randItem.nextInt( data.getNumberOfRows()))
            int bmu → call getBMU(x)
            for (int j → 0; j<codebook.getNumberOfRows(); j++) {
                  m → codebook.getRow(j)
                  radius → exp((mapunitDistance(bmu, j)*-1)/
(2*alpha*alpha))
                  codebook.addRowValues(vectorDistanceMultiply(x,
m, alpha*radius), j)
            }
            alpha → alpha - alpha_decrease
      }
}
```

Algoritma 5.11. *Train SOM*

5.1.3 Implementasi *Form* Hasil Pengujian

Gambar 5.3. Implementasi *form* hasil pengujian

Gambar 5.3 adalah *form* hasil pengujian. Pada sudut kiri atas *form* tersebut terdapat 4 buah parameter. Penjelasan mengenai parameter-parameter tersebut adalah sebagai berikut:

1. Ukuran *frame*

   Ukuran *frame* adalah jumlah *byte* yang digunakan untuk satu buah *frame* dalam proses *framing* untuk setiap data pelatihan dan pengujian. Satuan yang digunakan adalah *byte*. Pada gambar 5.3 terlihat ukuran *frame* adalah 512 *byte*. Hal ini berarti dalam satu buah *frame* terdapat 512 buah *byte*.

2. Jumlah MFCC *coefficient*

   Jumlah MFCC *coeffecients* adalah jumlah koefesien yang dihasilkan dari setiap *frame* untuk setiap data pelatihan dan pengujian. Pada gambar 5.3 terlihat ukuran *frame* 512 *byte* dan jumlah MFCC *coefficient* adalah 30. Hal ini dari setiap *frame-frame* berukuran 512 *byte* tersebut akan dihasil 30 koefesien dalam proses MFCC.

3. Jumlah data latih

   Jumlah data latih adalah jumlah *frame* yang diambil dari setiap suara pelatihan. Pada gambar 5.3. terlihat ukuran frame 512 *byte* dan jumlah data latih adalah 200 buah. Hal ini berarti dari setiap suara pelatihan akan diambil 200 buah *frame* yang berukuran 512 *byte*.

4. Jumlah iterasi

   Sedangkan jumlah iterasi jumlah iterasi yang dilakukan pada proses pengenelan pola.

Pada gambar 5.3 juga terdapat sebuah tabel yang terdiri dari 3 buah kolom. Penjelasan untuk masing masing kolom adalah sebagai berikut:

1. Kolom pembicara

   Kolom pembicara menampilkan nama-nama pembicara yang terdapat dalam *database*.

2. Kolom rinci

   Pada suara pelatihan, *frame-frame* setiap pembicara yang telah diekstrasi ciri menggunakan MFCC dikenali pola umumnya melalui

pengenalan pola yaitu SOM. Begitu juga pada data pengujian, juga dilakukan ekstrasi ciri pada setiap *frame*. Hasil ekstrasi ciri masing-masing *frame* pada data pengujian tersebut dihitung jarak *Euclidean*-nya terhadap masing-masing pola umum yang didapat dari pengenalan pola tadi. Setelah jaraknya diukur dengan masing pola maka ditentukan mana jarak terkecil. Jarak terkecillah yang akan menjadi pola pemenang, maka *frame* tersebut ditetapkan sebagai suara si pemilik pola tersebut. Setiap *frame* data pengujian pada masing-masing pembicara dilakukan hal yang sama, setelah itu ditentukan berapa jumlah *frame* yang mirip pola suara masing-masing pembicara. Untuk contohnya lihat gambar 5.3, setiap sub kolom pada kolom rinci mewakili pola masing-masing pembicara. Disana terlihat suara Adhi (baris pertama) 333 *frame* mirip pola suara Adhi, 27 *frame* mirip pola suara Habibi, 26 *frame* mirip pola suara Iin, 108 *frame* suara mirip pola suara Inggih, 38 *frame* mirip pola suara Ismail, 31 *frame* mirip pola suara Lena, 2 *frame* mirip pola suara Mila, 14 *frame* mirip pola suara Mitha, 54 *frame* mirip pola suara Veri dan 0 *frame* mirip pola suara Winda. Karena kemiripan dengan pola suara Adhi memiliki jumlah *frame* terbanyak maka suara pengujian tersebut terdeteksi sebagai suara Adhi. Pola suara dengan jumlah *frame* terbanyak akan diberikan warna merah.

3. Kolom persen

   Kolom persen adalah untuk menentukan akurasi hasil pengujian pada masing-masing pembicara. Suara Adhi (baris pertama) pada gambar 5.3 memiliki jumlah *frame* yang yang mirip dengan pola suaranya sebanyak 333 *frame* dari 633 *frame*, maka akurasinya adalah (333/633)x100 sehingga didapat akurasi sebesar 53%.

5.2 Pengujian

Pengujian dilakukan dengan perubahan beberapa parameter. Hal ini untuk menunjukan pengaruh perubahan parameter tersebut terhadap kemampuan

pendeteksian pembicara. Parameter-parameter yang akan diuji adalah pengaruh perubahan iterasi, pengaruh perubahan MFCC *coefficient* dan pengaruh perubahan jumlah data latih. Kombinasi antara dua atau lebih perubahan parameter juga dilihat pengaruhnya, yaitu pengaruh perubahan antara MFCC *coeffecient* dan jumlah iterasi.

5.2.1 Pengujian Pengaruh Iterasi

Pada pengujian ini digunakan ukuran *frame* 512 *byte*, MFCC *coefficient* 30, jumlah data latih sebanyak 200 *frame* dan jumlah iterasi yang digunakan adalah antara 1000 sampai 10000 iterasi. Pengujian dilakukan sebanyak 10 kali dengan jumlah iterasi terus meningkat. Hasil pengujian dapat dilihat di tabel 5.1.

Tabel 5.1. Pengaruh iterasi

No	Frame (*Byte*)	MFCC	Data Latih (*Frame*)	Jumlah Iterasi	Jumlah Terdeteksi	Rata-rata Akurasi
1	512	30	200	1000	60%	37.10%
2	512	30	200	2000	70%	39.30%
3	512	30	200	3000	70%	41.30%
4	512	30	200	4000	70%	42.30%
5	512	30	200	5000	70%	43.30%
6	512	30	200	6000	70%	42.80%
7	512	30	200	7000	70%	44%
8	512	30	200	8000	80%	43.60%
9	512	30	200	9000	80%	43.70%
10	512	30	200	10000	80%	43.30%
Rata-rata					**72%**	**42%**

Tabel 5.1. menunjukan bahwa pengujian ke 2 (2000 iterasi) mengalami peningkatan jumlah data yang terdeteksi dari pengujian pertama (1000 iterasi), yaitu dari 60% menjadi 70%. Pengujian kedua juga mengalami peningkatan jumlah rata-rata akurasi, yaitu dari 37.10% menjadi 39.30%. Sedangkan dari pengujian ke 2 sampai pengujian ke 7 menunjukkan peningkatan jumlah iterasi tidak membuat peningkatan jumlah pembicara yang terdeteksi, tetapi tingkat rata-rata akurasi meningkat secara signifikan. Pengujian ke 8 sampai ke 9 terlihat rata-rata akurasi cenderung menurun dari pengujian ke 7, tetapi meskipun begitu

jumlah pembicara yang terdeteksi meningkat menjadi 80%. Dari hasil 10 pengujian tersebut dapat diambil kesimpulan bahwa peningkatan jumlah iterasil secara relatif dapat meningkatkan jumlah pembicara yang terdeteksi dan tingkat akurasi pendeteksian.

5.2.2 Pengujian Pengaruh MFCC *coefficient*

Pada pengujian ini digunakan ukuran *frame* 512 *byte*, jumlah data latih sebanyak 200 *frame*, jumlah iterasi sebanyak 8000 kali dan jumlah MFCC *coefficient* yang digunakan adalah antara 30 sampai 70 koefesien. Pengujian dilakukan sebanyak 5 kali dengan jumlah MFCC *coeffecient* terus meningkat. Hasil pengujian dapat dilihat di tabel 5.2.

Tabel 5.2. Pengaruh MFCC *coefficient*

No	Frame (*Byte*)	MFCC	Data Latih (*Frame*)	Jumlah Iterasi	Jumlah Terdeteksi	Rata-rata Akurasi
1	512	30	200	8000	80%	43.60%
2	512	40	200	8000	80%	44.60%
3	512	50	200	8000	80%	44.20%
4	512	60	200	8000	80%	44.60%
5	512	70	200	8000	80%	44.80%
Rata-rata					**80%**	**44%**

Tabel 5.2. menunjukan bahwa dari pengujian ke 1 sampai ke 5 tidak mengalami peningkatan jumlah pembicara yang terdeteksi yaitu 80%. Sedangkan rata-rata akurasi relatif meningkat.

5.2.3 Pengujian Pengaruh Jumlah Data Latih

Hasil pengujian dapat dilihat di Tabel 5.3.

Tabel 5.3. Pengaruh jumlah data latih

No	Frame (*Byte*)	MFCC	Data Latih (*Frame*)	Jumlah Iterasi	Jumlah Terdeteksi	Rata-rata Akurasi
1	512	30	100	5000	70%	40.70%
2	512	30	150	5000	70%	42.20%
3	512	30	200	5000	70%	43.30%
4	512	30	250	5000	70%	43.90%

Tabel 5.3. Pengaruh jumlah data latih (lanjutan)

No	Frame (*Byte*)	MFCC	Data Latih (*Frame*)	Jumlah Iterasi	Jumlah Terdeteksi	Rata-rata Akurasi
5	512	30	300	5000	70%	44.20%
6	512	30	350	5000	70%	43.60%
7	512	30	400	5000	70%	43.60%
Rata-rata					**70%**	**43%**

Pada pengujian ini digunakan ukuran *frame* 512 *byte*, jumlah iterasi sebanyak 5000 kali, jumlah MFCC *coefficient* adalah 30 koefesien dan jumlah data latih adalah antara 100 sampai 400 *frame*. Pengujian dilakukan sebanyak 7 kali dengan jumlah iterasi terus meningkat.

Tabel 5.3. menunjukan bahwa dari pengujian ke 1 sampai ke 7 tidak mengalami peningkatan jumlah pembicara yang terdeteksi, yaitu 70%. Sedangkan rata-rata akurasi meningkat dari pengujian ke 1 sampai ke 5, sedangkan dari pengujian ke 6 sampai ke 7 mengalami penurunan akurasi. Ini menunjukan bahwa peningkata jumlah data latih tidak selalu memberikan hasil yang lebih baik.

5.2.4 Pengujian Pengaruh Jumlah MFCC *Coefficient* dan Jumlah Iterasi

Berdasarkan pengujian pengaruh MFCC *coefficient* dan jumlah iterasi, terlihat kedua parameter ini dapat meningkatkan jumlah pembicara yang terdeteksi dan nilai rata-rata akurasi pendeteksian. Oleh sebab itu pengujian kali akan menguji pengaruh peningkatan keduanya terhadap akurasi pendeteksian pembicara.

Pada pengujian ini digunakan ukuran *frame* 512 *byte*, jumlah data latih sebanyak 200 *frame*, jumlah iterasi yang digunakan adalah antara 6000 sampai 10000 kali dan jumlah MFCC *coefficient* yang digunakan adalah antara 50 sampai 70 koefesien. Pengujian dilakukan sebanyak 5 kali dengan jumlah MFCC *coefficient* dan jumlah iterasi terus meningkat. Hasil pengujian dapat dilihat di tabel 5.5.

Tabel 5.4. Pengaruh jumlah MFCC *coefficient* dan iterasi

No	Frame (*Byte*)	MFCC	Data Latih (*Frame*)	Jumlah Iterasi	Jumlah Terdeteksi	Rata-rata Akurasi
1	512	30	200	6000	70%	43.40%
2	512	40	200	7000	70%	44.80%
3	512	50	200	8000	80%	44.20%
4	512	60	200	9000	80%	44.50%
5	512	70	200	10000	80%	44.40%
Rata-rata					**76%**	**44%**

Tabel 5.4. menunjukan bahwa peningkatan MFCC *coefficient* dan jumlah iterasi berpengaruh terhadap jumlah pembicara yang terdeteksi dan rata-rata akurasi pendeteksian. Pada pengujian ke 2 terlihat bahwa terjadi peningkatan akurasi terhadap pengujian pertama. Pada pengujian ke 3 terjadi penurunan akurasi, tetapi terjadi peningkatan jumlah orang yang terdeteksi dari 70% ke 80%. Pada pengujian ke 5 terjadi penurunan akurasi, mungkin ini karena jumlah iterasi yang kurang banyak.

BAB VI
PENUTUP

6.1 Kesimpulan

Kesimpulan dari penelitian tugas akhir ini adalah:

1. Aplikasi ini dapat menguji berapa akurasi pengenalan pembicara dengan menggunakan kombinasi metode MFCC dan SOM.
2. Peningkatan iterasi dan MFCC *coefficient* dapat meningkatkan akurasi pendeteksian pembicara.
3. Peningkatan jumlah data latih tidak selalu memberikan hasil yang lebih baik dalam nilai akurasi.
4. Meskipun berhasil mendeteksi 80% pembicara tetapi rata-rata akurasi hanya 44%. Ini terjadi bukan karena MFCC, karena saat metode ini di kombinasikan dengan metode pengenalan pola lain seperti *K-Means* dapat mencapai nilai akurasi yang lebih baik. Mungkin ini terjadi karena metode SOM yang kurang baik untuk pengenalan pola dalam identifikasi pembicara.
5. Hasil kombinasi parameter terbaik adalah *frame size* 512, MFCC *coeffecient* 70, jumlah iterasi 8000 kali dan jumlah data latih 200 *frame*.

6.2 Saran

Agar aplikasi ini bermanfaat dan berdaya guna dimasa sekarang dan yang akan datang, maka penulis memberikan saran sebagai berikut:

1. Aplikasi ini perlu ditambahkan parameter-parameter MFCC dan SOM yang lain. Hal ini agar aplikasi menjadi lebih dinamis.
2. Aplikasi ini perlu dibuat media untuk menyimpan vektor bobot hasil pelatihan. Agar hasil pelatihan dapat digunakan untuk kepentingan lainnya.

3. Aplikasi ini perlu ditambahkan pendeteksian suara secara langsung untuk mendeteksi suara lain yang dimasukan selain *file* suara yang telah ada di data pengujian.
4. Aplikasi ini perlu diuji pada jumlah pembicara yang lebih banyak, agar hasil pengujian lebih representatif.
5. Pada SOM perlu ditambahkan kode program untuk menghentikan iterasi apabila vektor bobot hasil pelatihan sudah kovergen.
6. Perlu dicoba metode yang lebih baik dari SOM, seperti *Growing Hierarchical SOM* (*GHSOM*) dan *Incremental Gradient Descent* (*IGDSOM*) agar bisa menghasilkan akurasi yang lebih tinggi.

DAFTAR PUSTAKA

Al-Akaidi, Marwan, *Call admission control for multitier networks with integrated voice and data services*, 2007

Budhi, Gregorius Satia, Liliana dan Steven Harryanto, "*Cluster Analysis untuk Memprediksi Talenta Pemain Basket Menggunakan Jaringan Saraf Tiruan Self Organizing-Map (SOM)* ", UK Petra, Surabaya, 2006

Ganchev, Siafarikas, *Overlapping Wavelet Packet Features for Speaker Verification. INTERSPEECH*'05, 2005

Haykin, Simon, "*Neural Network A Comprehensive Foundation*", Edisi 2, Halaman 465-501, Pearson Prentice Hall, Singapore, 2005

Hermawan, Arif, "*Jaringan Saraf Tiruan: Teori dan Aplikasi"*, Penerbit Andi, Yogyakarta, 2006

Jacobson, Ivar. *Object-Oriented Software Engineering (A Use Case Driven Approach).* ACM Press, Addison-Wesley, Edinburgh Gate, England. 1992.

Jha, Girish Kumar, "*Artificial Neural Networks and Its Applications"*, I.A.R.I, New Delhi, 2007

Kusumadewi, Sri, "*Artificial Intelligence (Teknik dan Aplikasinya)",* Graha Ilmu, Yogyakarta, 2003

Lundsgaard. Mikael, *"JOPSPEECH", 2006, Sumber:* http://www.jopdesign.com/files/ext/SpokenLanguage.pdf, diakses tanggal 21 Februari 2011

Madarum, Arum, "*Clustering Specimen Daun Dikotiledon Dengan Menggunakan SOM",* Institut Pertanian Bogor, Bogor, 2003

Muda, Lindasalwa. Mumtaj Begam dan I. Elamvazuthi, *"Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Technoques"*, 2010, Sumber: https://sites.google.com/site/journalofcomputing, diakses tangganl 21 Februari 2011.

Nugroho, Adi. "*Rekayasa Perangkat Lunak Berorientasi Objek dengan Metode USDP (Unified Software Development Process)"*. Penerbit ANDI Yogyakarta, Yogyakarta. 2010.

Park, Alez. S, "*ASR DependentTechniques for Speaker Recognition*, Massachusetts Institude of Technology", 2002, Sumber : groups.csail.mit.edu/sls/ publications/2002/park_meng_thesis.pdf, diakses tanggal 26 Februari 2011

Rao, V.B dan Rao, H.V, "*Neural Network and Fuzzy Logic",* Management Information Source, New York, 1993

Rumelhart, D.E. and McClelland, J.L., "*Parallel Distributed Processing: Explorations in the Microstructure of Cognition"*. Cambridge, MA: MIT Press, 1986

Siang, Jong Jek, "*Jaringan Syaraf Tiruan dan Pemograman Menggunakan MATLAB"*. Andi, Yogyakarta, 2005

Widrow, G. and Hoff, M.E., "*Adaptive switching circuits*", *Western Electronic Show and Convention, Convention Record*, 1960

http://www.ehow.com/list_7690647_speaker-recognition-methods.html, diakses tanggal 28 Februari 2011

http://www.ifp.illinois.edu/~minhdo/teaching/speaker_recognition/speaker_recognition.htmdiakses tanggal 28 Februari 2011

http://www.speaker-recognition.org/, diakses tanggal 28 Februari 2011

http://id.wikipedia.org/wiki/Jaringan_saraf_tiruan, diakses tanggal 15 Februari 2011

http://www.ibm.com, diakses tanggal 28 Februari 2011

http://www.cis.hut._/harri/thesis/ valpola_thesis/node34.html, diakses tanggal 15 Februari 2011