BAB 2
LANDASAN TEORI

2.1 Pengenalan Suara

Dalam kehidupan sehari-hari, manusia melakukan berbagai jenis komunikasi dengan sesama manusia, misalnya: *body language*, berbicara (*speech*) dan lain-lain. Di antara banyak komunikasi yang dilakukan oleh manusia, berbicara memberikan paling banyak informasi penting dan paling efektif dalam berkomunikasi. Informasi-informasi tersebut antara lain : mengidentifikasi jenis kelamin, keadaan kesehatan, emosi, serta identitas pembicara [1]. Pengenalan suara dapat dikategorikan menjadi 2 bagian, yaitu sebagai berikut :

1. Speech recognition

Speech recognition adalah suatu teknologi yang mampu mengubah *input* suara menjadi suatu *output* tertentu, seperti contoh *output* penulisan *text* maupun *output running* program. *Speech recognition* merupakan teknologi yang dapat mengenali pembicaraan tanpa memperdulikan siapa pembicaranya. *Speech recognition* yang dikenal sebagai *Automatic Speech Recognition* (ASR), atau komputer pengenalan suara yang merupakan proses mengkonversi sinyal suara ke urutan kata-kata, melalui sebuah algoritma di implementasikan sebagai program komputer.

Speech Recognition juga merupakan pengenalan pola, dimana ada dua fase dalam pengenalan pola diawasi, yaitu, pelatihan dan pengujian. Proses ekstraksi fitur yang relevan untuk klasifikasi umum di kedua fase. Selama fase traning, parameter dari model klasifikasi yang diperkirakan dengan menggunakan sejumlah besar contoh kelas (*data training*). Selama fase pengujian atau fase pengenalan, fitur pola uji (*data speech test*) dicocokan dengan model yang dilatih dari masing-masing dan setiap kelas. Pola tes itu kemudian dinyatakan ke dalam model yang memiliki pola tes terbaik [8].

2. Speaker recognition

Speaker recognition adalah suatu proses yang bertujuan mengenali siapa yang sedang berbicara berdasarkan informasi yang terkandung dalam gelombang suara yang di inputkan. *Speaker recognition* dibagi menjadi 2 bagian, yaitu [1]:

a. *Speaker verification*

Speaker verification adalah proses verifikasi seorang pembicara, dimana sebelumnya telah diketahui identitas pembicara tersebut berdasarkan data yang telah diinputkan. *Speaker verification* melakukan perbandingan *one to one* (1:1) dalam arti bahwa fitur-fitur suara dari seorang pembicara dibandingkan secara langsung dengan fitur-fitur seorang pembicara tertentu yang ada dalam sistem. Bila hasil perbandingan (skor) tersebut lebih kecil atau sama dengan batasan tertentu (*treshold*), maka pembicara tersebut diterima, bila tidak maka akan ditolak (dengan asumsi semakin kecil skor berarti kedua sampel semakin mirip) [1].

b. Speaker identification

Speaker identification adalah proses mendapatkan identitas dari seorang pembicara dengan membandingkan fitur-fitur suara yang diinputkan dengan semua fitur-fitur dari setiap pembicara yang ada dalam *databas*e. Berbeda dengan pada *speaker verification*, proses ini melakukan perbandingan *one to many* (1:N) [1].

2.2 Jenis Pengenalan Suara

Sistem pengenalan suara dapat dipisahkan dalam beberapa kelas yang berbeda dengan mendeskripsikan jenis ucapan-ucapan mereka yaitu [9] :

a. *Isolated Words*

Isolated word recognizers biasanya memerlukan setiap ucapan harus tenang (karena kurangnya sinyal *audio*) pada kedua sisi sampel. Menerima satu kata atau ucapan tunggal pada satu waktu. Sistem ini memiliki pernyataan "Dengar / Tidak-Dengar", dimana jenis ini membutuhkan pembicara untuk menunggu dalam mengucapkan kata demi kata (biasanya melakukan pengolahan selama jeda).

b. *Connected Words*

Sistem *connected words* (ucapan yang terhubung) hampir serupa dengan *isolated words*, namun memungkinkan ucapan jeda minimal antara mereka.

c. *Continuous Speech*

Continuous speech recognizers memungkinkan pengguna untuk berbicara hampir secara alami, sedangkan komputer menentukan kontennya. Pengenalan

dengan kemampuan *continuous speech* adalah salah satu yang paling sulit karena mereka menggunakan metode khusus untuk menentukan batas ucapan.

d. *Spontaneous Speech*

Pada tingkat dasar, itu dapat dianggap sebagai pidato yang terdengar secara alami. Sebuah sistem *speech recognition* dengan kemampuan *spontaneous speech* harus mampu menangani berbagai variasi dari fitur-fitur speech alami seperti kata-kata yang dijalankan bersama-sama, "ums" dan "ahs", dan bahkan sedikit gagap.

2.3 Teknik-teknik speech recognition

Ada 4 teknik dalam *Speech Recognition* yaitu [8] :

a. *Speech Analysis Technique*

Speech data mengandung berbagai jenis informasi yang menunjukkan identitas pembicara. Tahapan *speech analysis* berkaitan dengan ukuran frame yang cocok untuk segmentasi sinyal suara dalam analisa dan ekstraksi yang lebih lanjut.

b. *Feature Extraction Technique*

Ekstraksi fitur *speech* dalam kategorisasi masalah adalah tentang mengurangi dimensi dari vektor input ketika mempertahankan membedakan kekuatan sinyal. Dari pembentukan dasar *speaker identification* dan *speaker verification*, bahwa jumlah pelatihan dan vektor uji diperlukan untuk masalah klasifikasi yang tumbuh dengan dimensi masukan yang diberikan sehingga kita membutuhkan fitur ekstraksi dari sinyal suara.

c. *Modeling Technique*

Tujuan dari *modeling technique* adalah untuk menghasilkan speaker models yang menggunakan fitur vektor pembicara khusus. *Speaker modeling technique* dibagi menjadi dua klasifikasi yaitu *speaker recognition* dan *speaker identification*. *Speaker identification technique* secara otomatis mengidentifikasi siapa yang berbicara berdasarkan informasi individual yang terintegrasi dalam sinyal suara. *Speaker recognition* juga dibagi menjadi dua bagian yaitu *speaker dependant* dan *speaker independent*. Dalam modus *speaker independent* dari *speech recognation*, komputer harus mengabaikan karakteristik khusus pembicara dari sinyal suara dan mengekstrak pesan yang dimaksudkan. Disisi lain dalam kasus *speaker recognation*

machine harus mengekstrak karakteristik pembicara dalam sinyal akustik. Tujuan utama dari *speaker identification* adalah membandingkan sinyal pidato dari pembicara tak dikenal ke database pembicara yang sudah dikenal. Sistem ini dapat mengenali pembicara, yang telah dilatih dengan sejumlah pembicara. *Speaker recognition* juga dapat dibagi menjadi dua metode, *text dependent* dan *text independent*. Dalam metode *text dependent* pembicara mengatakan kata kunci atau kalimat yang memiliki teks yang sama untuk menguji pelatihan dan pengenalan. Sedangkan *text independent* tidak bergantung pada teks tertentu yang diucapkan.

d. *Matching Techniques*

Mesin *speech recognition* mencocokkan sebuah kata yang terdeteksi dengan kata yang sudah diketahui salah satu dari teknik-teknik berikut :

1. Whole Word Matching

Mesin membandingkan sinyal *digital-audio* yang datang terhadap template rekaman kata. Teknik ini membutuhkan waktu lebih sedikit pengolahan dari pencocokan sub-kata, tetapi mensyaratkan bahwa pengguna (atau seseorang) merekam setiap kata yang akan dikenali, kadang-kadang beberapa ratus ribu kata. Template seluruh kata juga membutuhkan memori penyimpanan yang besar (antara 50 dan 512 byte per kata) dan hanya praktis jika pengenalan kosakata tersebut sudah dikenal ketika aplikasi dikembangkan.

2. Sub Word Matching

Mesin mencari sub-kata, biasanya fonem dan kemudian melakukan pengenalan pola lanjut. Teknik ini membutuhkan lebih banyak pemrosesan dari pencocokan seluruh kata, tetapi membutuhkan penyimpanan lebih sedikit (antara 5 dan 20 byte per kata). Selain itu, pengucapan kata dapat ditebak dari teks bahasa Inggris tanpa mengharuskan pengguna untuk berbicara kata yang sebelumnya.

2.4 Produksi Suara

Suara adalah sebuah *signal* yang merambat melalui media perantara. Suara dapat dihantarkan melalui media air, udara maupun benda padat. Dengan kata lain suara adalah gelombang yang merambat dengan frekuensi dan amplitude tertentu. Suara yang dapat didengar oleh manusia berkisar antara 20 Hz sampai dengan 20

KHz, dimana Hz adalah satuan dari frekuensi yang artinya banyaknya getaran per detik (cps / *cycle per second*) [1].

Perlengkapan produksi suara pada manusia secara garis besar terdiri dari jalur suara (*vocal track*) dan jalur hidung (*nasal track*). Jalur suara dimulai dari pita suara (*vocal cords*), celah suara (*glottis*) dan berakhir pada bibir. Jalur hidung dimulai dari bagian belakan langit-langit (*velum*) dan berakhir pada cuping hidung (*nostrils*) [1].

Proses menghasilkan suara dimulai dari udara masuk ke paru-paru melalui pernafasan, kemudian melalui trakea, udara masuk ke batang tenggorokan, dimana pada batang tenggorokan ini terdapat pita suara. Pita suara ini kemudian bergetar dengan frekuensi tertentu karena adanya aliran udara tersebut sehingga dihasilkan suara. Suara yang dihasilkan ini berbeda-beda sesuai dengan posisi lidah, bibir, mulut dan langit-langit pada saat itu [5].

Suara yang dihasilkan terdiri dari tiga bagian yaitu *voiced sound*, *unvoiced sound* dan *plosive sound*. *Voiced sound* terjadi jika pita suara bergetar dengan frekuensi antara 50 Hz sampai 250 Hz. Contoh *voiced sound* adalah bunyi pada kata "ah", "oh". *Unvoiced sound* terjadi jika pita suara tidak bergetar sama sekali. Contoh *unvoiced sound* adalah bunyi "shh". Sedangkan *plosive sound* terjadi jika pita suara tertutup sesaat kemudian tiba-tiba membuka. Contoh *plosive sound* adalah bunyi "beh" pada kata benar, "pah" pada kata pasar [5].

2.5 Klasifikasi sinyal eksitasi

Berdasarkan sinyal eksitasi yang dihasilkan pada proses produksi suara, sinyal suara ucapan dapat dibagi menjadi tiga bagian, yaitu :

1. Sinyal *silence*

Sinyal pada saat tidak terjadi proses produksi suara ucapan, dan sinyal yang diterima oleh pendengar dianggap sebagai bising latar belakang.

2. Sinyal *unvoiced*

Sinyal *unvoiced* terjadi pada saat pita suara tidak bergetar, dimana sinyal eksitasi berupa sinyal random.

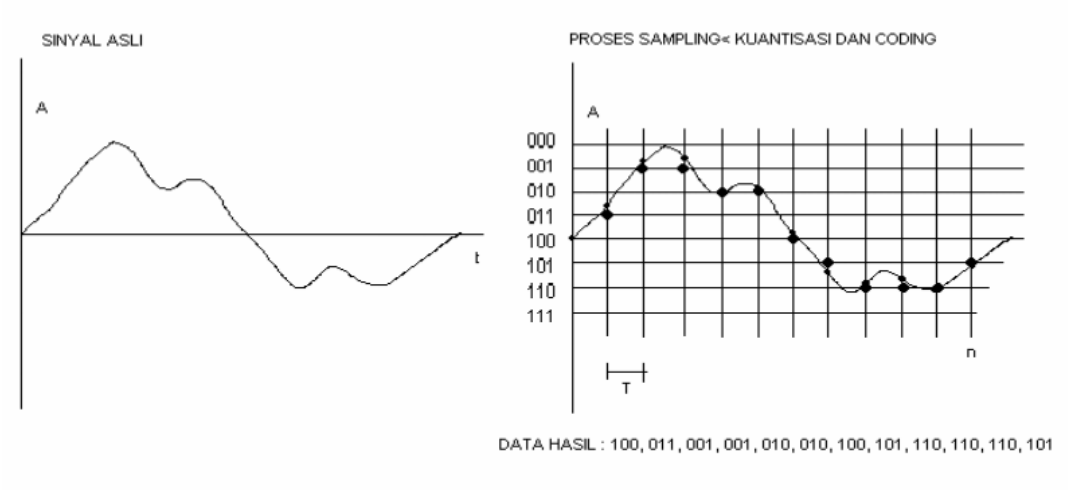
3. Sinyal *voiced*

Terjadi jika pita suara bergetar, yaitu pada saat sinyal eksitasi berupa sinyal pulsa kuasi-periodik. Selama terjadinya sinyal *voiced* ini, pita suara bergetar pada frekuensi fundamental – inilah yang dikenal sebagai pitch dari suara tersebut.

2.6 Konversi Analog Menjadi Digital

Signal – signal yang natural pada umumnya seperti *signal* suara merupakan *signal continue* dimana memiliki nilai yang tidak terbatas. Sedangkan pada komputer, semua signal yang dapat diproses oleh komputer hanyalah *signal discrete* atau sering dikenal sebagai istilah *digital signal*. Agar *signal* natural dapat diproses oleh komputer, maka harus diubah terlebih dahulu dari data *signal continue* menjadi *discrete*. Hal itu dapat dilakukan melalui 3 proses, diantaranya sebagai berikut [1] :

1. Proses sampling adalah suatu proses untuk mengambil data *signal continue* untuk setiap periode tertentu. Dalam melakukan proses sampling data, berlaku aturan Nyquist, yaitu bahwa frekuensi *s*ampling (sampling *rate*) minimal harus 2 kali lebih tinggi dari frekuensi maksimum yang akan di sampling. Jika signal sampling kurang dari 2 kali frekuensi maksimum signal yang akan di sampling, maka akan timbul efek *aliasing*. *Aliasing* adalah suatu efek dimana signal yang dihasilkan memiliki frekuensi yang berbeda dengan signal aslinya.
2. Proses kuantisasi adalah proses untuk membulatkan nilai data ke dalam bilangan-bilangan tertentu yang telah ditentukan terlebih dahulu. Semakin banyak *level* yang dipakai maka semakin akurat pula data *signal* yang disimpan tetapi akan menghasilkan ukuran data besar dan proses yang lama.
3. Proses pengkodean adalah proses pemberian kode untuk tiap-tiap data *signal* yang telah terkuantisasi berdasarkan *level* yang ditempati.



Gambar 2.1 Proses Pembentukan Sinyal *Digital*

Pada Gambar 2.1 terjadi proses konversi sinyal *analog* menjadi sinyal *digital*, gambar sebelah kiri merupakan gambar sinyal asli yang masih berbentuk sinyal *analog* sedangkan gambar sebelah kanan merupakan gambar setelah melakukan proses konversi sinyal analog menjadi sinyal *digital* yang telah melalui proses sampling, kuantisasi dan pengkodean. Pada gambar sebelah kanan didapatkan data hasil dalam bentuk kode yaitu 100, 011, 001, 001, 010, 010, 010, 100, 101, 110, 100 dan 101.

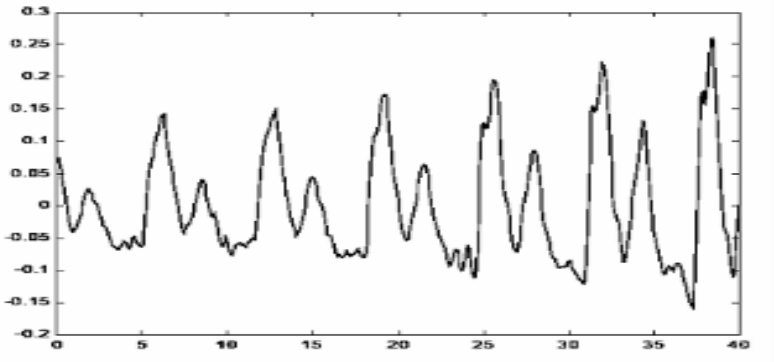
2.7 Mel Frequency Cepstrum Coefficients

MFCC (*Mel Frequency Cepstrum Coefficients*) merupakan salah satu metode yang banyak digunakan dalam bidang *speech technology*, baik *speaker recognition* maupun *speech recognition.* Metode ini digunakan untuk melakukan *feature extraction*, sebuah proses yang mengkonversikan *signal* suara menjadi beberapa parameter [2]. Beberapa keunggulan dari metode ini adalah:

a. Mampu untuk menangkap karakteristik suara yang sangat penting bagi pengenalan suara, atau dengan kata lain dapat menangkap informasi-informasi penting yang terkandung dalam *signal* suara.

b. Menghasilkan data seminimal mungkin, tanpa menghilangkan informasi-informasi penting yang dikandungnya.

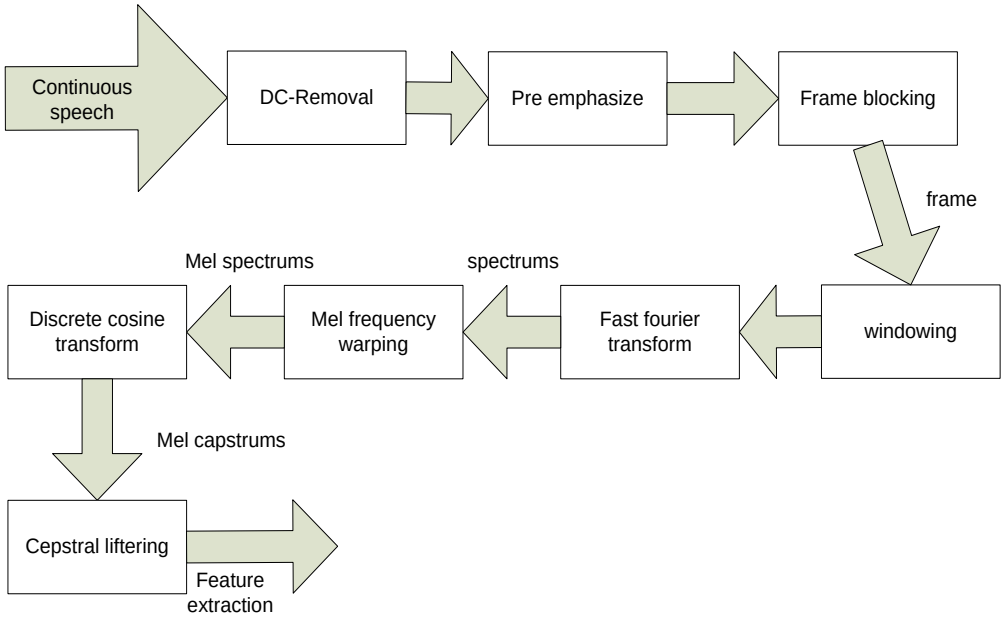
c. Mereplikasi organ pendengaran manusia dalam melakukan persepsi terhadap *signal* suara.



Gambar 2.2 Contoh *Signal* Suara

Pada Gambar 2.2 dapat dilihat contoh dari *signal* suara. Pengujian yang dilakukan untuk periode waktu yang cukup pendek (sekitar 10 sampai 30 milidetik) akan menunjukkan karakteristis *signal* suara yang *stationary.* Tetapi bila dilakukan dalam periode waktu yang lebih panjang karakteristik *signal* suara akan terus berubah sesuai dengan kata yang diucapkan.

MFCC *feature extraction* sebenarnya merupakan adaptasi dari sistem pendengaran manusia, dimana *signal* suara akan difilter secara linear untuk frekuensi rendah (dibawah 1000 Hz) dan secara logaritmik untuk frekuensi tinggi (diatas 1000 Hz). Gambar dibawah ini merupakan block diagram untuk MFCC [1].



Gambar 2.3 Blok Diagram Untuk MFCC

Pada Gambar 2.3 dapat dilihat proses MFFC, *Continuos speech* yaitu inputan suara dengan kata yang telah ditentukan oleh sistem, lalu proses *Dc Removal* untuk mendapatkan nilai normal dari sampel sinyal, lalu proses *Pre-emphasize* untuk memperbaiki signal dari gangguan *noise*, lalu proses *Frame Blocking* bertujuan untuk membagi sampel sinyal menjadi beberapa frame, lalu proses *Windowing* untuk mengurangi efek diskontinuitas pada ujung-ujung frame yang dihasilkan oleh proses frame blocking, lalu proses FFT sehingga diperoleh sampel sinyal dalam frekuensi domain, lalu proses *Filterbank* untuk mengetahui ukuran energi dari setiap frekuensi band, lalu proses DFT untuk mendapatkan *mel cepstrums*, langkah terakhir proses *Cepstral Liftering*, bertujuan menghaluskan *spectrum signal*.

2.7.1 DC Removal

Remove DC Components bertujuan untuk menghitung rata-rata dari data sampel suara, dan mengurangkan nilai setiap sampel suara dengan nilai rata-rata tersebut. Tujuannya adalah mendapat normalisasi dari data suara input [1].

$$D[n] = s[n] - \bar{x} \quad , 0 \leq n \leq N-1 \tag{2.1}$$

Dimana : D[n] = sampel signal hasil proses DC removal

s[n] = sampel signal asli

$\bar{x}$ = nilai rata-rata sampel signal asli.

N = panjang *signal*

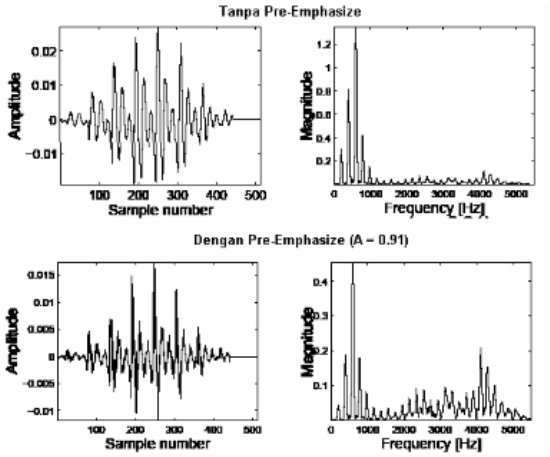
2.7.2 Pre Emphasize

Pre emphasize merupakan salah satu jenis *filter* yang sering digunakan sebelum sebuah *signal* diproses lebih lanjut. *Filter* ini mempertahankan frekuensi-frekuensi tinggi pada sebuah spektrum, yang umumnya tereliminasi pada saat proses produksi suara [1].

Tujuan dari *Pre – emphasize Filetering* ini adalah:

a. Mengurangi *noise ratio* pada *signal*, sehingga dapat meningkatkan kualitas *signal.*

b. Menyeimbangkan spektrum dari *voiced sound.* Pada saat memproduksi *voiced sound, glottis* manusia menghasilkan sekitar -12 dB *octave slope.* Namun ketika energy akustik tersebut dikeluarkan melalui bibir, terjadi peningkatan sebesar +6. Sehingga *signal* yang terekam oleh *microphone* adalah sekitar -6 dB *octave slope.* Dampak dari efek ini dapat dilihat pada Gambar 2.4 berikut:



Gambar 2.4 Contoh dari *pre-emphasize* pada sebuah *frame*

Pada Gambar 2.4, terdapat gambar tanpa *Pre-Emphasize* dan gambar dengan *Pre-Emphasize*, terlihat bahwa distribusi energi pada setiap frekuensi terlihat lebih seimbang setelah diimplementasikan *pre-emphasize filter*.

Bentuk yang paling umum digunakan dalam *pre-emphasize filter* adalah sebagai berikut :

$$H(z) = 1 - \alpha z^{-1} \qquad (2.2)$$

Dimana $0.9 \leq \alpha \leq 1.0$, dan α € R. Formula diatas dapat dijadikan sebagai *first order differentiator*, sebagai berikut:

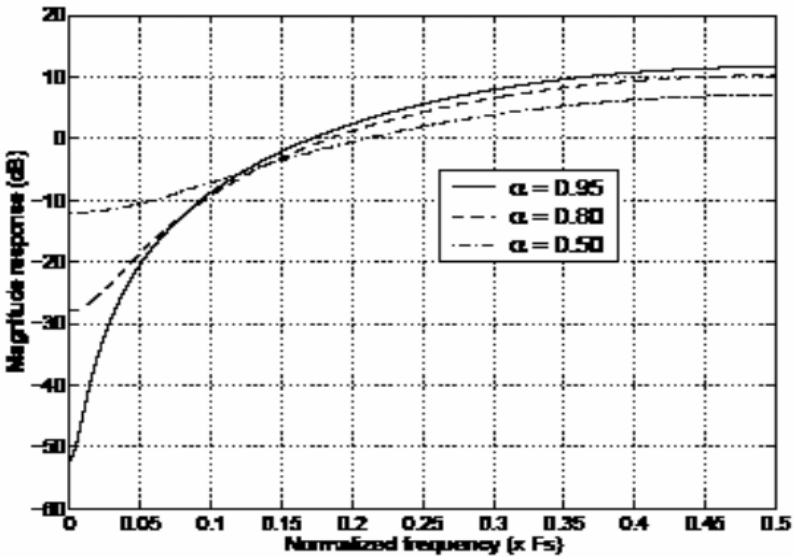
$$y[n] = s[n] - \alpha\, s[n-1] \qquad (2.3)$$

Dimana : y[n] = *signal* hasil *pre-emphasize filter*

s[n] = *signal* sebelum *pre-emphasize filter*

Pada umumnya nilai α yang paling sering digunakan adalah antara 0.9 sampai 1.0.

Magnitude response (dB scale) untuk nilai α yang berbeda dapat dilihat pada gambar 2.5 berikut:



Gambar 2.5 Magnitude *response* dari *pre-emphasize* filter

Pada gambar 2.5 dapat dilihat perbedaan magnitude *response* dengan menggunakan α yang berbeda, α yang digunakan pada gambar tersebut yaitu 0,50, 0,80 dan 0,95.

2.7.3 Frame Blocking

Karena sinyal suara terus mengalami perubahan akibat adanya pergeseran artikulasi dari organ produksi vokal, signal harus diproses secara *short segments* (*short frame*). Panjang *fram*e yang biasanya digunakan untuk pemrosesan signal adalah antara 10-30 milidetik. Panjang *fram*e yang digunakan sangat mempengaruhi keberhasilan dalam analisa spektral. Di satu sisi, ukuran dari frame harus sepanjang mungkin untuk dapat menunjukkan resolusi frekuensi yang baik. Tetapi di lain sisi, ukuran frame juga harus cukup pendek untuk dapat menunjukkan resolusi waktu yang baik [1].

Proses *frame* ini dilakukan terus sampai seluruh signal dapat diproses. Selain itu, proses ini umumnya dilakukan secara *overlapping* untuk setiap frame-nya. Panjang daerah *overlap* yang umum digunakan adalah kurang lebih 30% sampai 50% dari panjang *frame*. *Overlapping* dilakukan untuk menghindari hilangnya ciri atau karakteristik suara pada perbatasan perpotongan setiap *frame*.

Jumlah frame = ((I-N)/M)+1) (2.4)

I = Sample rate

N = Sample point (Sample rate * waktu framing(s))

M = N/2

2.7.4 Windowing

Proses *framing* dapat menyebabkan terjadinya kebocoran spektral (*spectral leakage*) atau *aliasing. Aliasing* adalah *signal* baru dimana memiliki frekuensi yang berbeda dengan *signal* aslinya. Efek ini dapat terjadi karena rendahnya jumlah *sampling rate,* ataupun karena proses *frame blocking* dimana menyebabkan *signal* menjadi *discontinue.* Untuk mengurangi kemungkinan terjadinya kebocoran spektral, maka hasil dari proses *framing* harus melewati proses *window* [1].

Sebuah fungsi *window* yang baik harus menyempit pada bagian *main lobe* dan melebar pada bagian *side lobe*-nya. Berikut ini adalah representasi dari fungsi *window* terhadap *signal* suara yang diinputkan.

$$x(n) = x_i(n)w(n) \quad (2.5)$$

n = 0,1,…,N-1

$x(n)$ = nilai sampel *signal* hasil *windowing*

$x_i(n)$ = nilai sampel dari *frame signal* ke i

$w(n)$ = fungsi *window*

N = *frame size*

Ada banyak fungsi *window,* namun yang paling sering digunakan dalam aplikasi *speaker recognition* adalah *hamming window.* Fungsi *window* ini menghasilkan *sidelobe level* yang tidak terlalu tinggi (kurang lebih -43 dB), selain itu *noise* yang dihasilkan pun tidak terlalu besar.

Fungsi Hamming *window* adalah sebagai berikut :

$$w(n) = 0.54 - 0.46\ cos\frac{2\pi n}{M-1} \quad (2.6)$$

n = 0,1,...,M-1

M = panjang *frame*

2.7.5 Fast Fourier Transform

DFT (*Discrete Fourier Transform*) merupakan perluasan dari transformasi *fourier* yang berlaku untuk *signal-signal* diskrit dengan panjang yang terhingga. Semua *signal* periodik terbentuk dari gabungan *signal-signal* sinusoidal yang menjadi satu yang dapat dirumuskan sebagai berikut :

$$F[k] = \sum_{n=0}^{N-1} f[n] e^{-\frac{j2\pi nk}{N}}, 0 \leq k \leq N-1 \quad (2.7)$$

N = jumlah sampel yang akan diproses ($N \in$ N)

f(n) = nilai sampel *signal*

K = variable frekuensi discrete, dimana akan bernilai ($k = {}^{N}/_{2}$, $k \in$ N)

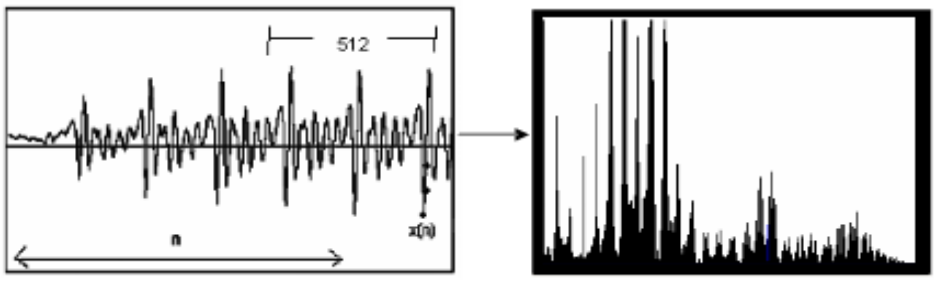
Bentuk yang digunakan dalam FFT adalah sebagai berikut

$$F(k) = \sum_{n=1}^{N} \left(f_{(n)} \cos \frac{2\pi nkT}{N} \right) - j \sum_{n=1}^{N} \left(f_{(n)} \sin \frac{2\pi nkT}{N} \right) \quad (2.8)$$

Untuk menghitung hasil FFT digunakan rumus

$$|f(u)| = [R^2 + I^2]^{\frac{1}{2}} \quad (2.9)$$

Dengan persamaan 2.9 suatu *signal* suara dalam domain waktu dapat kita cari frekuensi pembentuknya. Hal inilah tujuan penggunaan analisa *fourier* pada data suara, yaitu untuk merubah data dari domain waktu menjadi data spektrum di domain frekuensi. Untuk pemrosesan *signal* suara, hal ini sangatlah menguntungkan karena data pada domain frekuensi dapat diproses dengan lebih mudah dibandingkan data pada domain waktu, karena pada domain frekuensi, keras lemahnya suara tidak seberapa berpengaruh [1].



Gambar 2.6 Domain Waktu menjadi Domain Frekuensi

Pada Gambar 2.6 dapat dilihat bahwa gambar sebelah kiri menunjukan gambar domain waktu dan gambar sebelah kanan menunjukan gambar domain frekuensi.

Untuk mendapatkan spektrum dari sebuah *signal* dengan DFT diperlukan N buah sampel data berurutan pada domain waktu, yaitu x[m] sampai x[m+N-1]. Data tersebut dimasukkan dalam fungsi DFT maka akan menghasilkan N buah data. Namun karena hasil dari DFT adalah simetris, maka hanya N/2 data yang diambil sebagai spektrum.

2.7.6 Mel Frequency Warping

Mel Frequency Wrapping umumnya dilakukan dengan menggunakan *Filterbank. Filterbank* adalah salah satu bentuk dari *filter* yang dilakukan dengan tujuan untuk mengetahui ukuran energi dari *frequency band* tertentu dalam *signal* suara. *Filterbank* dapat diterapkan baik pada domain waktu maupun pada domain frekuensi, tetapi untuk keperluan MFCC, *filterbank* harus diterapkan dalam domain frekuensi [1].

Filterbank menggunakan representasi konvolusi dalam melakukan *filter* terhadap *signal.* Konvolusi dapat dilakukan dengan melakukan multiplikasi antara spektrum *signal* dengan koefisien *filterbank.* Berikut ini adalah rumus yang digunakan dalam perhitungan *filterbanks.*

$$Y[i] = \sum_{j=1}^{N} S[j]H_i[j] \tag{2.10}$$

N = jumlah *magnitude spectrum*

$S[j]$ = *magnitude spectrum* pada frekuensi j

$H_i[j]$ = koefisien *filterbank* pada frekuensi j ($1 \leq i \leq M$)

M = jumlah *channel* dalam *filterbank*

Filterbank dihitung menggunakan rumus sebagai berikut :

$$Hi = \frac{mel\ f}{Xi/_2} \tag{2.11}$$

Rumus fungsi Mel Scale sebagai berikut :

$$Mel\ f = 2595 \times log10\left(1 + \frac{f}{700}\right) \quad (2.12)$$

Mel f = Fungsi Mel Scale

f = frekuensi

Persepsi manusia terhadap frekuensi dari *signal* suara tidak mengikuti *linear scale.* Frekuensi yang sebenarnya (dalam Hz) dalam sebuah *signal* akan diukur manusia secara subyektif dengan menggunakan *mel scale. Mel frequency scale* adalah *linear* frekuensi *scale* pada frekuensi dibawah 1000 Hz, dan merupakan *logarithmic scale* pada frekuensi diatas 1000 Hz.

2.7.7 Discrete Cosine Transform

DCT merupakan langkah terakhir dari proses utama MFCC *feature extraction.* Konsep dasar dari DCT adalah mendekorelasikan *mel spectrum* sehingga menghasilkan representasi yang baik dari property spektral local. Pada dasarnya konsep dari DCT sama dengan *inverse fourier transform.* Namun hasil dari DCT mendekati PCA (*principle component analysis*). PCA adalah metode static klasik yang digunakan secara luas dalam analisa data dan kompresi. Hal inilah yang menyebabkan seringkali DCT menggantikan *inverse fourier transform* dalam proses MFCC *feature extraction* [1].

Berikut adalah formula yang digunakan untuk menghitung DCT.

$$Cn = \sum_{k=1}^{K} (\text{logSk}) \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right] \quad (2.13)$$

n = 1,2,…, K

S_k = keluaran dari proses *filterbank* pada *index* k

K = jumlah koefisien yang diharapkan

Koefisien ke nol dari DCT pada umumya akan dihilangkan, walaupun sebenarnya mengindikasikan energi dari frame signal tersebut. Hal ini dilakukan karena, berdasarkan penelitian-penelitian yang pernah dilakukan, koefisien ke nol ini tidak reliable terhadap *speaker recognition*

2.7.8 Cepstral Liftering

Hasil dari proses utama MFCC *feature extraction* memiliki beberapa kelemahan. *Low order* dari *cepstral coefficients* sangat sensitif terhadap *spectral slope,* sedangkan bagian *high order*nya sangat sensitif terhadap *noise.* Oleh karena itu, *cepstral liftering* menjadi salah satu standar teknik yang diterapkan untuk meminimalisasi sensitifitas tersebut. *Cepstral liftering* dapat dilakukan dengan mengimplementasikan fungsi *window* terhadap *cepsral features.*

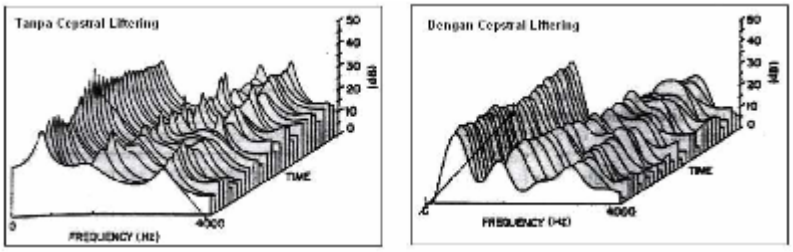
$$w[n] = \left\{1 + \frac{L}{2}\sin\left(\frac{n\pi}{L}\right)\right\} \qquad (2.14)$$

L = jumlah *cepstral coefficients*

n = *index* dari *cepstral coefficients*

Cepstral liftering menghaluskan spektrum hasil dari *main processor* sehingga dapat digunakan lebih baik untuk *pattern recognition.*

Cepstral liftering menghaluskan spektrum hasil dari *main processor* sehingga dapat digunakan lebih baik untuk *pattern recognition.* Gambar 2.7 menunjukkan perbandingan spektrum dengan dan tanpa *cepstral liftering*.



Gambar 2.7 Perbandingan Spectrum

Pada gambar 2.7, gambar sebelah kiri menunjukan gambar tanpa cepstral liftering dan gambar sebelah kanan menunjukan gambar dengan cepstral liftering, dapat dilihat bahwa pola spectrum setelah dilakukan *cepstral liftering* lebih halus daripada spektrum yang tidak melalui tahap spectrum *liftering.*

2.8 K-Means Clustering

Clustering merupakan faktor yang paling fundamental dalam *pattern recognition*. Masalah utama dari *clustering* adalah mendapatkan beberapa nilai vektor pusat yang dapat mewakili keseluruhan vektor dari hasil *feature extraction*. K-means *clustering* adalah salah satu metode yang digunakan untuk mempartisi vektor hasil *feature extraction* ke dalam k vektor pusat [2].

K-Means Clustering adalah proses memetakan vektor-vektor yang berada pada lingkup wilayah yang luas besar menjadi sejumlah tertentu (k) vektor. Wilayah yang terwakili oleh vektor pusat hasil dari proses kuantisasi disebut sebagai *cluster*. Sebuah vektor pusat hasil dari proses kuantisasi dikenal sebagai *codewords*. Sedangkan kumpulan dari vektor pusat dikenal *sebagai codebooks* [2].

Keuntungan dari diimplementasikannya *K-Means Clustering* dalam merepresentasikan *speech spectral vectors* adalah :

1. Mengurangi *storage memory* yang digunakan untuk analisis informasi spektral.
2. Mengurangi perhitungan yang digunakan untuk menentukan kemiripan dari vektor spektral.

Kelemahan dari penggunaan *K-Means Clustering codebooks* dalam merepresentasikan speech spectral vectors adalah :

1. Timbulnya spektral distorsi. Hal ini terjadi karena vektor yang dianalisa bukanlah vektor asli, tetapi sudah mengalami proses kuantisasi
2. Storage yang digunakan untuk menyimpan *codebooks* vektor sering kali menjadi masalah. Untuk jumlah *codebooks* yang besar, membutuhkan *storage* yang cukup besar juga.

Algoritma K-Means adalah seperti berikut [14] :

1. Tentukan k (jumlah *cluster*) yang ingin dibentuk.
2. Bangkitkan k *centroid* (titik pusat cluster) awal secara *random*.
3. Untuk setiap *record* , temukan pusat *cluster* terdekat.
4. Untuk setiap k *cluster* , temukan pusat *cluster*, dan *update* lokasi dari setiap pusat *cluster* dengan nilai *centroid* yang baru. Pusat *cluster* diperoleh dengan cara menghitung nilai rata-rata dari data-data yang berada pada *cluster* yang sama.

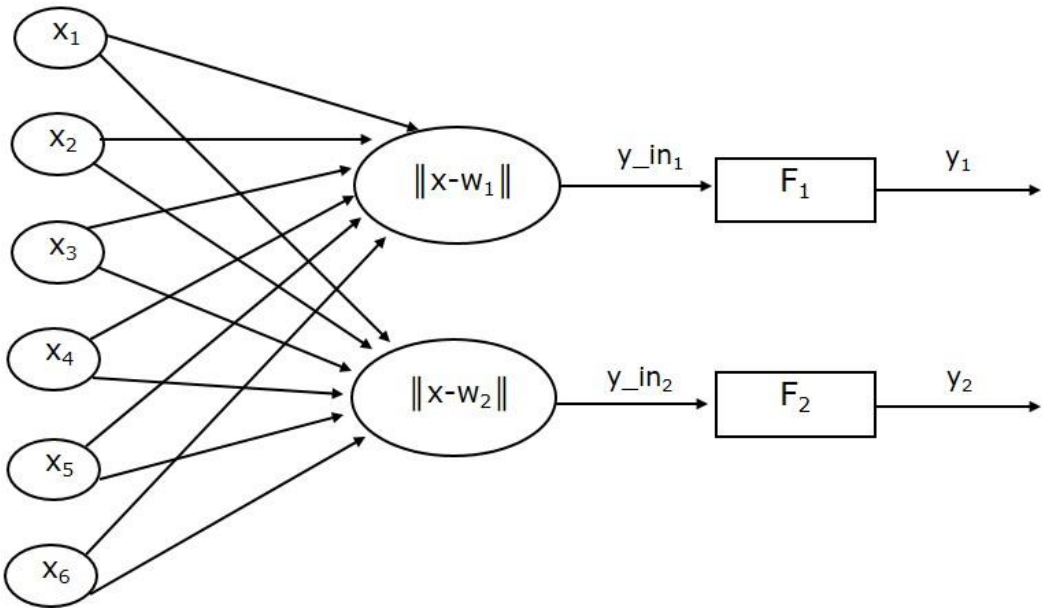
5. Kembali ke langkah 3 – 5 sampai konvergen.

2.9 Learning Vector Quantization

Learning Vector Quantization (LVQ) adalah suatu metode untuk melakukan pembelajaran atau pelatihan pada lapisan kompetitif yang terawasi. LVQ belajar mengklasifikasikan vektor masukan ke kelas target yang ditentukan oleh pengguna [3].

Jaringan LVQ terdiri dari dua lapisan tersembunyi yaitu lapisan kompetitif dan lapisan linear. Lapisan kompetitif disebut juga *Self Organizing Map* (SOM). Disebut lapisan kompetitif karena neuron-neuron berkompetisi dengan algoritma kompetisi yang akan menghasilkan neuron pemenang (*winning neuron*). Pada jaringan LVQ, bias pada lapisan kompetitif dihilangkan. Lapis linear mengalih ragamkan kelas-kelas pada lapisan kompetitif ke klasifikasi target yang ditentukan oleh pengguna [3].

Arsitektur LVQ dapat dilihat pada Gambar 2.8 berikut :



Gambar 2.8 Arsitektur LVQ

Algoritma LVQ adalah sebagai berikut :

0. Tetapkan:

a. Bobot awal variabel *input* ke-j menuju ke kelas (*cluster*) ke-i: Wij, dengan i=1,2,…, K; dan j=1,2,…, m.

b. *Maksimum* epoh: MaxEpoh.

c. Parameter *learning rate*: α.

d. Pengurangan *learning rate*: Decα.

1. Masukkan:

Data input: Xij; dengan i=1,2,…,n; dan j=1,2,…,m.

Target berupa kelas: Tk;dengan k=1,2,…,n.

2. Tetapkan kondisi awal: epoh=0;

3. Kerjakan jika: (epoh ≤ MaxEpoh)

a. epoh = epoh+1;

b. Kerjakan untuk i=1 sampai n

i. Tentukan J sedemikian hingga ‖Xi-Wj‖ minimum; dengan j=1,2,…,K.

ii. Perbaiki Wj dengan ketentuan:

- Jika T = Cj maka:

Wj = Wj + α (Xi - Wj) (21)

- Jika T ≠ Cj maka:

Wj = Wj - α (Xi - Wj) (22)

c. Kurangi nilai α.(pengurangan α bisa dilakukan dengan: α = α - Decα; atau dengan cara: α =α -α*Decα)

2.10 Euclidean Distance

Euclidean distance adalah sebuah metode yang digunakan untuk mengukur jarak (*distance*). *Euclidean distance* sebenarnya merupakan generalisasi dari teorema phytagoras. *Euclidean distance* diformulasikan pada persamaan berikut:

$$d(x,y) = \sqrt{\sum_{i=1}^{r}(x_i - y_i)^2} \quad (2.15)$$

d = jarak antara x dan y

r = jumlah fitur dalam vektor

x_i, y_i = nilai fitur ke i dari x dan y

Jarak minimal yang mungkin antara dua vektor data adalah 0. Selain itu, jarak dari *x* ke *y* akan sama dengan *y* ke *x*, $d(x, y) = d(y, x)$ [14].

2.11 K-Fold Cross Validation

Cross Validation adalah metode umum digunakan untuk mengevaluasi kinerja *classifier*. Dalam pendekatan *cross validation*, setiap *record* digunakan beberapa kali dalam jumlah yang sama untuk *training* dan tepat sekali untuk *testing*. Metode ini mempartisi data ke dalam dua sub set data yang berukuran sama. Pilih salah satu sebagai data *training* dan satu lagi untuk *testing*, kemudian dilakukan pertukaran fungsi dari subset sedemikian sehingga subset yang sebelumnya sebagai *training set* menjadi *test set* demikian sebelumnya. Pendekatan ini dinamakan *two-fold- cross-validation.* Total *error* diperoleh dengan menjumlahkan *error-error* untuk kedua proses tersebut. Metode *k-fold cross-validation* menggeneralisasi pendekatan ini dengan mensegmentasi data ke dalam k partisi berukuran sama. Selama proses, salah satu dari partisi dipilih untuk *training*, sedangkan sisanya untuk *testing*. Prosedur ini diulangi k kali sedemikian sehingga setiap partisi digunakan untuk *testing* tepat satu kali. Total *error* ditentukan dengan menjumlahkan *error* untuk semua k proses tersebut [6].

2.12 Confussion Matrix

Untuk mengevaluasi performa dari hasil proses klasifikasi dapat ditampilkan pada *confussion matrix*. *Confussion Matrix* berisi tentang klasifikasi aktual dan yang telah diprediksi yang dilakukan oleh sebuah sistem klasifikasi. Kinerja sebuah sistem klasifikasi umumnya dievaluasi dengan menggunakan data dalam matriks.Tabel berikut menunjukan contoh *confussion matrix* untuk klasifikasi masalah biner dua kelas [12].

Tabel 2.1 *Confussion Matrix*

		Predicted Class	
		Kelas = 1	**Kelas = 0**
Actual Class	**Kelas = 1**	*f11*	*f10*
	Kelas = 0	*f01*	*f00*

Berdasarkan isi matrik confusion, maka dapat diketahui jumlah data dari masing-masing kelas yang diprediksi secara benar yaitu (*f*11+*f*00) dan data yang

diklasifikasikan salah yaitu (*f*10+*f*01). Untuk menghitung akurasi digunakan formula sebagai berikut:

$$Akurasi = \frac{f11+f00}{f11+f10+f01+f00} \quad (2.16)$$

2.13 Pemograman Berorientasi Objek

Pemograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Menggunakan pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpuln objek yang berkorespondensi dengan objek dunia nyata.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemograman, dan pengujian perangkat lunak. Berikut adaah beberapa konsep dasar yang harus dipahami pemograman berorientasi objek [13] :

1. Kelas (*Class*)

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*Object)*

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, strutur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (*Method*)

Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada terstruktur. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri.

4. Atribut (*Attribute*)

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek

30 dalam kelas objek. Atribut secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*Abstraction*)

Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapulasi (*Encapsulation*)

Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja.

7. Pewarisan (*Inheritance*)

Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (*Interface*)

Antarmuka atau interface sangat mirip dengan kelas, tetapi tanpa atribut kelas dan tanpa memiliki metode yang dideklarasikan. Antarmuka biasanya digunakan agar kelas lain tidak langsung mengakses ke suatu kelas.

9. *Reusability*

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi antar objek

Komunikasi antar-objek dilakukan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.

12. Polimorfisme (*Polymorphism*)

Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. *Package*

Package adalah sebuah kontainer atau kemasan yang daoat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda.

2.14 C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain dengan beberapa penyederhanaan [10].

2.15 Unified Modeling Language

UML singkatan dari *Unifed Modeling Language* yang berarti bahasa pemodelan standar. UML merupakan bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara berorientasi objek. Ada beberapa diagram yang digunakan proses pembuatan perangkat lunak berorientasi objek diantaranya, *use case diagram, activity diagram, class diagram* dan *sequence diagram*[13].

2.15.1 Use Case Diagram

Use case diagram merupakan pemodelan untuk tingkah laku (*behavior*) pada sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use case diagram* digunakan

untuk mengetahui fungsi apa saja yang terdapat pada sistem. Terdapat dua hal utama yang diperlukan dalam pembentukan suatu *use case diagram* yaitu aktor dan *use case*.

1. Aktor merupakan orang, benda maupun sistem lain yang berinteraksi dengan sistem yang akan dibangun.
2. *Use Case* merupakan fungsionalitas atau layanan yang disediakan oleh sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.15.2 Class Diagram

Class diagram menggambarkan interaksi dan relasi antar kelas yang ada di dalam suatu sistem. Kelas memiliki atribut dan metode. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut:

1. *Private,* tidak dapat dipanggil dari luar kelas yang bersangkutan.
2. *Protected,* hanya dapat dipanggil oleh kelas yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public,* dapat dipanggil oleh siapa saja.

Class diagram menggambarkan relasi atau hubungan antar kelas dari sebuah sistem. Berikut ini beberapa gambaran relasi yang ada dalam *class diagram*:

1. *Association*

   Hubungan antar class yang statis. *Class* yang mempunyai relasi asosiasi menggunakan *class* lain sebagai atribut pada dirinya.
2. *Aggregation*

   Relasi yang membuat *class* yang saling terikat satu sama lain namun tidak terlalu berkegantungan.
3. *Composition*

   Relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan.

4. *Dependency*

   Hubungan antar *class* dimana *class* yang memiliki relasi *dependency* menggunakan *class* lain sebagai atribut pada *method*.

5. *Realization*

   Hubungan antar *class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya.

2.15.3 Sequence Diagram

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya *sequence diagram* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *sequence diagram* sehingga semakin banyak *use case* yang didefinisikan maka *sequence diagram* yang harus dibuat juga semakin banyak. Penomoran pesan berdasarkan urutan iteraksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

2.15.4 Activity Diagram

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem, proses bisnis atau menu yang ada pada perangkat lunak. Setiap *use case* yang telah dibentuk digambarkan aktivitasnya dalam *activity diagram*, mulai dari peran aktor, peran sistem, dan *decision*. *Activity diagram* juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem.
2. Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan tampilan antarmuka.

3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

2.16 Tools Pembangunan Perangkat Lunak

Adapun beberapa perangkat lunak pendukung dalam pembuatan aplikasi pengenalan suara ini diantaranya:

2.16.1 Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi *console*, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, Integrated Development Environment (IDE), dan dokumentasi yang umumnya berupa MSDN Library. Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam native code (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun managed code (dalam bentuk Microsoft Intermediate Language di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework) [10].

2.16.2 Basis Data

Basis Data dapat didefinisikan dalam sejumlah sudut pandang seperti himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah. Basis Data dapat diartikan pula sebagai kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan [14].

Operasi dalam basis data:

1. Pembuatan basis data baru (*create database*).
2. Penghapusan basis data (*drop database*).
3. Pembuatan file/tabel baru ke suatu basis data (*create table*).
4. Penghapusan file/tabel dari suatu basis data (*drop table*).
5. Penambahan/pengisian data baru ke sebuah file/tabel di sebuah basis data (*insert*).
6. Pengambilan data dari sebuah file/tabel (*retrieve/search*).
7. Pengubahan data dari sebuah file/tabel (*update*).
8. Penghapusan data dari sebuah file/tabel (*delete*).

Operasi yang berkenaan dengan pembuatan objek (basis data dan tabel) merupakan operasi awal yang hanya dilakukan sekali dan berlaku seterusnya. Sedang operasi-operasi yang berkaitan dengan isi tabel (data) merupakan operasi rutin yang akan berlangsung berulang-ulang dan karena itu operasi-operasi inilah yang lebih tepat mewakili aktifitas pengelolaan dan pengolahan dalam basis data [14].

Database Management System (DBMS) atau dalam bahasa Indonesia sering disebut sebagai Sistem Manajemen Basis Data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Suatu sistem aplikasi disebut DBMS jika memenuhi persyaratan minimal sebagai berikut:

1. Menyediakan fasilitas untuk mengelola akses data.
2. Mampu menangani integritas data.
3. Mampu menangani akses data yang dilakukan secara bersamaan.
4. Mampu menangani *backup* data.

Database MySQL merupakan sistem manajemen basis data SQL yang sangat terkenal dan bersifat *open source*. MySQL dibangun, didistribusikan, dan didukung oleh MySQL AB. MySQL AB merupakan perusahaan komersial yang dibiayai oleh pengembang MySQL. MySQL mempunyai dua macam lisensi yaitu lisensi yang bersifat *open source* dengan menggunakan GNU (*General Public License*) dan lisensi kedua berupa *Standard Commercial License* yang dapat dibeli dari MySQL AB.

Dalam konteks bahasa SQL, informasi disimpan dalam tabel-tabel yang secara logis merupakan struktur dua dimensi yang tersimpan atas baris-baris data (*row* atau *record*) yang berada dalam satu atau lebih kolom (*column*). Baris pada tabel disebut *insance* dari data sedangkan kolom sering disebut sebagai *atributes* atau *field*. MySQL yang berjalan di sistem operasi *Windows* terbukti sangat stabil. MySQL versi *Windows* mempunyai fitur yang sama dengan MySQL versi *Unix* [14].