Buku Ajar

Teknik Pengolahan Suara Digital

oleh :

Achmad Hidayatno, S.T., M.T.

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS DIPONEGORO
SEMARANG

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamualaikum Wr. Wb

Segala puja dan puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik, serta hidayah-Nya kepada penulis sehingga penulis dapat menyelesaikan penyusunan buku ajar dalam rangka Hibah Pendidikan Proyek DUE LIKE BATCH III untuk Mata Kuliah Teknik Pengolahan Suara Digital.

Dalam penyelesaian buku ajar ini, penulis banyak mendapatkan bantuan dari berbagai pihak, oleh karena itu pada kesempatan kali ini penulis ingin menyampaikan ucapan terima kasih kepada beberapa pihak sebagai berikut.

1. Direktur Eksekutif Proyek DUE LIKE BATCH III UNDIP, beserta seluruh staf LPIU yang ikut memonitor, menyukseskan dan terus memacu kegiatan
2. Ir. Hj. Sri Eko Wahyuni, MS, dan segenap Civitas Akademika Fakultas Teknik UNDIP
3. Segenap Civitas Akademika Jurusan Teknik Elektro FT UNDIP.
4. Semua pihak-pihak yang telah banyak membantu penulis yang tidak dapat disebutkan satu per satu disini.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, baik dari segi isi maupun cara penyusunannya. Oleh karena itu, penulis sangat mengharapkan saran dan kritik yang bersifat membangun dari pembaca. Penulis berharap, semoga hasil kecil ini dapat memberikan manfaat kepada para pembaca pada umumnya dan bagi penulis pada khususnya.

Wassalamualaikum Wr. Wb.

Semarang, Desember 2007

Penulis

DAFTAR ISI

Bab I
Pendahuluan

Buku ini dimaksudkan untuk menunjukkan penerapan beberapa teknik pengolahan suara digital. Untuk itu pembaca diharapkan mempunyai latarbelakang pengetahuan di bidang pengolahan sinyal digital secara umum. Pada bab pendahuluan ini akan dibahas beberapa masalah dasar seperti bentuk sinyal suara, operasi dasar pengolahan sinyal digital yang mungkin akan diterapkan pada sinyal suara, dan beberapa penerapan penting yang saat ini sudah dikembangkan atau masih dalam tahap penelitian.

Teknik pengolahan suara digital saat ini telah banyak digunakan untuk keperluan sehari-hari, seperti penerapan teknik pemampatan suara dalam bentuk file .mp3 yang memungkinkan pengguna teknologi ini menyimpan lebih banyak lagu dalam suatu media penyimpan seperti CD. Penerapan lainnya adalah pada piranti telepon genggam yang memungkinkan seorang pengguna untuk mengawali sambungan telepon hanya dengan mengucapkan suatu kata kunci tanpa harus repot untuk menekan tombol. Luasnya bidang penerapan pengolahan suara digital menjadikan bidang ini lahan penelitian yang sangat menarik.

Sinyal Suara (voice) merujuk pada segala energi akustik yang dikeluarkan oleh organ pembicaraan manusia. Istilah Bunyi (sound) adalah segala sesuatu yang dapat didengar oleh indera pendengaran manusia, sedangkan ucapan (speech) adalah suara yang mengandung makna untuk tujuan komunikasi. Tujuan bersuara adalah untuk berkomunikasi. Sehubungan dengan teori informasi ucapan dapat diwakili dalam bentuk isi pesan atau informasi atau sinyal yang membawa informasi, yaitu dapat dinyatakan sebagai gelombang akustik.

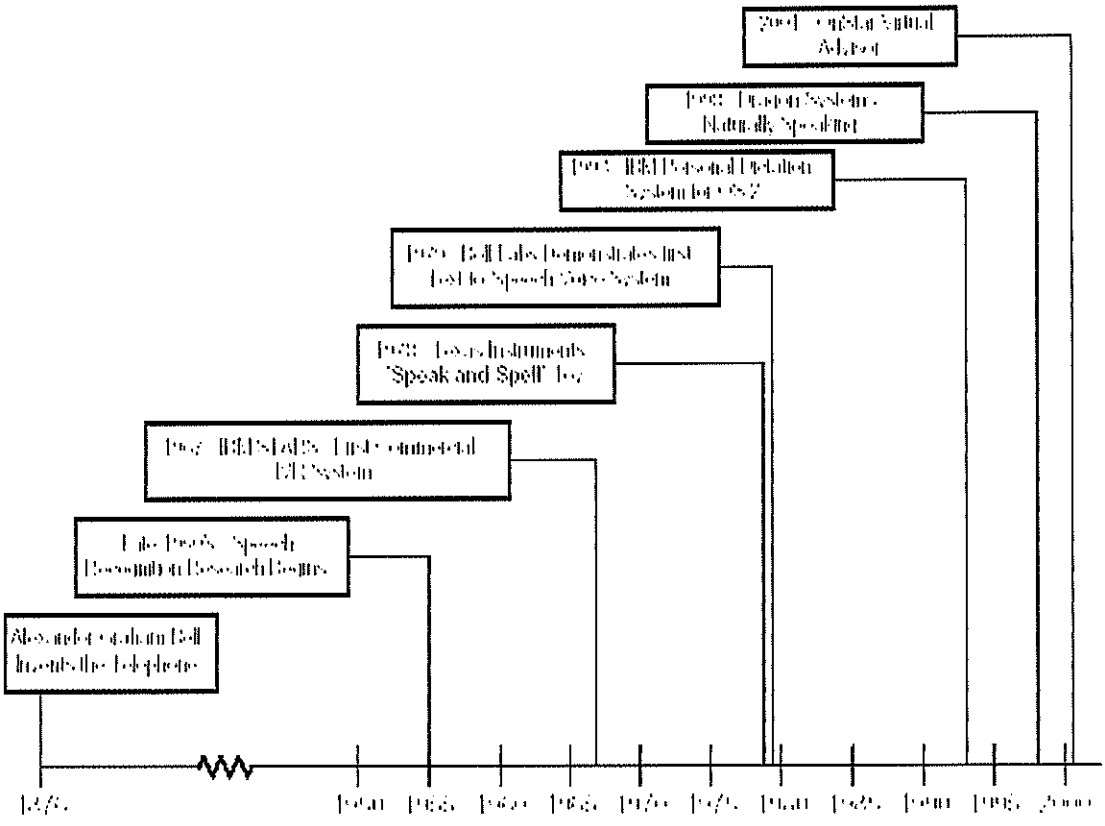
Beberapa penerapan pengolahan suara digital

1. Penyimpanan dan penyaluran digital atas sinyal suara
2. Sistem pembangkit ucapan
3. Sistem identifikasi dan verifikasi pengucap
   Sistem ini akan mengenali siapa yang mengucapkan kata tersebut, atau memastikan bahwa suara yang ada milik seorang penutur.
4. Sistem pengenal ucapan
   Sistem ini akan mengenali apa yang diucapkan oleh seorang penutur.
5. Bantuan untuk penyandang cacat
6. Peningkatan kualitas sinyal

1.1 Pengenalan Ucapan (Speech Recognition)

Pemrosesan sinyal suara manusia dengan komputer sekarang telah banyak dilakukan untuk berbagai macam tujuan diantaranya untuk membuat suatu sistem komputer yang dapat diperintah melalui sinyal ucapan. Dengan adanya sistem pengenalan ucapan ini dapat dibuat suatu sistem komputer interaktif yang secara langsung dapat merespon ucapan seorang penutur. Ucapan tersebut dapat ditampilkan dalam tulisan atau juga dapat merintahkan komputer untuk melakukan suatu aksi yang dapat diterapkan untuk tujuan mengendalikan sesuatu atau juga untuk tujuan keamanan. Luasnya cakupan penerapan pemrosesan sinyal

suara ini menuntut adanya penelitian yang berkesinambungan untuk membuat sistem komputer yang dapat berinteraksi secara nyata dengan isyarat tutur.



Gambar 1.1 Perkembangan Hasil Penelitian ASR secara global

Penelitian di bidang pengenalan ucapan manusia telah dikembangkan sejak awal abad 20 dengan diluncurkannya produk komersial Radio Rex yang dibuat tahun 1920-an [1]. Sampai saat ini, dengan pendanaan yang sangat besar telah banyak dikembangkan produk-produk pengenalan ucapan termasuk program pendikte otomatis yang ditujukan untuk para penutur dengan bahasa di negara pengembangnya, namun belum ada satupun produk pendikte untuk Bahasa Indonesia. Perkembangan penelitian dan produk komersial di bidang pengenalan ucapan otomatis dapat dilihat pada Gambar 1.

Merujuk pada begitu panjangnya penelitian mengenai ucapan manusia ini tentunya memperlihatkan betapa pentingnya pengetahuan mengenai teknologi ini untuk bekal menghadapi kecenderungan perkembangan teknologi pada umumnya yang begitu cepat. Sampai saat ini belum dicapai suatu metode dan teknik yang benar-benar tegar (*robust*) yang mampu mengenali ucapan untuk semua kondisi lingkungan yang mungkin. Hal ini disebabkan setiap bahasa menghasilkan sinyal ucapan yang mempunyai karakteristik khas, sehingga metode dan teknik yang telah berhasil diterapkan pada suatu bahasa penutur, belum tentu menghasilkan kinerja yang sama baiknya jika diterapkan untuk penutur dengan bahasa yang lain. Sehingga untuk Bahasa Indonesia perlu dilakukan penelitian yang intensif agar

diperoleh pengetahuan yang memadai untuk bisa menentukan metode dan teknik yang tepat.

Mengingat luasnya penggunaan pengenal ucapan otomatis, maka diperlukan suatu upaya pengembangan dan pembuatan pendikte ucapan sebagai salah satu aplikasi pengenal ucapan otomatis khususnya untuk penutur Bahasa Indonesia. Untuk itu proposal penelitian ini merupakan salah satu upaya untuk mewujudkan sebuah pendikte otomatis untuk penutur Bahasa Indonesia.

Kontribusi lainnya adalah dimulainya penelitian mengenai karakteristik sinyal ucapan penutur Bahasa Indonesia, sehingga diharapkan hasilnya akan menjadi acuan penelitian kelanjutannya yang memerlukan informasi mengenai sifat-sifat sinyal yang harus disesuaikan dengan metode yang akan dipilih. Beberapa karakteristik tersebut antara lain ditunjukkan oleh besaran formant, pitch, dan segmentasi yang tepat untuk ucapan Bahasa Indonesia.

Pengembangan pengenal ucapan otomatis mempunyai potensi sebagai industri jutaan dollar di masa yang akan datang [1,2] seperti yang telah ditunjukkan oleh penerapan sederhana atas teknologi ini dalam sistem telepon. Pengenal ucapan berpeluang sangat berguna, misalnya:

- Aplikasi telepon: untuk sistem voice-mail yang umum saat ini, seseorang dituntun untuk menekan sederetan tombol sebagai pengarah menuju aplikasi yang diinginkannya. Pengenal ucapan akan mehilangkan kerumitan itu.
- Operasi tangan-bebas: banyak situasi yang memaksa tangan tidak bisa digunakan untuk menjalankan piranti, untuk itu pengenal ucapan akan menolong seseorang tetap menjalankan aktivitasnya saat dia ingin melakukan kegiatan lainnya.
- Bantuan untuk penyandang cacat fisik: pengenal ucapan adalah antarmuka alternatif alami antara komputer dengan orang yang gerakan tangan dan lengannya terbatas atau gangguan penglihatan.

Sinyal ucapan merupakan hasil pengolahan organ pembicaraan manusia yang memiliki banyak peubah antara lain: warna suara (timbre), kata, intonasi atau irama bicara, emosi dan logat. Bagian organ penghasil suara manusia dapat dilihat pada Gambar 2. Banyaknya peubah yang terkandung dalam sinyal ucapan membuat penelitian di bidang pengenalan ucapan otomatis memiliki tingkat kesulitan yang relatif tinggi, sehingga memerlukan waktu dan kemampuan peneliti yang sesuai. Untuk itu usul penelitian ini dibuat sebagai wujud keinginan yang tinggi dari peneliti untuk terlibat secara aktif dalam bidang pengenalan ucapan otomatis khususnya untuk penutur Bahasa Indonesia.

Saat ini telah dikembangkan pengenal vokal Bahasa Indonesia yang mampu mengenali ucapan vokal dari beberapa penutur (pria dan wanita) dengan pencirian menggunakan transformasi Fourier [3], transformasi wavelet [4] dan penyandian ramalan linear (LPC) [5,6].

Penelitian mengenai pencirian yang tepat untuk ucapan Bahasa Indonesia masih terus dikembangkan dan yang sedang kami kerjakan adalah pencirian MFCC (*mel-frequency cepstral coefficient*). Selain itu telah dikembangkan pula pengenal ucapan kata terisolasi misalnya program pengenal kata angka dan kata yang digunakan untuk operasi komputer. Sedangkan untuk aplikasinya telah dikembangkan suatu antarmuka lisan untuk menjalankan komputer, seperti menjalankan program, membuka berkas, dan mengaktifkan piranti lain. Saat ini

tengah dikaji juga perancangan sistem aktivasi piranti berdasar pengenalan ucapan, yaitu suatu sistem untuk menghidupkan atau mematikan piranti elektronik dengan menggunakan suara manusia.

Pendekatan proses tak-stasioner dan sistem taklinear atas sinyal ucapan merupakan upaya untuk lebih memperbaiki unjukkerja sistem. Sinyal ucapan secara alami adalah proses taklinear dan tak-stasioner. Selama ini pendekatan stasioner dapat dilakukan dengan cara menganalisis sinyal ucapan per ruas waktu (pada umumnya sinyal ucapan disegmentasi masing-masing sekitar 30 milidetik). Berbagai usulan untuk pendekatan yang baru ini juga telah diajukan [7,8,9].

1.2 Pengubahan Tulisan-ke-Ucapan (Text-to-Speech)

Penelitian di bidang TTS untuk Bahasa Indonesia sampai saat ini baru satu yaitu IndoTTS dari Arry Arman Akhmad, dengan sumber suara adalah suara pembuatnya yang direkam di Belgia (id1) dengan basis mesin pengubah MBROLA. Untuk itu penelitian ini diajukan untuk melengkapi dan memperkaya pengungkapan TTS untuk bahasa sendiri.

Penelitian ini harus dilakukan karena meskipun telah dikembangkan mesin pengubah seperti MBROLA, TTS untuk Bahasa Indonesia belum bisa dijalankan kalau belum ada basis data suaranya. Data itu bisa berupa rekaman berdasar jenis kelamin, logat, ataupun emosi penutur.

Kontribusi utama dari penelitian ini adalah peluang dilakukannya penelitian lanjutan mengingat luasnya aplikasi yang dapat dikembangkan berdasar pada TTS. Salah satu contohnya adalah dimungkinkan dengan TTS, penyandang tuna netra masih bisa mendengarkan email yang masuk di komputernya, tentunya dengan prosedur tertentu. Namun jika teknologi TTS digabung dengan ASR (automatic speech recognition) atau program pengenal ucapan telah dikembangkan untuk penutur Bahasa Indonesia, maka dimungkinkan seorang penyandang tuna netra bisa berkomunikasi dengan komputer secara lisan. Artinya, untuk menjalankan program komputer cukup diperintah dengan suara si pengguna, sedangkan hasil pengolahan program tersebut dapat didengar olehnya.

Sistem *Text to Speech* pada prinsipnya terdiri dari dua sub sistem, yaitu :
1) bagian Konverter Teks ke Fonem *(Text to Phoneme)*, serta
2) bagian Konverter Fonem to Ucapan *(Phoneme to Speech)*.

Gambar 1.2 Diagram Blok Pengubah Tulisan menjadi Ucapan

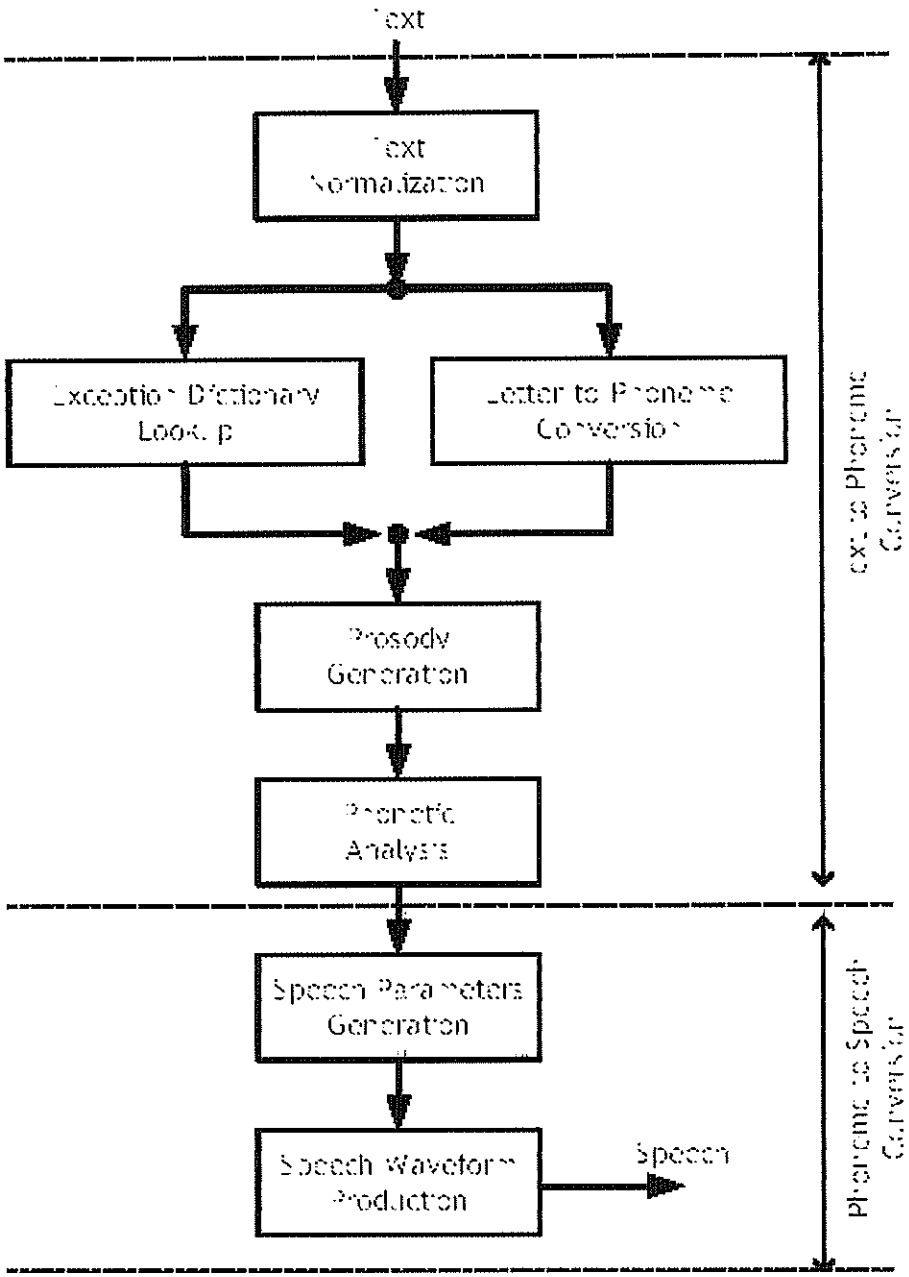
Bagian Konverter Teks ke Fonem berfungsi untuk mengubah kalimat masukan dalam suatu bahasa tertentu yang berbentuk teks menjadi rangkaian kode-kode bunyi yang biasanya direpresentasikan dengan kode fonem, durasi serta *pitch*-nya. Bagian ini bersifat sangat *language dependant*. Untuk suatu bahasa baru, bagian ini harus dikembangkan secara lengkap khusus untuk bahasa tersebut.

Bagian Konverter Fonem ke Ucapan akan menerima masukan berupa kode-kode fonem serta pitch dan durasi yang dihasilkan oleh bagian sebelumnya. Berdasarkan kode-kode tersebut, bagian Konverter Fonem ke Ucapan akan menghasilkan bunyi atau sinyal ucapan yang sesuai dengan kalimat yang ingin diucapkan. Ada beberapa alternatif teknik yang dapat digunakan untuk implementasi bagian ini. Dua teknik yang banyak digunakan adalah *formant synthesize*r, serta *diphone concatenatio*n. *Formant synthesizer* bekerja berdasarkan suatu model matematis yang akan melakukan komputasi untuk menghasilkan sinyal ucapan yang diinginkan. Synthesizer jenis ini telah lama digunakan pada berbagai aplikasi. Walaupun dapat menghasilkan ucapan dengan tingkat kemudahan interpretasi yang baik, synthesizer ini tidak dapat menghasilkan ucapan dengan tingkat kealamian yang tinggi.

Synthesizer yang menggunakan teknik *diphone concatenation* bekerja dengan cara menggabung-gabungkan segmen-segmen bunyi yang telah direkam sebelumnya. Setiap segmen berupa *diphone* (gabungan dua buah fonem). Synthesizer jenis ini dapat menghasilkan bunyi ucapan dengan tingkat kealamian (*naturalness*) yang tinggi. Struktur sistem seperti di atas pada prinsipnya merupakan konfigurasi tipikal yang digunakan pada berbagai sistem Text to Speech berbagai bahasa. Namun demikian, pada setiap sub-sistem terdapat sifat-sifat serta proses-proses yang sangat spesifik dan sangat tergantung dari bahasanya.

Konversi dari teks ke fonem sangat dipengaruhi oleh aturan-aturan yang berlaku dalam suatu bahasa. Pada prinsipnya proses ini melakukan konversi dari simbol-simbol tekstual menjadi simbol-simbol fonetik yang merepresentasikan unit bunyi terkecil dalam suatu bahasa. Setiap bahasa memiliki aturan cara pembacaan dan cara pengucapan teks yang sangat spesifik. Hal ini menyebabkan implementasi unit konverter teks ke fonem menjadi sangat spesifik terhadap suatu bahasa. Untuk mendapatkan ucapan yang lebih alami, ucapan yang dihasilkan harus memiliki intonasi (*prosody*). Secara kuantisasi, prosodi adalah perubahan nilai *pitch* (frekuensi dasar) selama pengucapan kalimat dilakukan atau pitch sebagai fungsi waktu. Pada prakteknya, informasi pembentuk prosodi berupa data-data pitch serta durasi pengucapannya untuk setiap fonem yang dibangkitkan. Nilai-nilai yang dihasilkan diperoleh dari suatu model prosodi. Prosodi bersifat sangat spesifik untuk setiap bahasa, sehingga model yang diperlukan untuk membangkitkan data-data prosodi menjadi sangat spesifik juga untuk suatu bahasa. Beberapa model umum prosodi pernah dikembangkan, tetapi untuk digunakan pada suatu bahasa masih perlu banyak penyesuaian yang harus dilakukan.

Konverter fonem ke ucapan berfungsi untuk membangkitkan sinyal ucapan berdasarkan kode-kode fonem yang dihasilkan dari proses sebelumnya. Sub sistem ini harus memiliki pustaka setiap unit ucapan dari suatu bahasa. Pada sistem yang menggunakan teknik *diphone concatenation*, sistem harus didukung oleh suatu *diphone database* yang berisi rekaman segmen-segmen ucapan yang berupa diphone. Ucapan dalam suatu bahasa dibentuk dari satu set bunyi yang mungkin berbeda untuk setiap bahasa, oleh karena itu setiap bahasa harus dilengkapi dengan diphone database yang berbeda. Tahapan-tahapan utama konversi dari teks menjadi ucapan dapat dinyatakan dengan diagram seperti terlihat pada Gambar 1.3.



Gambar 1.3 Diagram lengkap Pengubah Tulisan menjadi Ucapan

Tahap berikutnya adalah melakukan konversi dari teks yang sudah secara lengkap merepresentasikan kalimat yang ingin diucapkan menjadi kode-kode fonem. Konversi teks menjadi fonem biasanya dilakukan dengan dua cara. Sebagian proses konversi dapat dilakukan dengan aturan konversi yang sederhana dan berlaku umum untuk berbagai kondisi. Sebagian proses lainnya bersifat

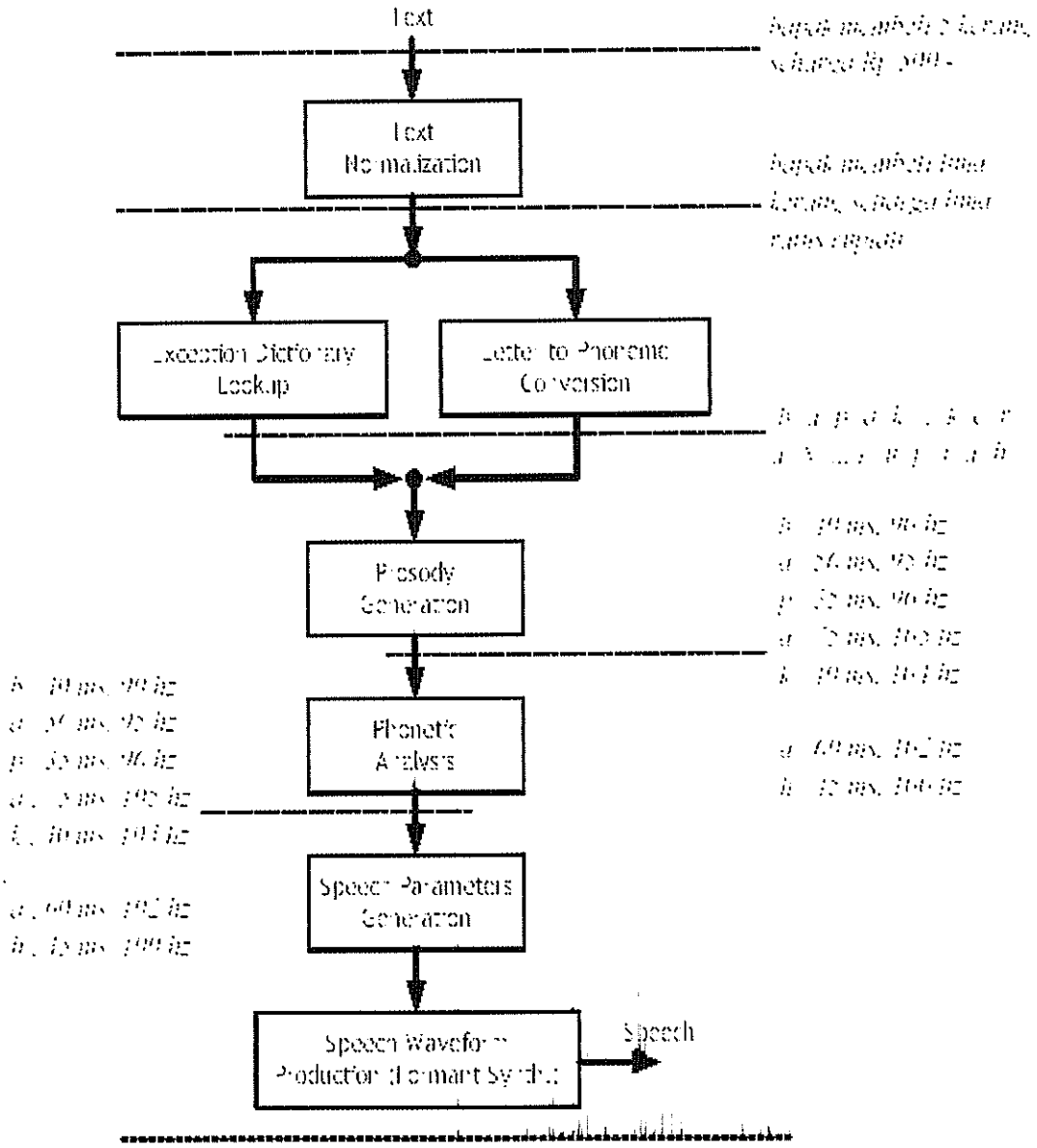
kondisional, tergantung dari huruf-huruf atau fonem-fonem tetangganya, bahkan terdapat bentuk-bentuk translasi yang tidak dapat ditemukan keteraturannya.

Konversi yang teratur dapat diimplementasikan dengan tabel konversi yang berisi pasangan antara urutan huruf dan urutan fonem, bahkan mungkin hanya berisi satu huruf dan satu fonem. Aturan yang lebih sulit biasanya diimplementasikan dengan tabel konversi yang akan diterapkan jika kondisi rangkaian huruf tetangga kiri dan kanannya terpenuhi. Contoh bentuk aturan konversi huruf ke fonem yang memenuhi teknik tersebut adalah sebagai berikut.

Left-context [letter-set] right-context = phoneme string

Huruf tertentu yang ditunjuk dalam posisi ***[letter-set]*** akan dikonversikan menjadi suatu
fonem dalam "***phoneme string***" jika ***left-context*** dan ***right context*** terpenuhi. Bahasa Inggris termasuk bahasa yang mempunyai keteraturan yang rendah untuk proses konversi teks ke fonem. Suatu TTS bahasa Inggris biasanya dilengkapi dengan suatu basis data yang berisi ribuan kata serta konversi padanan urutan fonemnya. Bahasa Indonesia termasuk bahasa yang jelas aturan konversinya. Sebagian besar kata dalam Bahasa Indonesia dapat dikonversikan menjadi fonem dengan aturan yang jelas dan sederhana, walaupun tetap ada kondisi-kondisi yang tidak dapat ditemukan keteraturannya. Sebagai contoh, simbol huruf e dapat diucapkan sebagai *e pepet* atau *e taling*, artinya harus dikonversikan menjadi fonem yang berbeda untuk kondisi yang berbeda. Dalam blok diagram di atas, kondisi yang masih dapat ditangani oleh aturan diimplementasikan dengan blok *Letter to Phoneme Conversion.* Konversi yang tidak teratur ditangani oleh bagian *Exception Dictionary Lookup*. Hasil dari tahap tersebut adalah rangkaian fonem yang merepresentasikan bunyi kalimat yang ingin diucapkan. Bagian *prosody generator* akan melengkapi setiap unit fonem yang dihasilkan dengan data durasi pengucapannya serta pitchnya. Data durasi serta pitch diperoleh berdasarkan kombinasi antara tabel atau database serta model prosodi. Secara simbolik, hasil dari bagian ini sudah menghasilkan informasi yang cukup untuk menghasilkan ucapan yang diinginkan.

Satu tahap berikutnya yang masih sering dilakukan adalah *Phonetic Analysis*. Tahap ini dapat dikatakan sebagai tahap penyempurnaan, yaitu melakukan perbaikan di tingkat bunyi. Sebagai contoh, dalam bahasa Indonesia, fonem /k/ dalam kata *bapak* tidak pernah diucapkan secara tegas, atau adanya sisipan fonem /y/ dalam pengucapan kata *alamiah* antara fonem /i/ dan /a/.



Gambar 1.4 Besaran dalam tiap tahap pengubahan tulisan menjadi ucapan

Bab 2
Sinyal Suara

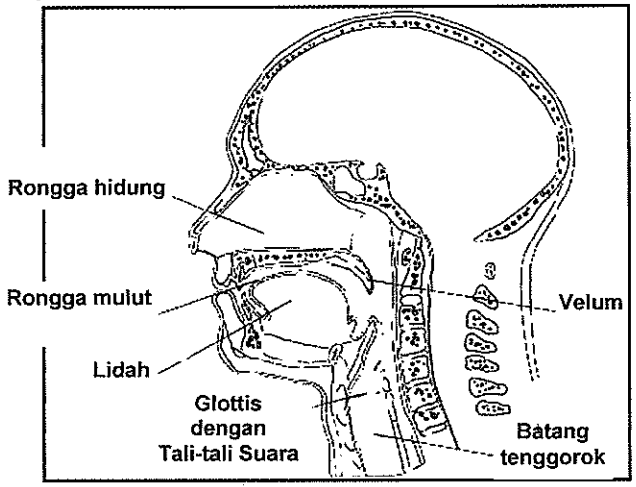
2.1 Teori Produksi Suara

Bunyi atau suara dihasilkan dari getaran yang diteruskan oleh medium udara, ditangkap oleh telinga manusia normal. Bunyi sebagai getaran udara dapat pula merupakan hasil yang dibuat oleh alat ucap manusia seperti pita suara, lidah, dan bibir. Bunyi bahasa dibuat oleh manusia untuk mengungkapkan sesuatu, yang mana dapat juga berwujud nyanyian, tuturan dan bentuk lainnya.

2.1.1 Pembentukan Bunyi Bahasa

Dalam pembentukan bunyi bahasa ada 3 faktor utama yang berperan yaitu sumber tenaga, alat ucap yang menimbulkan getaran, dan rongga pengubah getaran. Proses pembentukan bunyi bahasa dimulai dengan memanfaatkan pernafasan sebagai sumber tenaganya. Pada saat kita mengeluarkan nafas, paru-paru kita menghembuskan tenaga berupa arus udara. Arus udara itu bisa mengalami perubahan pada pita suara yang terletak pada pangkal tenggorokan. Arus udara dari paru-paru itu dapat membuka kedua pita suara yang merapat sehingga mengakibatkan corak bunyi bahasa tertentu. Gerakan membuka dan menutup pita suara itu menyebabkan arus udara dan udara di sekitar pita suara berubah tekanannya atau bergetar. Perubahan bentuk saluran suara yang terdiri dari rongga faring, rongga mulut, dan rongga hidung menghasilkan bunyi bahasa yang berbeda-beda. Udara dari paru-paru dapat keluar dari rongga mulut, rongga hidung, atau keduanya sekaligus. Bunyi bahasa yang arus udaranya keluar lewat rongga mulut disebut bunyi oral, misalnya /p/, /g/, /f/; bunyi bahasa yang arus udaranya keluar dari hidung disebut bunyi sengau atau bunyi nasal, misalnya /m/ dan /n/. Sedang bunyi yang arus udaranya keluar lewat rongga hidung dan rongga mulut disebut bunyi yang disengaukan atau dinasalisasi.

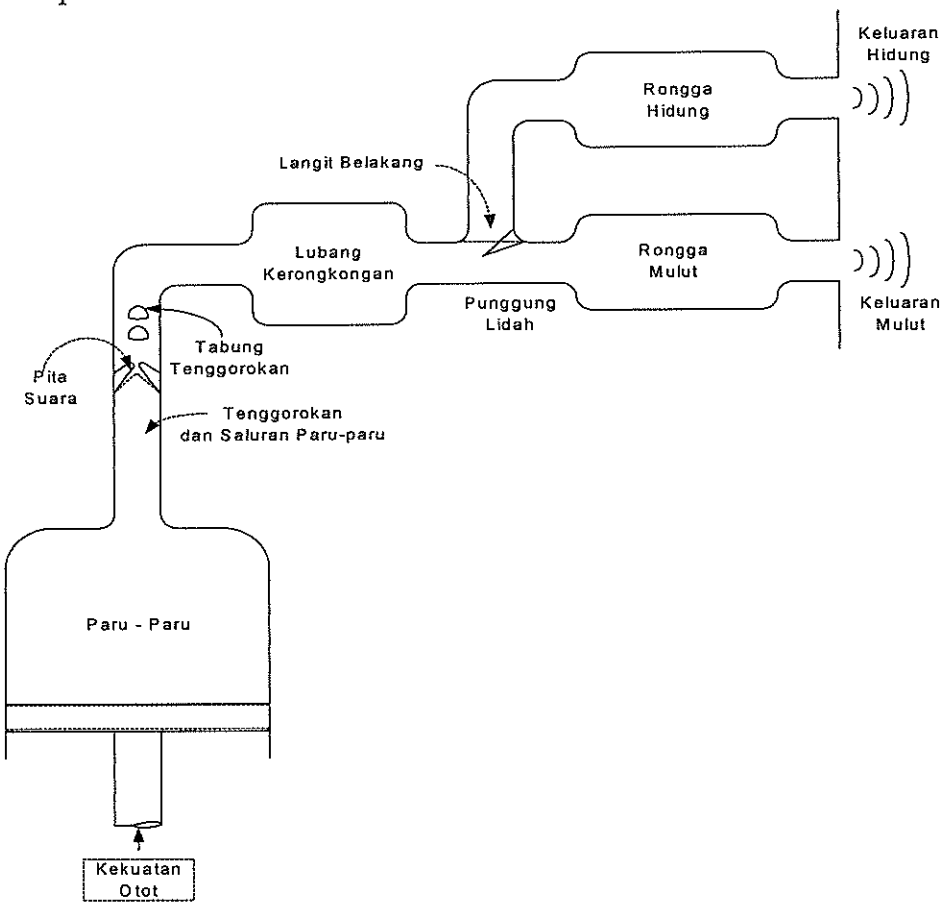
Gambar 2.1 memperlihatkan alat ucap yang digunakan dalam pembentukan bunyi bahasa.



Gambar 2.1 Organ-organ produksi suara

Celah pita suara (*vocal cord*) yang menegang pada pangkal tenggorokan akan bergetar oleh aliran udara. Aliran udara dicuplik dalam bentuk pulsa periode seperempat gelombang yang kemudian mengalami modulasi frekuensi melalui hulu kerongkongan, rongga mulut dan kemungkinan rongga hidung. Bunyi suara hasil proses diatas akan berbeda-beda tergantung dari posisi artikulator. Artikulator adalah bagian alat ucap yang dapat digerakkan yang mana meliputi rahang, lidah dan langit-langit belakang, bibir dan mulut.

Contoh mekanisme sederhana proses terbentuknya sinyal bicara dapat dilihat pada Gambar 2.2.



Gambar 2.2 Mekanisme Sederhana Terbentuknya Sinyal Bicara

Paru-paru dan otot diibaratkan sebagai penghasil mekanisme vokal. Otot akan mendorong udara keluar dari paru-paru, kemudian mengalir melalui tenggorokan dan cabangnya. Ketika celah pita suara menegang aliran udara yang ada akan menggetarkannya sehingga menghasilkan bunyi. Bunyi ini disebut bunyi bersuara / vokal. Jika celah pita suara mengendur maka aliran udara akan mengalir pada jalur vokal yang sesak sehingga terjadi pusaran udara (*turbulensi*) didalam jalur vokal yang menghasilkan bunyi. Bunyi ini disebut bunyi tidak bersuara / konsonan.

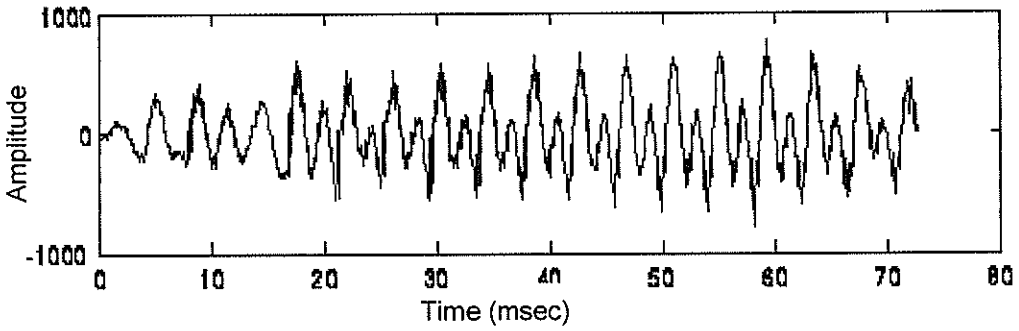
Jadi disini dapat diamati bahwa saat kita berbicara, suara yang dihasilkan dapat berubah-ubah disebabkan karena kondisi celah pita suara yang seperti halnya posisi, ukuran, dan bentuk artikulator dapat berubah-ubah dari waktu ke waktu tergantung suara yang akan dihasilkan.

Organ-organ pembentuk sinyal suara meliputi :

1. Bagian *Pulmonary Tract*; terdiri dari paru-paru dan batang tenggorokan. Paru-paru berfungsi untuk membangkitkan tekanan udara, sedangkan batang tenggorokan sebagai tempat aliran udara oleh tekanan dari paru-paru.
2. Bagian *Larynk*; terdiri dari tali-tali vokal. Bagian ini berfungsi mengubah aliran udara dari batang tenggorokan menjadi pulsa-pulsa udara. Ruang antara pita suara disebut *glottis*.
3. Bagian *Vocal Tract*; terdiri dari *pharynk*, rongga mulut, dan rongga hidung.
   Vocal tract merupakan rongga tempat terjadinya proses resonansi pulsa-pulsa udara yang masuk.

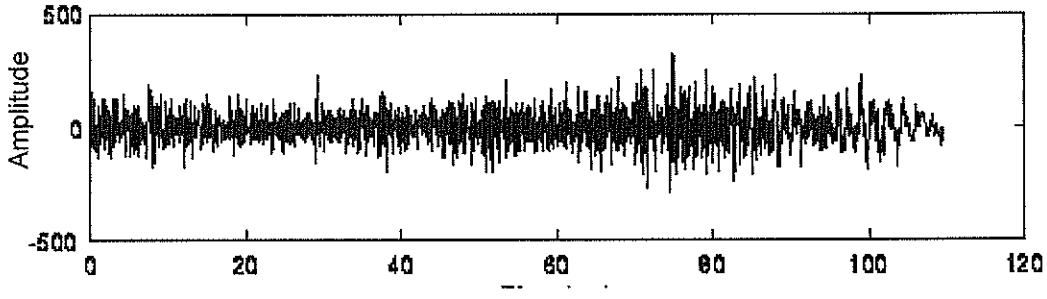
Berdasarkan cara rangsangan (eksitasi) terhadap *vocal tract*, suara yang dihasilkan dibagi menjadi 3 yaitu :

1. Suara Voiced (*Voiced Sound*)
   Tekanan udara yang dibangkitkan dilewatkan pada *glottis* dengan pita suara yang tegang. Getaran pita suara akan mengubah aliran udara menjadi pulsa-pulsa yang *quasi-periodic*. Selanjutnya pulsa-pulsa ini akan merangsang *vocal tract* untuk menghasilkan suara. Contoh suara *voiced* antara lain: /a/ , /i/ , /u/ , /e/ , dan /o/. Bentuk sinyal suara *voiced* dapat dilihat pada Gambar 2.3 yang menunjukkan bunyi vokal /i/.



Gambar 2.3 Bentuk sinyal *voiced* yang berbunyi /i/

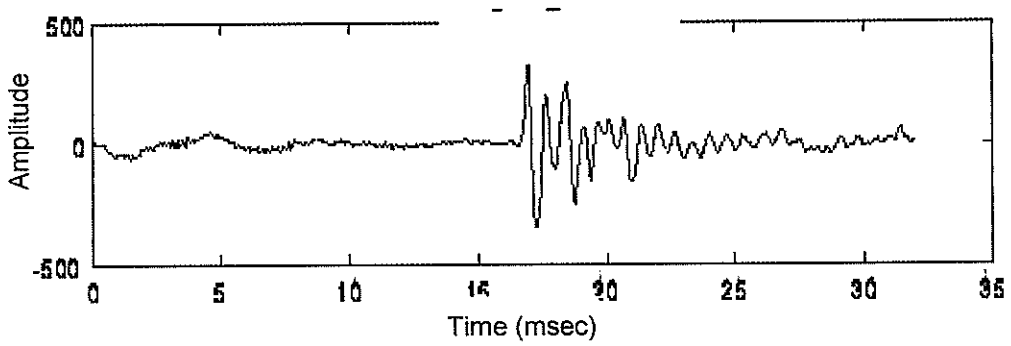
2. Suara Desah (*Fricative/Unvoiced Sound*)
   Suara desah diakibatkan penyempitan pada beberapa bagian *vocal tract*. Aliran udara yang berasal dari tenggorokan dipaksa melewati daerah penyempitan tersebut dengan kecepatan yang cukup tinggi. Akibatnya timbul turbulensi udara yang mengeksitasi *vocal tract* untuk menghasilkan suara desah. Turbulensi ini bertindak sebagai sumber gangguan (*noise*) yang memiliki spektrum sinyal yang lebar. Contoh suara desah : /sh/, /s/



Time (msec)

Gambar 2.4 Bentuk sinyal suara desah yang berbunyi /s/

3. Suara letupan (*Plosive/Stop Sound*)
   Adanya penutupan pada bagian *vocal tract* (biasanya di bagian depan) mengakibatkan aliran udara berkumpul di belakangnya sehingga menimbulkan tekanan yang lebih besar. Bagian *vocal tract* yang tertutup ini dibuka secara tiba-tiba untuk menghasilkan eksitasi letupan. Eksitasi ini menghasilkan suara dengan spektrum yang mirip dengan spektrum *noise*. Contoh suara letupan : /t/, /p/, /g/. Pada bunyi /t/ aliran udara dihambat oleh ujung lidah yang bersentuhan dengan pangkal gigi atas. Udara yang tertahan sejenak kemudian dilepaskan menghasilkan bunyi bahasa.



Gambar 2.5 Bentuk sinyal suara letupan (*plosive*) yang berbunyi /g/

Fricative dan *Plosive Sound* tidak menyebabkan pita suara bergetar sehingga keduanya disebut juga *Unvoiced Sound*

2.1.2 Klasifikasi Bunyi dan Vokal

Pada umumnya bunyi dibedakan atas vokal dan konsonan. Bunyi vokal dihasilkan dengan pita suara terbuka sedikit . Pita suara yang terbuka sedikit ini menjadi bergetar ketika dilalui arus udara yang dipompakan dari paru-paru. Selanjutnya arus udara itu keluar melalui rongga mulut yang berbentuk tertentu sesuai dengan jenis vokal yang dihasilkan.

Vokal (*vowel*) adalah bunyi bahasa yang arus udaranya tidak mengalami rintangan dan kualitasnya ditentukan oleh 3 faktor: tinggi-rendahnya posisi lidah, bagian lidah yang dinaikkan, dan bentuk bibir pada pembentukan vokal tersebut. Pada saat vokal diucapkan, lidah dapat dinaikkan atau diturunkan bersama rahang. Bagian lidah yang dinaikkan atau diturunkan itu dapat di bagian depan, tengah atau belakangnya. Dalam bahasa Indonesia terdapat 5 vokal yaitu /a/ , /i/ , /u/ , /e/ , dan /o/. Tabel 2.1 memperlihatkan kelima vokal bahasa Indonesia berdasarkan parameter tinggi-rendah dan depan-belakang lidah dalam proses pembentukannya.

Tabel 2.1 Posisi lidah dalam vokal Bahasa Indonesia

Bagian Lidah	Depan	Tengah	Belakang
Tinggi	i		u
Sedang	e		o
Rendah		a	

Di samping tinggi-rendah dan depan-belakang lidah seperti digambarkan Tabel 2.1, kualitas vokal juga dipengaruhi bentuk bibir. Untuk vokal tertentu, seperti /a/, bentuk bibir adalah normal, sedangkan untuk vokal /u/ bibir dimajukan sedikit dan bentuknya agak bundar. Untuk vokal /i/ bibir direntangkan ke kiri dan ke kanan sehingga bentuknya melebar. Dengan 3 faktor itu bunyi vokal dapat berciri tinggi, depan, dan bibir terentang, misalnya bunyi /i/, atau tinggi, belakang, dan bibir bundar, misalnya bunyi /u/. Supaya suara yang dikeluarkan menjadi jelas dan indah, maka cara pengucapan huruf hidup (a, i, u, e, o) dengan bentuk mulut yang benar, salah satunya dengan melakukan sikap sebagai berikut.

- Pengucapan /a/ : Mulut dibuka kurang lebih selebar dua jari. Lidah ditarik ke dalam dan suara didukung dengan getaran dalam rongga mulut sehingga terdengar utuh.
- Pengucapan /e/ : Mulut dibuka lebih kecil dari pengucapan /a/, dan setelah dilebarkan ke kiri dan ke kanan. Pengucapan /e/ dibunyikan menggema.
- Pengucapan /i / : Bentuk mulut pengucapan /i/ hampir sama dengan pengucapan /e/. Perbedaanya ialah bibir atas dan bibir bawah lebih dirapatkan dan dibunyikan dengan menggema. Usahakan agar udara yang keluar dari mulut dapat mendukung menggetarkan rongga hidung dan rongga kepala.
- Pengucapan /o/ : Mulut dibuka agak lebar, bibir dibentuk bulat dan rongga mulut dibuka cekung. Lidah ditarik ke dalam.
- Pengucapan /u/ : Untuk mengucapkan /u/ mulut dibuka lebih kecil dari pengucapan /o/.

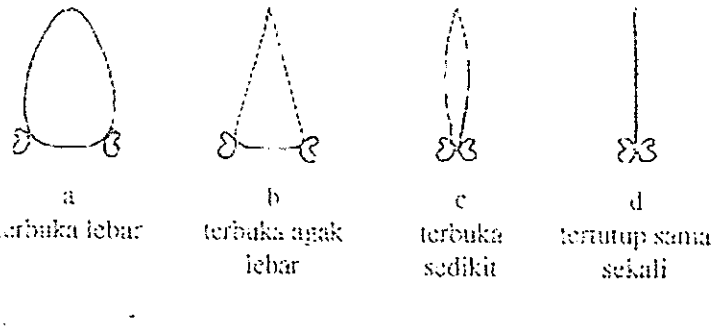
Bunyi konsonan terjadi, setelah arus udara melewati pita suara yang terbuka sedikit atau agak lebar, diteruskan ke rongga mulut atau hidung dengan mendapat hambatan di tempat-tempat artikulasi tertentu. Jadi, beda terjadinya bunyi vokal dan konsonan adalah; arus udara dalam pembentukkan bunyi vokal, setelah melewati pita suara, tidak mendapat hambatan apa-apa; sedangkan dalam pembentukkan bunyi konsonan arus udara itu masih mendapat hambatan atau gangguan. Bunyi konsonan ada yang bersuara ada yang tidak. Yang bersuara terjadi apabila pita suara terbuka sedikit, dan yang tidak bersuara apabila pita suara terbuka agak lebar. Bunyi vokal, semuanya adalah bersuara, sebab dihasilkan dengan pita suara terbuka sedikit.

Bunyi vokal biasanya diklasifikasikan berdasarkan posisi lidah dan bentuk mulut[2]. Posisi lidah bisa bersifat vertikal dan horisontal. Secara vertikal dibedakan adanya vokal tinggi, misalnya bunyi /i/ dan /u/, vokal tengah, misalnya bunyi /e/ dan *[ə]*, dan vokal rendah, misalnya bunyi /a/. Secara horisontal dibedakan adanya vokal depan, misalnya bunyi /i/ dan /e/, vokal pusat, misalnya bunyi *[ə]*, dan vokal belakang, misal bunyi /u/ dan /o/. Kemudian menurut bentuk mulut dibedakan adanya vokal bundar dan vokal tak bundar. Disebut vokal bundar karena bentuk mulut membundar ketika mengucapkan vokal itu, misalnya vokal

/o/ dan /u/. Disebut vokal tak bundar karena bentuk mulut tidak membundar, melainkan melebar, pada waktu mengucapkan vokal tersebut, misalnya vokal /i/ dan /e/.

Pita suara bisa juga dirapatkan sehingga udara menggetarkannya secara teratur pada kecepatan yang berbeda-beda sewaktu udara itu memaksa melewati pita suara tersebut. Getaran ini secara teknis disebut bersuara *(voiced)*, yang juga merupakan sumber penting dari tinggi nada *(pitch)*. Getaran ini dapat dirasakan dari luar dengan menyentuh bagian depan laring atau jakun dengan ujung jari ketika mendengungkan atau mengucapkan *Ah*. Jika tidak ada getaran disebut tak bersuara *(unvoiced)*.

Berdasarkan dengan hambatan pada pita suara ini, perlu dijelaskan adanya empat macam posisi pita suara seperti pada Gambar 2.6 :



Gambar 2.6 Posisi Pita Suara Manusia

Kalau posisi pita suara terbuka lebar,yakni pada posisi (2.6.a), berarti tidak ada hambatan apa-apa, maka berarti juga tidak ada bunyi yang dihasilkan. Posisi ini adalah posisi untuk bernapas secara normal. Bunyi baru dapat dihasilkan kalau ada hambatan atau gangguan terhadap arus udara yang dipompakan dari paru-paru. Oleh karena itu, pita suara dengan posisi (2.6.b), (2.6.c), dan (2.6.d) adalah awal dari adanya bunyi bahasa. Posisi (2.6.b) akan menghasilkan bunyi tak bersuara *(unvoiced)*, karena tidak ada getaran apa-apa pada pita suara ketika arus udara melewatinya. Posisi (2.6.c) akan menghasilkan bunyi bersuara *(voiced)* apabila arus udara itu diteruskan ke rongga mulut atau rongga hidung, karena terjadi getaran pada pita suara ketika arus udara melewatinya. Sedangkan posisi (2.6.d) langsung menghasilkan bunyi hamzah atau bunyi glotal.

Setelah melewati pita suara, tempat awal terjadinya bunyi, arus udara diteruskan ke alat-alat ucap (alat ucap supraglotal) tertentu yang terdapat di rongga mulut atau rongga hidung, dimana bunyi bahasa tertentu dihasilkan. Tempat bunyi ini terjadi disebut tempat artikulasi; proses terjadinya disebut proses artikulasi; dan alat-alat yang digunakan disebut *artikulator*. Berdasarkan proses artikulasi, artikulator dibedakan menjadi artikulator aktif dan artikulator pasif. Artikulator aktif adalah alat ucap yang bergerak atau digerakkan, misalnya, bibir bawah, ujung lidah, dan daun lidah. Sedangkan artikulator pasif adalah alat ucap yang bergerak, atau yang didekati oleh artikulator aktif, misalnya, bibir atas, gigi atas, dan langit-langit keras. Keadaan, cara, atau posisi bertemuanya artikulator aktif dan artikulator pasif disebut striktur.

Bunyi vokal biasanya diklasifikasikan berdasarkan posisi lidah dan bentuk mulut. Posisi lidah bisa bersifat vertikal dan horisontal. Secara vertikal dibedakan adanya vokal tinggi, misalnya bunyi *[i]* dan *[u]*, vokal tengah, misalnya bunyi *[e]* dan *[ə]*, dan vokal rendah, misalnya bunyi *[a]*. Secara horisontal dibedakan adanya vokal depan, misalnya bunyi *[i]* dan *[e]*, vokal pusat, misalnya bunyi *[ə]*, dan vokal belakang, misal bunyi *[u]* dan *[o]*. Kemudian menurut bentuk mulut dibedakan adanya vokal bundar dan vokal tak bundar. Disebut vokal bundar karena bentuk mulut membundar ketika mengucapkan vokal itu, misalnya vokal *[o]* dan *[u]*. Disebut vokal tak bundar karena bentuk mulut tidak membundar, melainkan melebar, pada waktu mengucapkan vokal tersebut, misalnya vokal *[i]* dan *[e]*.

Diftong atau vokal rangkap

Vokal rangkap adalah vokal dimana posisi lidah ketika memproduksi bunyi pada bagian awalnya dan bagian akhirnya tidak sama. Ketidaksamaan itu menyangkut tinggi rendahnya lidah, bagian lidah yang bergerak, serta strikturnya. Namun, yang dihasilkan bukan dua buah bunyi, melainkan hanya sebuah bunyi karena beradadalam satu silabel. Contoh diftong dalam bahasa Indonesia adalah kata *kerbau, harimau, cukai* dan *landai*. Apabila ada dua buah vokal berturutan, namun yang pertama terletak pada suku kata yang berlainan dari yang kedua, maka tidak ada diftong. Contohnya pada kata *bau* dan *lain*.

2.1.3 Klasifikasi Konsonan

Bunyi-bunyi konsonan dibedakan berdasarkan tempat artikulasi dan cara artikulasi [2].

Tempat artikulasi adalah alat ucap yang digunakan dalam pembentukkan bunyi. Berdasarkan tempat artikulasi dibedakan konsonan :

1. *Bilabial*

   Yaitu konsonan yang terjadi pada kedua belah bibir, bibir bawah merapat pada bibir atas. Yang termasuk konsonan *bilabial* adalah bunyi *[b]*, *[p]* dan *[m]*.

2. *Labiodental*

   Yaitu konsonan yang terjadi pada gigi bawah dan bibir atas; gigi bawah merapat pada bibir atas. Yang termasuk konsonan *labiodenta*l adalah bunyi *[f]* dan *[v]*.

3. *Laminoalveolar*

   Yaitu konsonan yang terjadi pada daun lidah dan gusi; dalam hal ini, daun lidah menempel pada gusi. Yang termasuk konsonan l*aminoalveolar* adalah bunyi *[t]* dan *[d]*.

4. *Dorsovelar*

   Yaitu konsonan yang terjadi pada pangkal lidah dan velum atau langit-langit lunak. Yang termasuk konsonan *dorsovelar* adalah bunyi *[k]* dan *[g]*.

Berdasarkan cara artikulasinya, artinya bagaimana gangguan atau hambatan yang dilakukan terhadap arus udara, dapat dibedakan adanya konsonan :

1. Hambat

Disini artikulator menutup sepenuhnya aliran udara, sehingga udara mampat di belakang tempat penutupan itu. Kemudian penutupan itu terbuka secara tiba-tiba, sehingga menyebabkan terjadinya letupan. Yang termasuk konsonan hambat antara lain, bunyi *[p]*, *[b]*, *[t]*, *[k]* dan *[g]*.

2. Geseran
   Disini artikulator aktif mendekati artikulator pasif, membentuk celah sempit, sehingga udara yang lewat mendapat gangguan dicelah. Contoh yang termasuk konsonan geseran adalah bunyi *[f]*, *[s]*, dan *[z]*.
3. Paduan
   Disini artikulator aktif menghambat sepenuhnya aliran udara, lalu membentuk celah sempit dengan artikulator pasif. Cara ini merupakan gabungan antara hambatan dan frikatif. Yang termasuk konsonan paduan, antara lain, bunyi *[c]* dan *[j]*.
4. Sengauan

   Di sini artikulator menghambat sepenuhnya aliran udara melalui mulut, tetapi membiarkannya keluar melalui rongga hidung dengan bebas. Contoh konsonan sengauan adalah bunyi *[m]*, *[n]*, dan *[ɳ]*.
5. Getaran

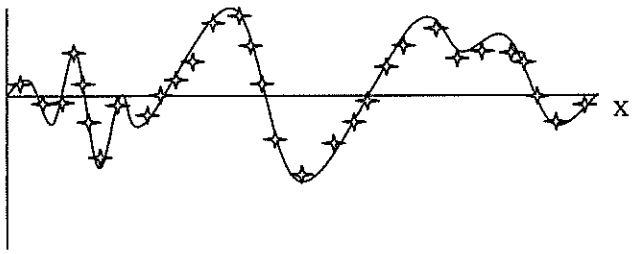
   Di sini artikulator aktif melakukan kontak beruntun dengan artikulator pasif, sehingga getaran bunyi terjadi berulang-ulang. Contohnya adalah konsonan *[r]*.
6. Sampingan

   Di sini artikulator aktif menghambat aliran udara pada bagian tengah mulut, lalu membiarkan udara keluar melalui samping lidah. Contohnya adalah konsonan *[l]*.
7. Hampiran

   Di sini artikulator aktif dan pasif membentuk ruang yang mendekati posisi terbuka seperti dalam pembentukan vokal, tetapi tidak cukup sempit untuk menghasilkan konsonan geseran. Oleh karena itu, bunyi yang dihasilkan sering disebut semi vokal. Di sini hanya ada dua buah, yaitu *[w]* dan *[y]*.

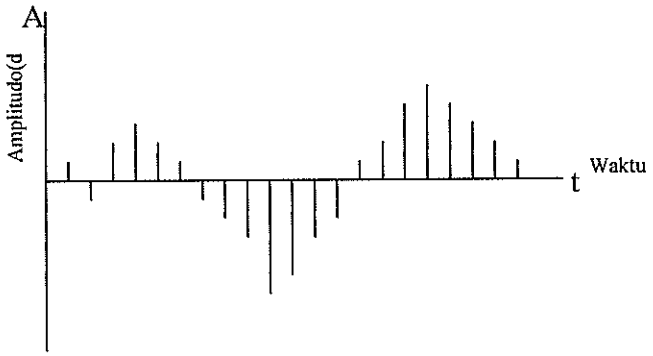
2.2 Sinyal Suara

Definisi dari sinyal suara yaitu suatu sinyal yang mewakili dari suara[12]. Sinyal suara dibentuk dari kombinasi berbagai frekuensi pada berbagai amplitudo dan fasa. Untuk dapat melakukan proses pengolahan sinyal suara langkah pertama yaitu penyamplingan. Dimana pencuplikan adalah mengukur masukan sinyal suara pada sela waktu tertentu, kemudian mengkonversinya ke dalam skala tersendiri, dan menyimpannya. Gambar 2.7 menunjukkan sinyal suara yang dicuplik pada frekuensi 8 KHz. Ini berarti bahwa terjadi pengukuran masukan sinyal suara sebanyak 8000 kali selama 1 detik. Fungsi pencuplikan yaitu agar sinyal suara dapat disimpan dalam bentuk digital.



Gambar 2.7 Sinyal suara yang dicuplik pada frekuensi pencupikan 8 KHz[12]

Pada Gambar 2.8 yaitu sinyal suara yang sudah ditampilkan dalam bentuk digital, dimana bentuk sinyal bukanlah sinyal kontinyu tetapi merupakan sinyal diskret.



Gambar 2.8 Sinyal suara dalam bentuk diskret.[12]

Menganalisis suatu suara dapat diartikan menganalisis suara dalam komponen frekuensi dan waktu dari suara tersebut. Dimana frekuensi menentukan tinggi rendahnya suara sedangkan komponen waktu sangat menentukan kapan suatu suara dibunyikan. Karena dengan mengetahui komponen-komponen tersebut sinyal suara dapat dianalisis dengan rinci.

Agar semua informasi isyarat analog tidak hilang setelah mengalami pencuplikan, maka proses pencuplikan tersebut harus mengikuti teorema pencuplikan, yaitu jika komponen frekuensi tertinggi didalam suatu sinyal adalah f_{maks} , maka sinyal tersebut harus dicuplik pada frekuensi paling sedikit $2f_{maks}$ agar sampel dapat menggambarkan sinyal asli kembali secara penuh. Frekuensi cuplik dapat dinyatakan dengan $F_s \geq 2f_{maks}$, , dimana Fs adalah frekuensi pencuplikan atau disebut laju Nyquist. Oleh karena itu, jika komponen frekuensi maksimum dalam sinyal analog adalah 4 KHz, maka agar bisa mendapatkan semua informasi dalam sinyal tersebut, kita harus mencupliknya dengan kecepatan frekuensi 8 KHz atau lebih.

2.2.1 Persepsi Pendengaran Terhadap Suara

Pendengaran manusia terhadap suara sangat terbatas karena manusia hanya dapat mendengar suatu bunyi pada daerah frekuensi 16 Hz sampai 20 kHz. Diatas 20 kHz sinyal suara tidak akan kita dengar, sebaliknya dibawah 16 Hz sinyal suara akan terdengar lambat dan berdetak-detak. Sedangkan level suara yang masih aman kita dengar adalah antara 0 dB hingga 130 dB. Diatas level 130 dB,

suara tersebut akan menyakitkan telinga kita, dan dapat pula mengakibatkan selaput telinga kita pecah.

Disamping pendengaran terbatas persepsi pendengaran manusia terhadap suara dapat berbeda-beda tergantung pada keras atau tidaknya suara, tinggi rendahnya nada, warna suara dan gaya bahasa seseorang.

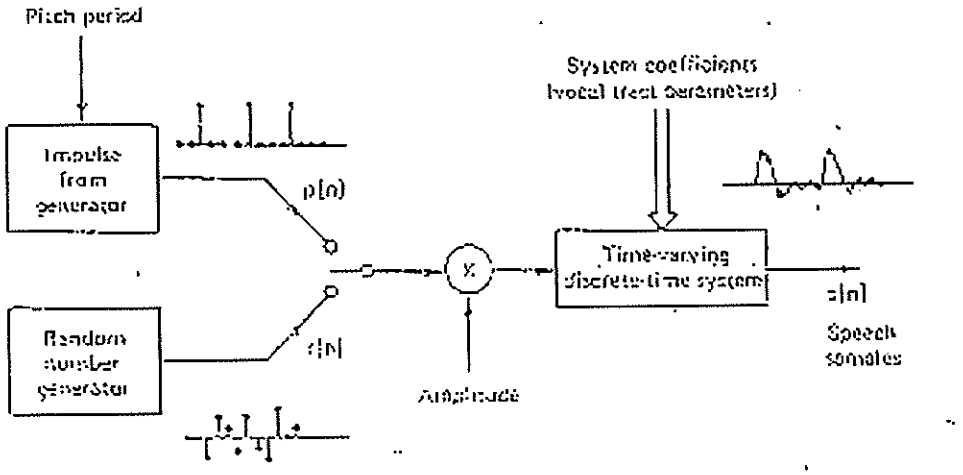
Suara yang terdengar akan terasa keras apabila amplitudo sinyal suara yang sampai telinga cukup besar. Hal ini biasanya terjadi pada awal kata yang diucapkan karena saat tersebut timbul transisi dari diam ke aras suara tertentu. Sebaliknya suara yang terdengar akan terasa pelan atau lembut apabila amplitudo sinyal suara yang terdengar cukup kecil. Biasanya hal ini terjadi pada akhir kata yang diucapkan karena saat tersebut terjadi transisi dari aras suara tertentu ke keadaan diam.

Tinggi nada yang terdengar berhubungan dengan frekuensi suara yang tertangkap telinga. Makin tinggi frekuensi suara yang dihasilkan akan semakin nyaring suara itu terdengar. Frekuensi suara manusia tergolong ke dalam frekuensi rendah yang berkisar 320 hingga 4000 Hz.

Warna suara berhubungan dengan pola spektrum dan bentuk pita suara yang membentuknya. Disini akan lebih mudah untuk mengungkapkan perbedaan warna suara daripada menjelaskan mekanisme yang terjadi. Hal ini juga nantinya akan mempengaruhi adanya perbedaan pola spektrum pada fonem vokal antara /a/, /i/, /u/, /e/, dan /o/ .

Bab 3
Analisis Homomorphic Sinyal Suara

3.1 Pemodelan Sinyal Ucapan



Gambar 3.1 Model Waktu Diskrit Produksi Sinyal Ucapan

Gambar 3.1 adalah penggambaran bentuk gelombang sinyal ucapan pada waktu diskrit. Pada model ini, contoh sinyal ucapan dianggap bentuk keluaran konvolusi dari variasi waktu sistem waktu diskrit dan resonansi sistem jalur vokal. Bentuk permodelan saklar antara impulse periodik dan derau acak tergantung pada tipe suara yang dihasilkan.

Homomorphic dekonvolusi dapat diterapkan untuk menghitung parameter sinyal ucapan, jika dianggap model berada pada interval waktu tertentu. L adalah panjang sampel sinyal ucapan, sehingga sinyal ucapan diasumsikan sebagai hasil dari konvolusi 2 buah sinyal.

$$s(n) = v(n) * p(n) \qquad untuk\ 0 \le n \le L-1 \tag{3.1}$$

dengan $v(n)$ adalah respon impuls jalur vokal dan $p(n)$ adalah periodic (untuk sinyal suara ucapan) atau random noise (untuk sinyal ucapan tak bersuara).

Pada kenyataannya, model dari persamaan (3.1) terdapat efek diskontinuitas pada awal dan akhir sinyal, sehingga diperlukan penjedelaan untuk memperhalus awal dan akhir sinyal. Sehingga masukan sistem homomorphic dekonvolusi adalah :

$$x(n) = w(n)s(n)$$

jika $w(n)$ berubah lambat terhadap perubahan $v(n)$, maka analisis dapat dituliskan,

$$x(n) = v(n) * p_w(n)$$

dengan

$$p_w(n) = w(n)\, p(n)$$

3.2 Pembingkaian dan Penjendelaan

Salah satu sifat dari sinyal ucapan adalah berubah terhadap waktu (*non stationary*), sehingga untuk menganalisa sinyal ucapan diperlukan pembatasan waktu sinyal ucapan yaitu pembingkaian dan penjendelan.

Pembingkaian adalah pembatasan sinyal suara yang digunakan untuk pergeseran penjendelaan (*windowing*).

Pada tahap ini tiap sinyal didalam masing-masing bingkai diatas dijendelakan satu per satu. Kegunaan dari penjendelaan ini adalah untuk mengurangi kesenjangan (*discontinuitas*) sinyal pada awal dan akhir bingkai (*frame*). Dengan penjendelaan ini sinyal akan meruncing menuju nol pada awal dan akir bingkai. Adapun persamaan penjendelaan adalah:

$$x(n) = s(n) \; . \; w(n), \qquad 0 \le n \le N-1 \tag{3.2}$$

Untuk metode analisis cepstrum pada buku ini digunakan jendela Hamming sebagai berikut:

$$w(n) = 0{,}54 - 0{,}46 \cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1 \tag{3.3}$$

Jendela hamming dipilih, karena jendela ini sering dipakai pada analisa sinyal untuk sistem pengenalan pola (*pattern recognition*).

Metode penjendelaan (*window*) pada umumnya dimulai dengan adanya respon frekuensi ideal :

$$H_d\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} h_d(n) e^{-j\omega n} \tag{3.4}$$

dimana $h_d(n)$ adalah respon deret impuls, yaitu

$$h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\omega}) e^{j\omega n} d\omega \tag{3.5}$$

cara yang paling mudah untuk membentuk sistem dari $h_d(n)$ adalah dengan membentuk sistem respon impuls baru

$$h(n) = \begin{cases} h_d(n) & ,0 \le n \le M \\ 0 & ,lainnya \end{cases} \tag{3.6}$$

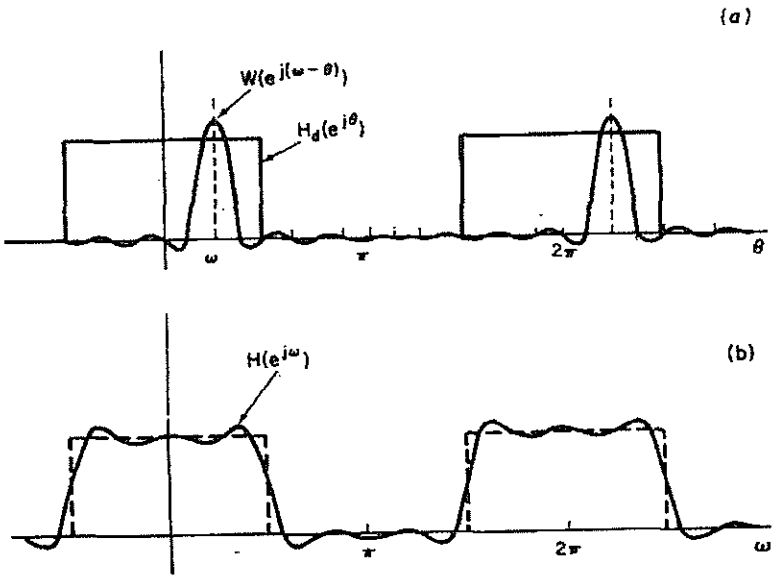
Sehingga h(n) dapat dikatakan sebagai respon dari respon impuls dengan durasi terbatas *window* w(n),

$$h(n) = h_d(n)\, w(n) \tag{3.7}$$

Sesuai dengan teori *windowing* , maka

$$H(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d\left(e^{j\theta}\right) W\left(e^{j\omega-\theta}\right) d\theta \tag{3.8}$$

$H(e^{j\omega})$ adalah hasil dari konvolusi periodik respon frekuensi dan Fourier Transform dari *window*. $H(e^{j\omega})$ akan disesuaikan dengan bentuk dari respon $H_d(e^{j\omega})$. Pada Gambar 3.2 (a) akan diperlihatkan bentuk dari $H_d(e^{j\theta})$dan $W_d(e^{j(\omega-\theta)})$, yang kemudian akan membentuk $H(e^{j\omega})$ (Gambar 3.2(b)).



Gambar 3.2

(a) Proses konvolusi dari $H_d(e^{j\theta})$ dan $W_d(e^{j(\omega-\theta)})$;(b) Hasil dari proses *windowing*

Jika W(n) =1 untuk semua n, $W(e^{j\omega})$ adalah impuls periodik dengan periode 2π, maka $H(e^{j\omega})=H_d(e^{j\omega})$

Karena sifat dari sinyal ucapan adalah *non-stationary*, sehingga untuk menganalisis sinyal ucapan diperlukan pembatasan waktu sinyal ucapan dengan menggunakan penjendelan.

Pada tahap ini tiap sinyal didalam masing-masing bingkai diatas dijendelakan satu per satu. Kegunaan dari penjendelaan ini adalah untuk mengurangi kesenjangan (diskontinuitas) sinyal pada awal dan akhir bingkai (*frame*).

3.3 Sistem Homomorphic

Sistem Homomorphic adalah usaha Pengolahan Sinyal Digital membuat suatu proses dimana tidak linier seperti fungsi biasanya dan ditunjukkan dengan transformasi linear secara aljabar antara masukan dan keluaran yang mematuhi prinsip general superposisi .

Cepstrum merupakan transformasi Fourier dari logaritma spektrum suatu sinyal. Transformasi dari sinyal menjadi cepstrumnya disebut transformasi homomorphic, dan konsep dari cepstrum merupakan bagian dasar teori sistem homomorphic untuk pemrosesan sinyal yang dikombinasikan dengan konvolusi.

3.3.1 Prinsip Superposisi

Prinsip superposisi seperti yang telah ditetapkan untuk sistem linear menunjukkan bahwa, jika T adalah transformasi sistem, kemudian untuk dua masukan $x_1(n)$ dan $x_2(n)$ dan scalar c ,

$$T[x_1(n) + x_2(n)] = T[x_1(n)] + T[x_2(n)] \quad (3.9a)$$

dan $$T[cx_1(n)] = cT[x_1(n)] \quad (3.9b)$$

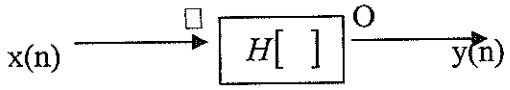
Untuk membuat umum sistem ini, kita artikan □ sebuah aturan untuk menggabungkan masukan satu dengan yang lain (misalnya penjumlahan,

perkalian dan konvolusi) dan dengan : untuk menggabungkan masukan dengan scalar. Begitu juga dengan O kita artikan untuk menggabungkan sistem keluaran dan Δ untuk menggabungkan keluaran dengan scalar. Kemudian dengan H menunjukkan transformasi sistem, kita dapat membuat persamaan (3.9) menjadi :

$$H[x_1(n) \square x_2(n)] = H[x_1(n)] \, O \, H[x_2(n)] \quad (3.10a)$$

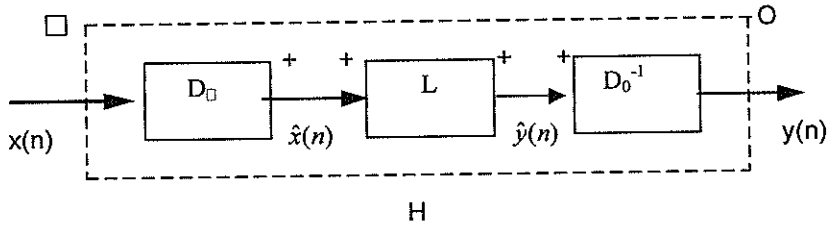
$$H[c : x_1(n)] = c \Delta H[x_1(n)] \quad (3.10b)$$

Beberapa sistem dikatakan memenuhi prinsip superposisi dengan operasi masukan □ dan operasi keluaran O. Beberapa sistem ditunjukkan seperti Gambar 3.3. Secara jelasnya, sistem linear adalah kasus khusus yang mana □ dan O adalah penjumlahan dan : dan Δ adalah perkalian.



Gambar 3.3 Sistem Homomorphic dengan Operasi Masukan □ dan Operasi Keluaran O , dan Transformasi Sistem $H[\ ]$

Jika masukan sistem merupakan sebuah ruang vektor dengan □ dan : menunjukkan penjumlahan vektor dan perkalian skalar dan keluaran sistem merupakan sebuah ruang vektor dengan O dan Δ menunjukkan penjumlahan vektor dan perkalian skalar, maka semua sistem kelas ini dapat ditunjukkan sebagai kaskade dari tiga sistem, seperti dalam Gambar 3.4.



Gambar 3.4. Sistem Homomorphic Kanonik.

Sistem $D_□$ mempunyai sifat :

$$D_□ [x_1(n) □ x_2(n)] = D_□ [x_1(n)] + D_□ [x_2(n)]$$
$$= \hat{x}_1(n) + \hat{x}_2(n) \quad (3.11a)$$

$$D_□ [c : x_1(n)] = cD_□ [x_1(n)] = c\hat{x}_1(n) \quad (3.11b)$$

L adalah sistem linear, sehingga :

$$L [\hat{x}_1(n) + \hat{x}_2(n)] = L [\hat{x}_1(n)] + L [\hat{x}_2(n]$$
$$= \hat{y}_1(n) + \hat{y}_2(n) \quad (3.12a)$$

$$L [c\hat{x}_1(n)] = cL [\hat{x}_1(n)] = c\hat{y}_1(n) \quad (3.12b)$$

Akhirnya, sistem $D^{-1}{}_O$ ditransformasikan dari penjumlahan ke O, menjadi :

$$D_o^{-1}[\hat{y}_1(n) + \hat{y}_2(n)] = D_O^{-1} [\hat{y}_1(n)] \, O \, D_O^{-1} [\hat{y}_2(n)]$$
$$= y_1(n) \, O \, y_2(n)$$
$$D_O^{-1} [c\hat{y}_1(n)] = c \Delta D_O^{-1} [\hat{y}_1(n)] = c \Delta y_1(n)$$

Karena sistem $D_□$ ditetapkan dengan operasi □ dan : hal itu merupakan karakteristik dari kelas sehingga disebut sistem karakteristik untuk operasi □. Begitu juga D_O adalah sistem karakteristik untuk operasi O.

3.3.2 Karakteristik Sistem untuk Konvolusi

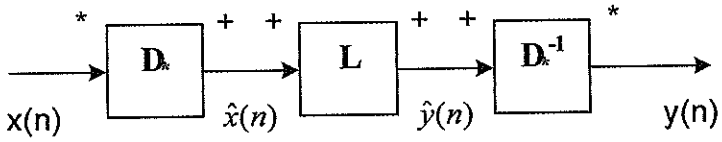
Gambar 3.5 adalah bentuk kanonik kelas sistem yang mematuhi prinsip superposisi untuk konvolusi. Jika masukan adalah:

$$x(n) = x_1(n) * x_2(n) = \sum_{k=-\infty}^{\infty} x_1(k)\, x_2(n-k), \qquad (3.13)$$

dan keluarannya adalah :

$$y(n) = y_1(n) * y_2(n)$$

dimana $y_1(n)$ *dan* $y_2(n)$ adalah respons dari sistem $x_1(n)$ *dan* $x_2(n)$. Penggunaan sistem dengan bentuk seperti Gambar 3.5 untuk mengubah satu komponen dari konvolusi disebut *homomorphic deconvolusi.*



Gambar 3.5 Bentuk Kanonik Sistem Homomorphic dengan Konvolusi pada Operasi Masukan dan Keluaran

Sistem pertama dari sistem kanonik pada Gambar 3.5 adalah sistem karakteristik untuk konvolusi. Itu adalah sistem homomorphic dengan konvolusi pada operasi masukan dan penjumlahan pada operasi keluaran, sehingga dengan :

$$x(n) = x_1(n) * x_2(n),$$

Dan sistem D* mempunyai karakteristik:

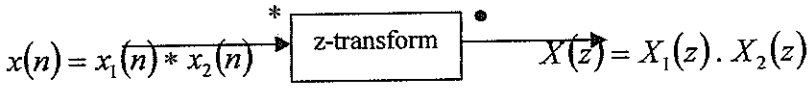
$$D_*\left[x_1(n) * x_2(n)\right] = D_*\left[x_1(n)\right] + D_*\left[x_2(n)\right]$$
$$= \hat{x}_1(n) + \hat{x}_2(n)$$
$$D_*\left[c : x_1(n)\right] = cD_*\left[x_1(n)\right] = c\hat{x}_1(n)$$

Sistem L adalah sistem linear dan D_*^{-1} adalah invers dari D_*.

Untuk menggambarkan karakteristik sistem D_*, didasarkan pada transformasi –z dari persamaan di atas yaitu :

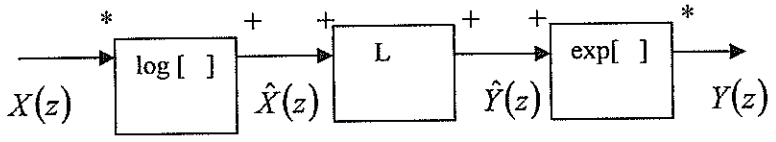
$$X(z) = X_1(z).\, X_2(z)$$

dengan operasi z-transform dapat dijabarkan sebagai transformasi homomorphic dengan konvolusi pada operasi masukan dan perkalian pada operasi keluaran seperti pada Gambar 3.6. menggunakan transformasi-z, kombinasi konvolusi dapat diubah menjadi perkalian.



Gambar 3.6 Z- Transform Digambarkan sebagai Transformasi Homomorphic dari Konvolusi ke Perkalian

Apabila digambarkan sinyal dengan z-transform, maka sistem kanonik pada Gambar 3.5 dapat diubah seperti pada Gambar 3.7 :



Gambar 3.7 Sistem dari Gambar 3.5 dengan Sinyal yang digambarkan dengan Transformasi –Z nya.

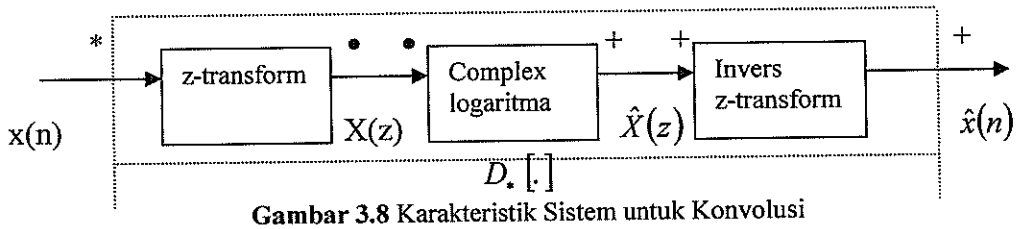
Sistem $D_*[\cdot]$ dapat dipilih misalnya bahwa $\hat{x}[n]$ adalah complex cepstrum $x[n]$, sehingga kita bisa menggambarkan sistem D_* seperti Gambar 3.8.

$$\hat{X}(z) = \log [X(z)]$$

Dan karena $X(z) = X_1(z).X_2(z)$, maka :

$$\hat{X}(z) = \log [X(z)] = \log [X_1(z)] + \log [X_2(z)]$$
$$= \hat{X}_1(z) + \hat{X}_2(z)$$

Transformasi –Z memetakan konvolusi ke dalam perkalian, complex logaritma merubah perkalian menjadi penjumlahan dan invers z-transform adalah transformasi linear.



Gambar 3.8 Karakteristik Sistem untuk Konvolusi

3.3.3 Complex Cepstrum

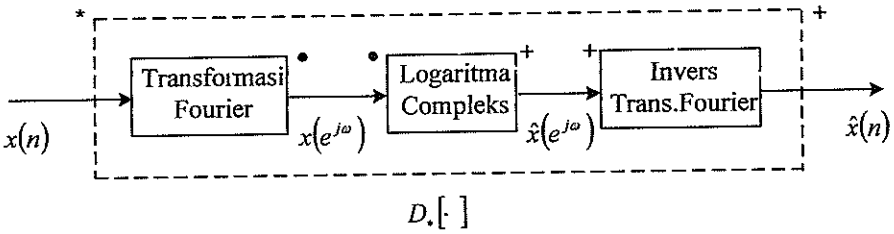
Jika sistem $D_*[\cdot]$ pada Gambar 3.8 digambarkan dalam bentuk Transformasi Fourier, dengan mengganti nilai z dengan $z = e^{j\omega}$ akan didapatkan persamaan :

$$X(e^{j\omega}) = \sum_{-\infty}^{\infty} x(n).e^{-j\omega n} \qquad (3.14)$$

$$\hat{X}(e^{j\omega}) = \log[X(e^{j\omega})] \qquad (3.15)$$

$$\hat{x}(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{X}(e^{j\omega})e^{j\omega n}\,d\omega \qquad (3.16)$$

Dan didapatkan bentuk kanoniknya seperti pada Gambar 3.9.



Gambar 3.9 Sistem Pada Gambar 3.8 dengan Transformasi Fourier

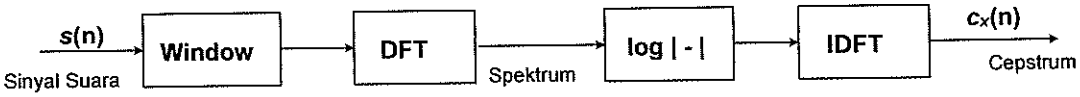
Sedangkan $D_*[\cdot]$ dalam sistem homomorphic dapat dipilih misalnya bahwa $\hat{x}[n]$ adalah complex cepstrum $x[n]$. Complex Cepstrum terdiri dari bagianreal dan imajiner, yang bila dituliskan dalam transformasi Fourier:

$$\hat{x}(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left[X\left(e^{jw}\right)\right]e^{j\omega n}d\omega$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[\log\left|X\left(e^{j\omega}\right)\right| + j\arg X\left(e^{j\omega}\right)\right]e^{j\omega n}d\omega$$

Berdasar persamaan terakhir, yang akan dipakai dalam analisis homomorphic ini hanyalah bentuk realnya yang disebut *Real Cepstrum* atau biasa disebut *cepstrum.*

Untuk proses perhitungan real ceptrum dalam dilihat pada Gambar 3.10:



Gambar 3.10 Blok Analisa Perhitungan Real Cepstrum

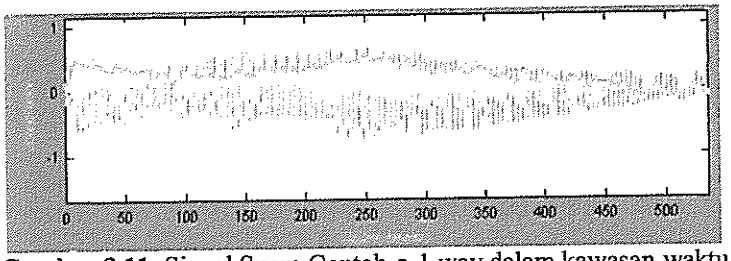
Sinyal suara $s(n)$, melalui penjendelaan data yang ada, di-transformasi Fourier-kan menghasilkan sesuatu yang disebut dengan *spektrum*, kemudian hasilnya di-log-kan dan di-invers-kan menghasilkan keluaran yang disebut dengan *cepstrum.* Dengan kata lain, cepstrum $c_x(n)$ atau real cepstrum merupakan invers transformasi fourier dari logaritma magnitudo transformasi Fourier. Sehingga berdasarkan Gambar 3.10, bentuk cepstrum sinyal $s(n)$ dan dapat ditulis :

$$c_x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\log\left|X\left(e^{j\omega}\right)\right| e^{j\omega n}d\omega$$

Untuk memperlihatkan kegunaan analisis homomorphic atas suatu sinyal suara selanjutnya akan dijelaskan penerapannya sebagai berikut. Analisis dilakukan dengan lebih dari satu sampel suara yang semuanya menggunakan ekstension wav yang setiap sampelnya mengucapkan lima buah fonem vokal yaitu /a/, /i/, /u/, /e/, dan /o/. Hal ini untuk menghindari ketidaktelitian dalam analisis, karena setiap manusia dalam mengucapkan vokal tidak selalu sama persis, tapi tergantung keras tidaknya suara, lafal maupun intonasinya.

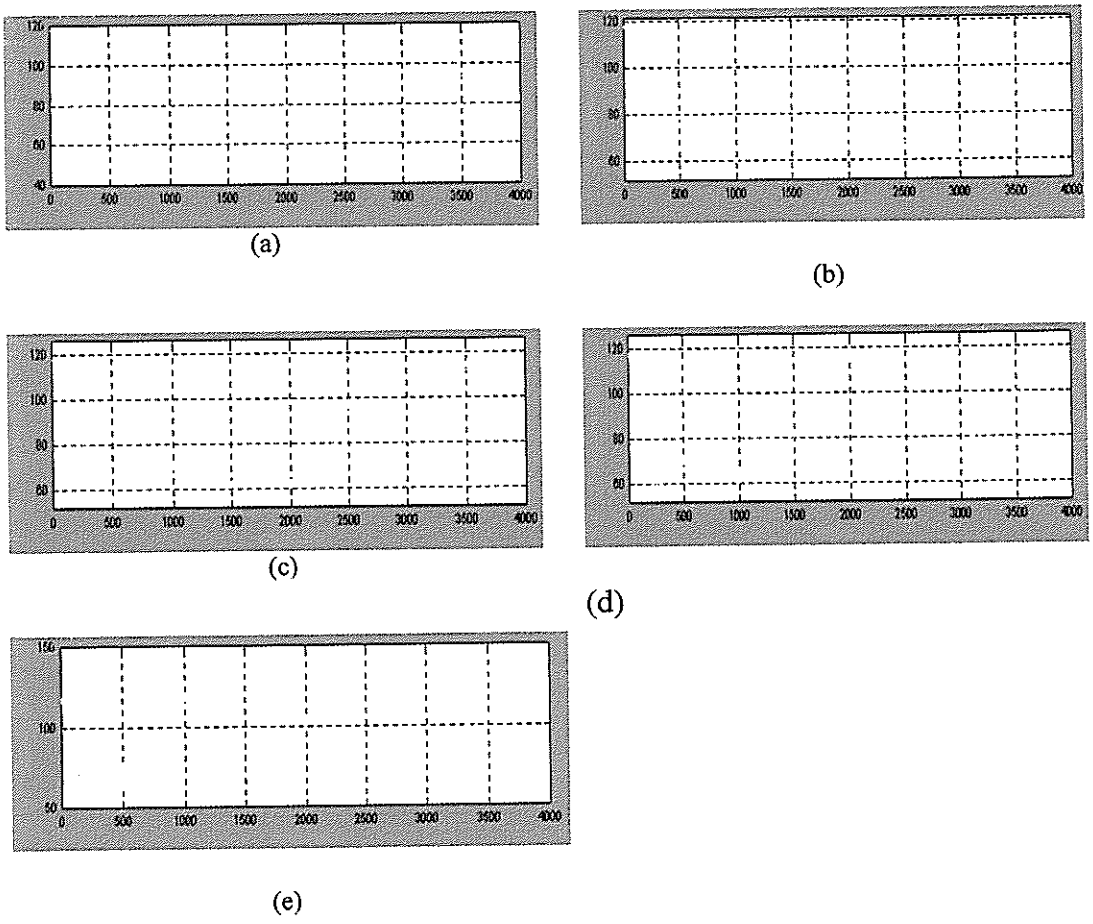
Analisis Homomorphic Untuk Vokal /a/

Pada Gambar 3.11 terlihat tampilan sinyal suara dalam kawasan waktu untuk vokal /a/ dari sampel contoh-a-1.



Gambar 3.11 Sinyal Suara Contoh-a-1.wav dalam kawasan waktu

Adanya nilai amplitudo yang besar dari sinyal suara dikarenakan terjadi penekanan saat mengucapkan vokal /a/. Sedangkan untuk bentuk spektrum vokal /a/ dari sampel contoh-a-1.wav dengan ukuran fft yang berbeda mulai 64, 128, 256,512 sampai 1024 dapat dilihat pada Gambar 3.12 berikut ini.



Gambar 3.12 Bentuk Spektrum Vokal /a/ pada Contoh-a-1.wav (a) pada fft size 64 (b)fft size 128 (c) fft size 256 (d) fft size 512 (e) fft size 1024

Pengaruh ukuran FFT hanya pada pengeplotan titik yang diambil. Seperti yang telah disebutkan bahwa frekuensi sampling yang digunakan adalah 8000 Hz dan durasi waktu perekaman suara adalah 1 detik. Sehingga untuk 1 detik ada sejumlah 8000 titik.

Untuk ukuran fft 64 berarti hanya ada 64 titik dalam waktu :

$$x = \frac{64}{8000} = 8 \text{ milidetik}$$

Untuk ukuran fft 128 berarti terdapat 128 titik dalam waktu :

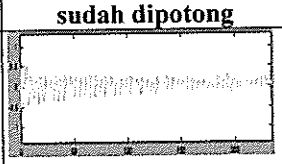
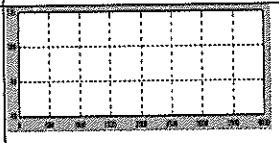
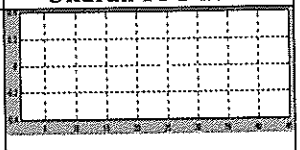
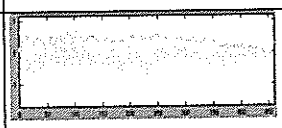
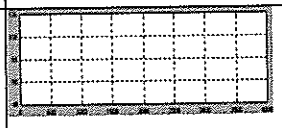
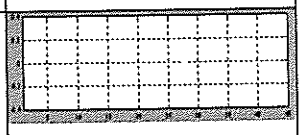
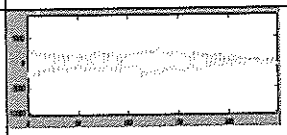
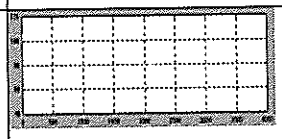
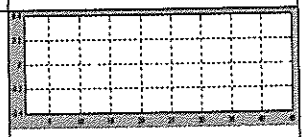
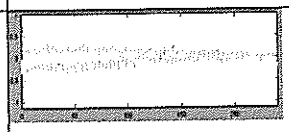
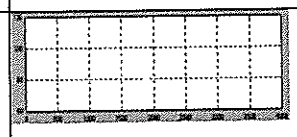
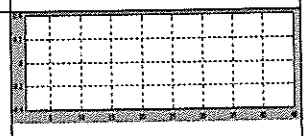
$$x = \frac{128}{8000} = 16 \text{ milidetik}$$

Dengan cara yang sama didapatkan untuk fft size 256 → x = 32 milidetik
512 → x = 64 milidetik
1025 → x = 128 milidetik

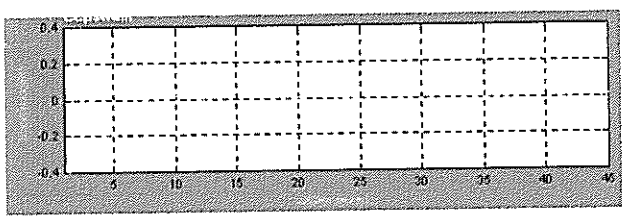
Dengan ukuran fft yang semakin banyak akan semakin banyak titik yang diambil sehingga menghasilkan pengeplotan yang semakin rapat seperti yang terlihat pada Gambar 3.12. Dalam penelitian ini hanya akan ditampilkan spektrum pada fft size 64 dari sampel untuk memperjelas analisis. Dan untuk hasil pengamatan pada sinyal suara vokal /a/ untuk lima penutur bisa dilihat pada Tabel 3.1.

Sinyal yang akan diproses adalah sinyal yang telah dipotong pada bagian awal dan akhir. Karena untuk pemrosesan sinyal yang dibutuhkan hanyalah informasi, sedangkan informasi berada pada tengah sinyal suara. Sedangkan bagian awal dan akhir yang dipotong kemungkinan adalah noise yang terjadi di awal dan akhir pada saat pengucapan, sehingga harus dihilangkan.

Tabel 3.1 Bentuk Tampilan antara Sinyal Asli – Spektrum (fft size 64) dan Cepstrum (fft size 1024) pada Vokal /a/ untuk beberapa penutur

Sampel /a/	Sinyal Asli yang sudah dipotong	Spektrum dengan Ukuran FFT 64	Cepstrum dengan Ukuran FFT 1024
Contoh-a-1			
Contoh-a-2			
Contoh-a-3			
Contoh-a-4			

Besarnya magnitudo yang diperoleh dari sampel adalah kurang dari 130 dB dan masih dalam taraf wajar batas persepsi pendengaran suara manusia. Besarnya magnitudo menunjukkan adanya tekanan pada saat pengucapan. Sehingga antara pengucapan vokal yang satu dengan yang lain meskipun dilakukan oleh orang yang sama dan dengan jenis vokal yang sama hasilnya akan berbeda, tergantung dari tekanan pada saat mengucapkannya. Sedangkan bentuk Cepstrum dari Contoh-a-11.wav dapat dilihat pada Gambar 3.13 berikut ini.



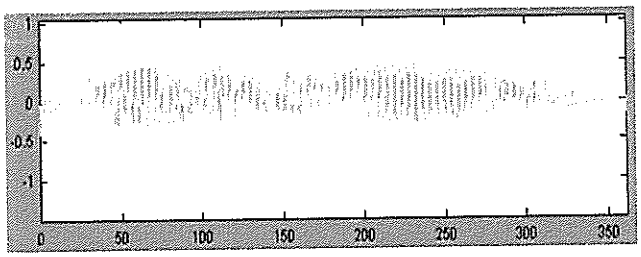
Gambar 3.13 Bentuk Cepstrum dari Vokal /a/ pada Contoh-a-1.wav

Dari Gambar 3.13 bisa diamati bahwa saat berada pada waktu ± 35– 40 milidetik, pada amplitudo terjadi lonjakan nilai yang tinggi yang dianggap sebagai suatu ciri . Dan dari hasil pangujian sebanyak enam kali percobaan untuk sampel Contoh-a-1.wav seperti terlihat pada Tabel 3.1, ciri tersebut juga berada pada interval waktu ± 35 - 40 milidetik.

Ukuran fft pada spektrum tidak berpengaruh pada bentuk cepstrum dari sinyal suara. Ciri cepstral bisa dilihat dengan jelas apabila menggunakan ukuran fft 1024 karena jumlah data yang diambil semakin banyak sehingga data lebih akurat.

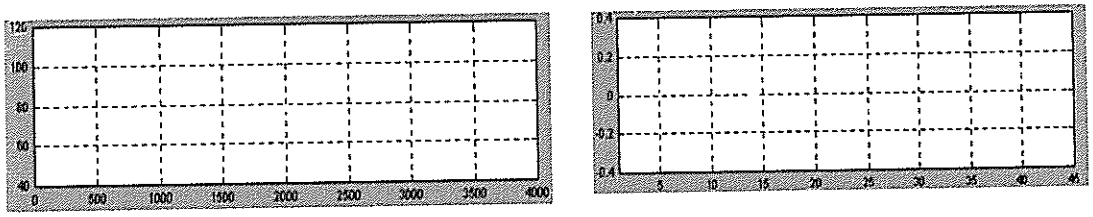
Analisis Homomorphic Untuk Vokal /i/

Pada Gambar 3.14 terlihat tampilan sinyal suara dalam kawasan waktu untuk vokal /i/ pada Contoh-i-1.wav.



Gambar 3.14 Sinyal Suara Asli Vokal /i/ pada Contoh-i-1.wav

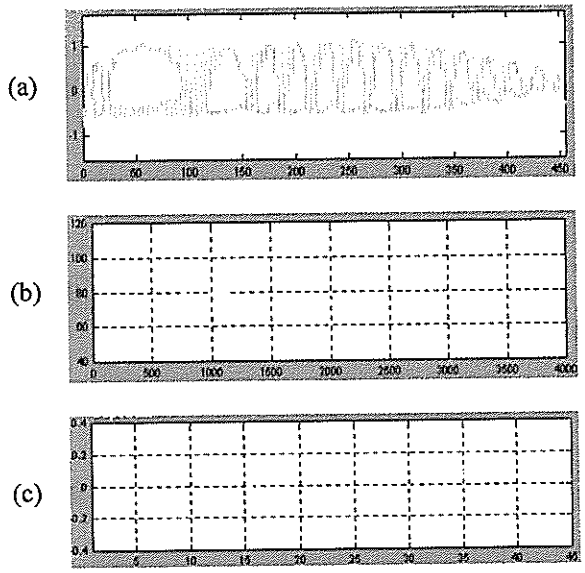
Bentuk spektrum dari vokal /i/ pada sampel Contoh-i-1.wav dengan ukuran fft 1024 dapat dilihat pada Gambar 3.15 (a). Cepstrum yang dihasilkan terlihat pada Gambar 3.15 (b) yang dipilih pada interval 45 milidetik yaitu saat waktu berada pada ±35 – 39 milidetik, terdapat suatu ciri berupa amplitudo menunjukkan nilai yang tinggi. Bila dibandingkan dengan vokal /a/, tidak begitu terlihat adanya perbedaan yang terjadi saat nilai amplitudo mencapai nilai yang melonjak tinggi.



(a) (b)

Gambar 3.15 (a) Bentuk Spektrum Vokal /i/ pada Contoh-i-1.wav dgn fft size 64
(b) Bentuk Cepstrum Vokal /i/ dgn fft size 1024 pada Contoh-i-1.wav

Analisis Homomorphic Untuk Vokal /u/

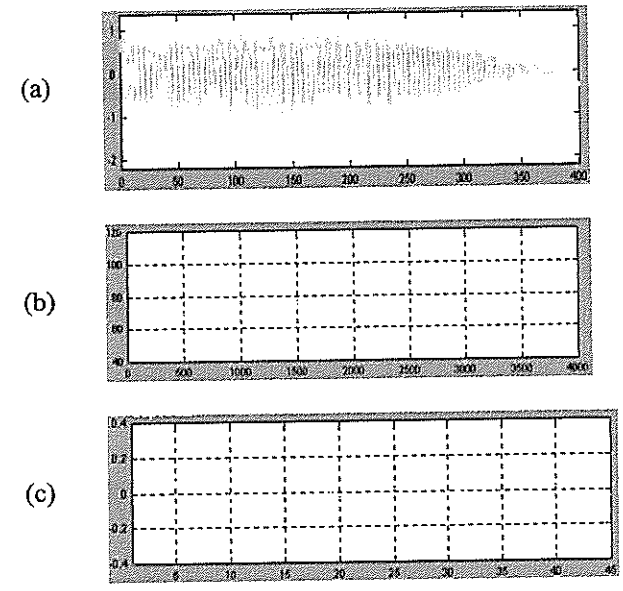


Gambar 3.16 (a) Sinyal Suara Asli Vokal /u/ pada Contoh-u-1.wav
(b) Bentuk Spektrum (c) Bentuk Cepstrum

Dari hasil yang didapat pada Contoh-u-1.wav (Gambar 3.16 c) bisa dilihat bahwa saat berada pada waktu $\pm 30 - 40$ milidetik, pada amplitudo terjadi lonjakan nilai yang tinggi yang dianggap sebagai suatu ciri .

Analisis Homomorphic Untuk Vokal /e/

Bentuk tampilan dari sinyal suara asli, spektrum dan cepstrum vokal /e/ untuk file Contoh-e-1.wav bisa dilihat pada Gambar 3.17 berikut ini.

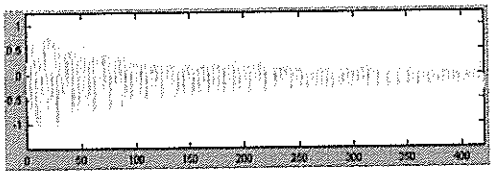


Gambar 3.17 (a) Sinyal Suara Asli Vokal /e/ pada Contoh-u-1.wav (b) Bentuk Spektrum
(c) Bentuk Cepstrum

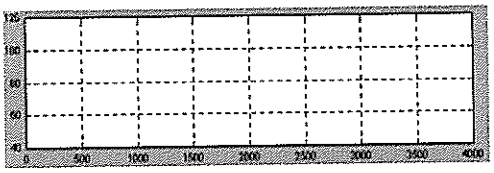
Dari hasil yang didapat pada contoh-e-1.wav (Gambar 3.17 c), nilai amplitudo yang tinggi terjadi pada interval waktu ±35 – 40 milidetik yang dianggap sebagai suatu ciri.

Analisis Homomorphic Untuk Vokal /o/

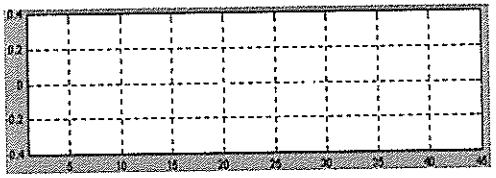
(a)



(b)



(c)

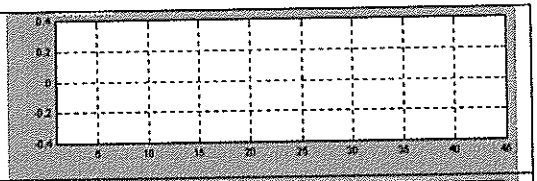


Gambar 3.18 (a) Sinyal Suara Asli Vokal /o/ pada Contoh-o-1.wav (b) Bentuk Spektrum (c) Bentuk Cepstrum

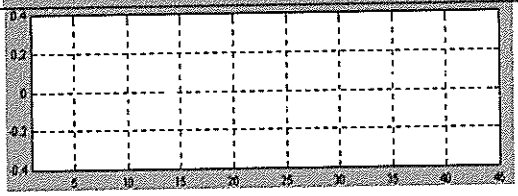
Dari hasil yang didapat pada Contoh-o-1.wav (Gambar 3.18 c) , nilai amplitudo yang tinggi terjadi pada interval waktu ±35 – 44 milidetik yang dianggap sebagai suatu ciri .

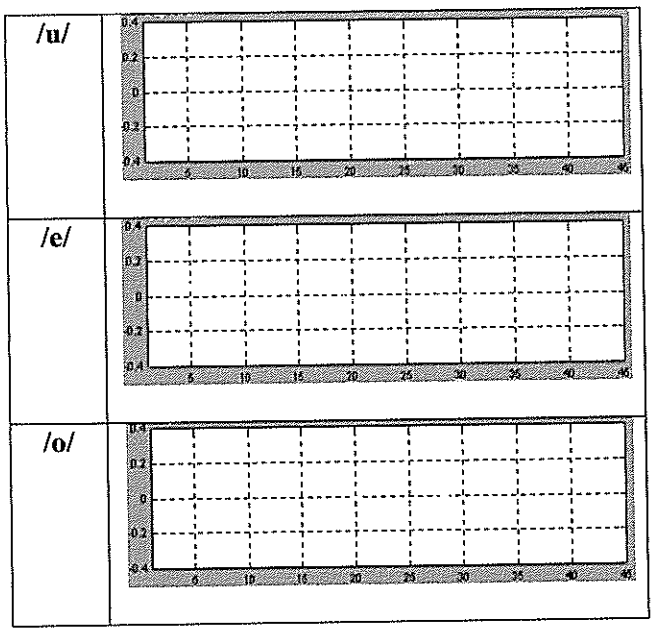
Untuk lebih jelasnya, bentuk cepstrum vokal /a/, /i/, /u/, /e/, dan /o/ dari sampel suara seorang penutur dapat dilihat pada Gambar 3.19 berikut ini.

/a/



/i/





Gambar 3.19 Cepstrum Vokal seorang penutur

Ciri yang menunjukkan cepstrum suatu vokal sinyal suara ditunjukkan dengan adanya nilai amplitudo yang tinggi saat interval waktu tertentu. Ciri tersebut adalah suatu informasi dari vokal suara manusia dan disebut dengan istilah ciri cepstral. Antara /a/, /i/, /u/, /e/ dan /o/ dimulainya informasi tidak selalu berada pada interval waktu yang sama akan tetapi bergeser.

Rumus konvolusi digunakan untuk menghitung respon sistem invarian waktu liniear terhadap setiap masukan sinyal masukan x(n) yang berubah-ubah. Dengan bentuk persamaan,

$$y(n)= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \qquad (3.17)$$

dengan, x(n) adalah sinyal masukan tehadap sistem, h(n) adalah respon impuls sistem dan y(n) adalah keluaran sistem dengan respon terhadap sinyal masukan x(n). Pada saat perhitungan sinyal masukan x(n) diilustrasikan dengan deret masukan x(k) dan h(n) diilustrasikan dengan deret respon impuls sistem h(k), dengan menggunakan k sebagai indeks waktu agar konsisten dengan persamaan (3.17)

Konvolusi mencakup empat operasi, yakni:

1. Pencerminan salah satu respon impuls untuk menghasilkan h(-k)
2. Pergeseran deret yang dicerminkan dengan n satuan waktu untuk menghasilkan salah satu h(n-k)
3. Perkalian dua buah barisan untuk menghasilkan deret produk salah satu dari x(k) h(n-k)
4. Penjumlahan seluruh deret produk untuk menghasilkan keluaran sistem y(n) pada waktu n.

Operasi pencerminan hanya dilakukan sekali , dan ketiga operasi lainnya diulangi untuk semua pergeseran yang mungkin $-\infty < n < \infty$ agar memperoleh y(n) untuk $-\infty < n < \infty$.

3.4 Alihragam Fourier Diskret (*Discrete Fourier Transform*)

Agar suatu sinyal dalam kawasan waktu mudah untuk dianalisis, maka sinyal tersebut dapat dialihragamkan terlebih dahulu ke dalam kawasan frekuensi. Untuk itu dapat digunakan Alihragam Fourier *(Fourier Transform).*

Apabila suatu sinyal diskret dinyatakan dengan runtun $X(n)$, maka Alihragam Fouriernya adalah $X(e^{j\omega})$. Besar $X(e^{j\omega})$ didapat dari persamaan (3.18), sedangkan Alihragam Fourier Balik-nya (*Inverse Fourier Transform*) didapat dari persamaan (2.10) di bawah ini :

$$X(e^{j\omega}) = \sum_{-\infty}^{\infty} x(n).\, e^{-j\omega n} \tag{3.18}$$

$$X(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}).\, e^{j\omega n}\, d\omega \tag{3.19}$$

Agar alihragam tersebut dapat dihitung dengan komputer maka jumlah runtun $x(n)$ yang akan diolah harus terbatas. Dalam hal ini dapat digunakan persamaan Alihragam Fourier Diskret *(Discrete Fourier Transform/* DFT*)* yang menerapkan perhitungan pada $x(n)$ dengan selang tertentu.

$$X(k) = \sum_{n=0}^{N-1} x(n)\ .\ W^{kn} \qquad ,k = 0, 1, 2, 3 \ldots N\text{-}1 \tag{3.20}$$

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k)\ .\ W^{-kn} \qquad ,k = 0, 1, 2, 3 \ldots N\text{-}1 \tag{3.21}$$

dengan $W = e^{-j2\pi / N}$

Persamaan (3.20) menunjukan DFT dan persamaan (3.21) menunjukan Alihragam Fourier Diskret-Balik (*Inverse Discrete Fourier Transform* / IDFT) untuk N buah data. Apabila data yang diolah semakin banyak (N makin besar), maka akan membutuhkan waktu yang semakin lama pula.

3.5 Analisi *Pitch* (F0) untuk *Voiced* atau *Unvoiced Speech*

Salah satu parameter dari sinyal suara adalah frekuensi fundamental. Frekuensi fundamental dalam istilah instrumen musik dikenal sebagai *pitch* atau nilai frekuensi dari suatu jenis nada. *Pitch* atau tinggi nada adalah hasil akustik dari kecepatan getaran pita suara. Semakin cepat getaran pita suara, semakin tinggi, tinggi nadanya. Sehingga *pitch* ini dapat digunakan sebagai ciri bersuara.

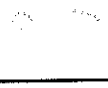
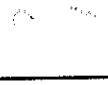
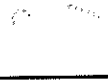
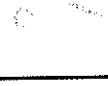
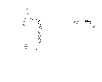
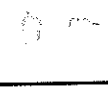
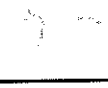
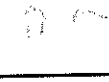
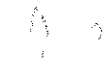
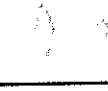
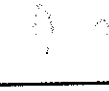
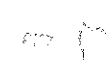
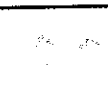
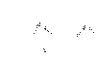
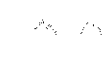
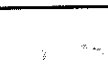
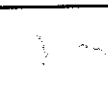
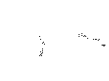
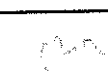
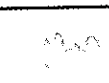
Dengan melihat pemodelan untuk sintesa sinyal suara pada Gambar 2.4, generator impuls memberikan sumber pembangkitan untuk sinyal suara berbunyi berupa fonem vokal (a/e/i/o/u) yang dapat diatur selang waktunya oleh parameter-parameter periode *pitch*. Berdasarkan penelitian, periode *pitch* berkisar antara 10-20 milidetik. Disamping generator impuls, generator derau acak juga berfungsi sebagai sumber pembangkitan untuk sinyal suara tidak bersuara berupa fonem konsonan (h/k/s/t).

Setiap manusia akan mempunyai kisaran *pitch* tersendiri, tergantung dari bentuk pangkal tenggorok yang dimilikinya. Untuk pria memiliki kisaran *pitch* sebesar 50-250 Hz. Dan untuk wanita memiliki kisaran *pitch* sebesar 120-500 Hz. Tinggi rendahnya nilai *pitch* tergantung pada intonasi suara, dan tingkat emosi dari manusia.

Pengujian dilakukan dengan beberapa sampel suara yang mengucapkan kata dalam bahasa Indonesia yaitu, ada, api, fakir, gitar, kaset, kubu, lama, nenek, raja, wakil, yakin dan zakat.

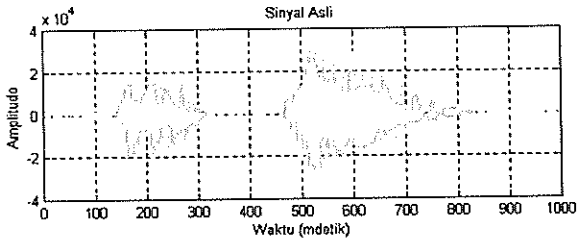
Hasil pengujian penggunaan pembingkaian untuk pengenalan *voiced* dan *unvoiced*

Tabel 3.2 Hasil pengujian pembingkaian

Kata	Pembingkaian				
	15 milidetik	**30 milidetik**	**35 milidetik**	**40 milidetik**	**45 milidetik**
Ada					
Api					
Fakir					
Gitar					
Kaset					
Kubu					
Lama					
Nenek					
Raja					
Wakil					
Yakin					
Zakat					

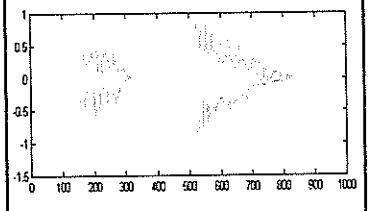
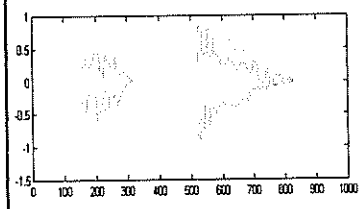
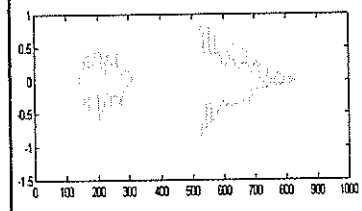
Tabel 3.1 menunjukkan pembingkaian dengan bingkai setiap 15 milidetik, 30 milidetik, 35 milidetik, 40 milidetik dan 45 milidetik, dimana pada pembingkaian setiap 40 milidetik menghasilkan pengenalan *voiced* dan *unvoiced* lebih baik dibandingkan yang lain.

Hasil pengujian dengan sampel suara "ada"



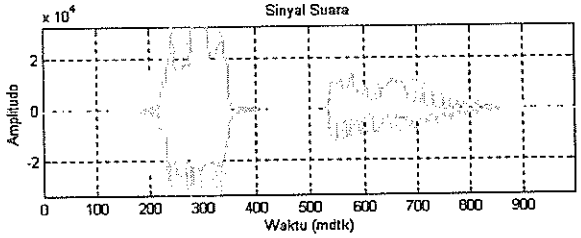
Gambar 3.20 Tampilan sinyal asli sampel suara "ada"

Tabel 3.3 Hasil pengujian dengan sampel suara "ada"

Sinyal	Interval waktu (milidetik)	Nilai *pitch* (Hz)	Suara	Keterangan
	**121** - 160 161 - 200 201 - 240 241 - 280 281 - **320**	235.2941 235.2941 242.4242 242.4242 242.4242	A	*Voiced*
	**321** - 360 361 - 400 401 - 440 441 - 480 481 - **520**	0 0 0 0 0	Mewakili konsonan d	*Unvoiced*
	**521** - 560 561 - 600 601 - 640 641 - 680 681 - 720 721 - 760 761 - 800 801 - **840**	242.4242 228.5714 222.2222 216.2162 216.2162 222.2222 228.5714 216.2162	a	*Voiced*

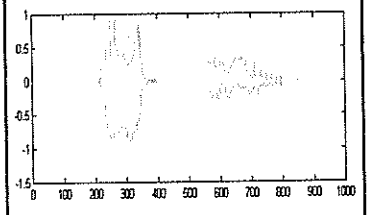
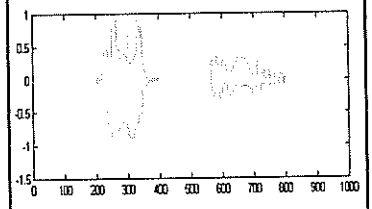
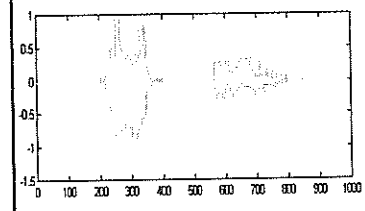
Tabel 3.3 menunjukkan bahwa sinyal suara "ada" pada saat interval waktu antara 121 – 320 dan 521 – 840 milidetik mempunyai nilai *pitch* antara 216.2162 – 242.4242 Hz dan menyuarakan vokal "a", sehingga termasuk *voiced*. Sedangkan pada interval waktu antara 321 – 520 milidetik mempunyai nilai *pitch* nol (0) dan menyuarakan mewakili konsonan "d", sehingga termasuk *unvoiced*.

Hasil pengujian dengan sampel suara "api"



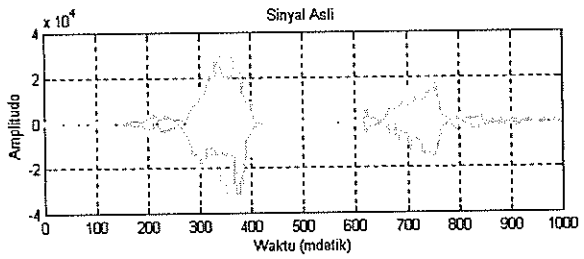
Gambar 3.21 Tampilan sinyal asli sampel suara "api"

Tabel 3.4 Hasil pengujian dengan sampel suara "api"

Sinyal	Interval waktu (milidetik)	Nilai *pitch* (Hz)	Suara	Keterangan
	**201** - 240 241 - 280 281 - 320 321 - 360 361 - 400 401 - 440 441 - 480 481 - **520**	216.2162 222.2222 222.2222 222.2222 222.2222 216.2162 210.5263 242.4242	A	*Voiced*
	**521 - 560**	0	Mewakili konsonan p	*Unvoiced*
	**561** - 600 601 - 640 641 - 680 681 - 720 721 - 760 761 - **800**	235.2941 228.5714 210.5263 200 195.122 101.2658	I	*Voiced*

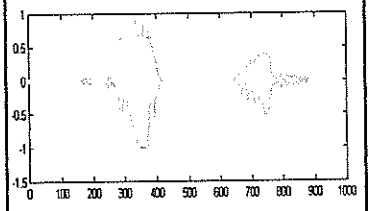
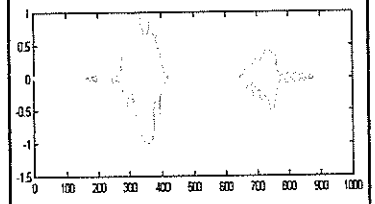
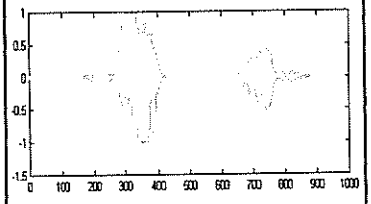
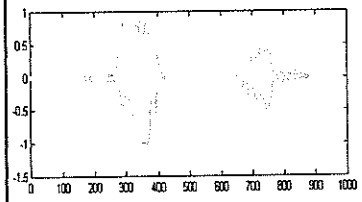
Tabel 3.4 menunjukkan bahwa sinyal suara "api" pada saat interval waktu antara 201 – 520 dan 561 – 800 milidetik mempunyai nilai *pitch* antara 101.2658 – 242.4242 Hz dan menyuarakan vokal "a" dan "i", sehingga termasuk *voiced.* Sedangkan pada interval waktu antara 521 – 560 milidetik mempunyai nilai *pitch* nol (0) dan menyuarakan mewakili konsonan "p", sehingga termasuk *unvoiced.*

Hasil pengujian dengan sampel suara "fakir"



Gambar 3.22 Tampilan sinyal asli sampel suara "fakir"

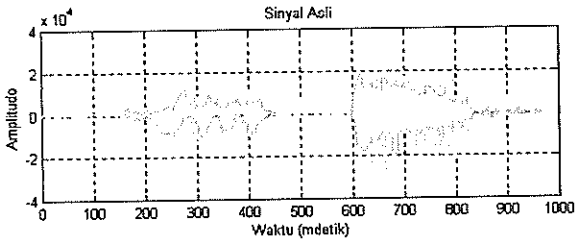
Tabel 3.5 Hasil pengujian dengan sampel suara "fakir"

Sinyal	Interval waktu (milidetik)	Nilai *pitch* (Hz)	Suara	Keterangan
	**201 - 240**	0	Mewakili konsonan F	*Unvoiced*
	**241** - 280 281 - 320 321 - 360 361 - 400 401 - 440 441 - **480**	228.5714 228.5714 235.2941 235.2941 235.2941 228.5714	a	*Voiced*
	**481** - 520 521 - 560 561 - 600 601 - **640**	0 0 0 0	Mewakili konsonan k	*Unvoiced*
	**641** - 680 681 - 720 721 - 760 761 - 800 801 -840 841 - **880**	242.4242 235.2941 210.3263 195.122 177.7778 177.7778	ir	*Voiced*

Tabel 3.5 menunjukkan bahwa sinyal suara "fakir" pada saat interval waktu antara 201 – 240 dan 481 – 640 milidetik mempunyai nilai *pitch* nol (0)

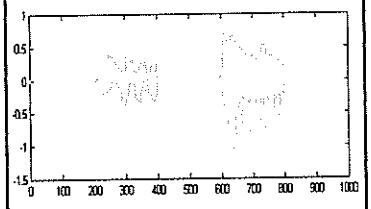
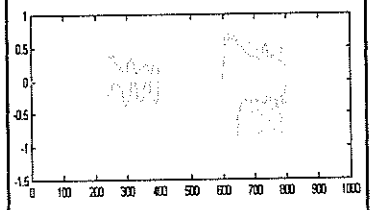
dan menyuarakan mewakili konsonan "f" dan "k", sehingga termasuk *unvoiced.* Sedangkan pada interval waktu antara 241 – 480 milidetik mempunyai nilai *pitch* antara 228.5714 – 235.2941 Hz, menyuarakan vokal "a" dan pada interval waktu antara 641 – 880 milidetik mempunyai nilai pitch antara 177.7778 – 242.4242 Hz, menyuarakan "ir" , sehingga termasuk *voiced.*

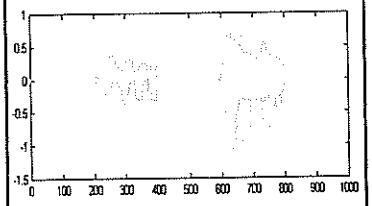
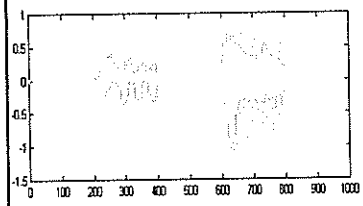
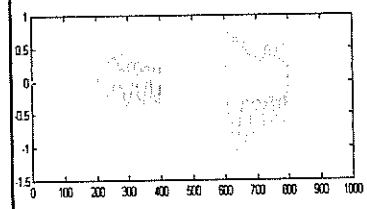
Hasil pengujian dengan sampel suara "gitar"



Gambar 3.23 Tampilan sinyal asli sampel suara "gitar"

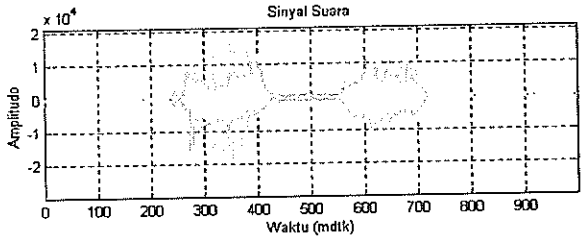
Tabel 3.6 Hasil pengujian dengan sampel suara "gitar"

Sinyal	Interval waktu (milidetik)	Nilai *pitch* (Hz)	Suara	Keterangan
	**161 - 200**	0	Mewakili konsonan G	*Unvoiced*
	**201** - 240 241 - 280 281 - 320 321 - 360 361 - **400**	228.5714 235.2941 235.2941 235.2941 235.2941	I	*Voiced*

Sinyal	Interval waktu (milidetik)	Nilai *pitch* (Hz)	Suara	Keterangan
	**401** - 440 441 - 480 481 - 520 521 - **560**	0 0 0 0	Mewakili konsonan t	*Unvoiced*
	**561** - 600 601 - 640 641 - 680 681 - 720 721 - 760 760 - **800**	228.5714 222.2222 222.2222 216.2162 205.1282 200	a	*Voiced*
	**801 - 840**	0	Mewakili konsonan r	*Unvoiced*

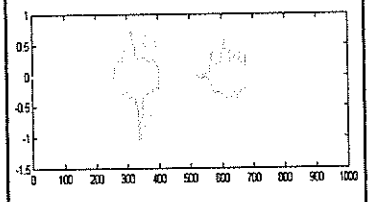
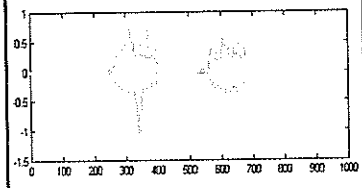
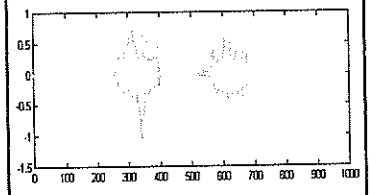
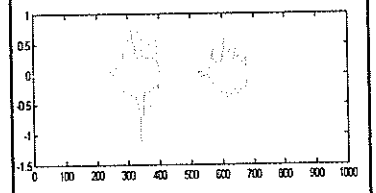
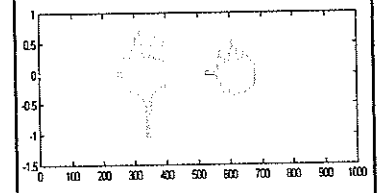
Tabel 3.6 menunjukkan bahwa sinyal suara "gitar" pada saat interval waktu antara 161 – 200, 401 - 560 dan 801 – 840 milidetik mempunyai nilai *pitch* nol (0) dan menyuarakan mewakili konsonan "g", "t" dan "r", sehingga termasuk *unvoiced.* Sedangkan pada interval waktu antara 201 – 400 dan 561 – 800 milidetik mempunyai nilai *pitch* antara 200 – 228.5714 Hz dan menyuarakan vokal "i" dan "a", sehingga termasuk *voiced.*

Hasil pengujian dengan sampel suara "kaset"



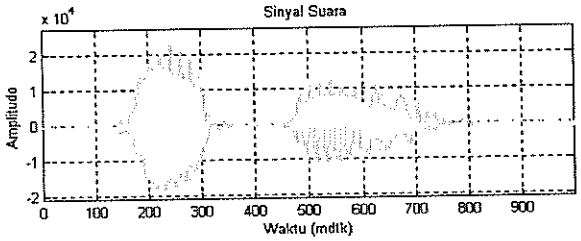
Gambar 3.24 Tampilan sinyal asli sampel suara "kaset"

Tabel 3.7 Hasil pengujian dengan sampel suara "kaset"

Sinyal	Interval waktu (milidetik)	Nilai *pitch* (Hz)	Suara	Keterangan
	**201 - 240**	0	Mewakili konsonan k	*Unvoiced*
	**241** - 280 281 - 320 321 - 360 361 - **400**	228.5714 228.5714 228.5714 222.2222	a	*Voiced*
	**401** - 440 441 - 480 481 - **520**	0 0 0	Mewakili konsonan s	*Unvoiced*
	**521** - 560 561 - 600 601 - 640 641 - **680**	235.2941 242.4242 235.2941 235.2941	e	*Voiced*
	**681 - 720**	0	Mewakili konsonan t	*Unvoiced*

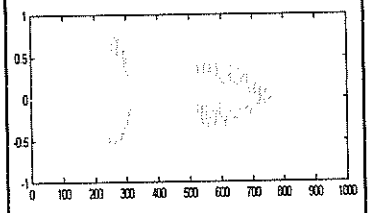
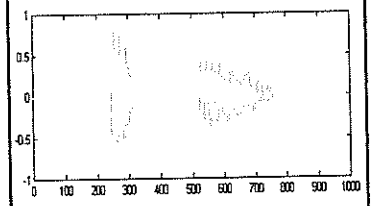
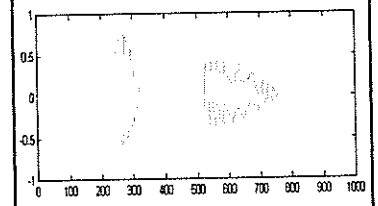
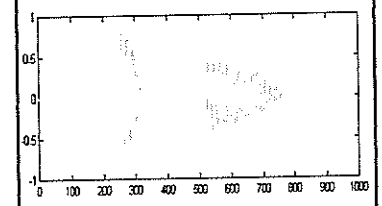
Tabel 3.7 menunjukkan bahwa sinyal suara "kaset" pada saat interval waktu antara 200 - 240, 401 - 520 dan 681 – 720 milidetik mempunyai nilai *pitch* nol (0) dan menyuarakan mewakili konsonan "k", "s" dan "t", sehingga termasuk *unvoiced.* Sedangkan pada interval waktu antara 241 – 400 dan 521 – 680 milidetik mempunyai nilai *pitch* antara 222.2222 – 242.4242 Hz dan menyuarakan vokal "a" dan "e", sehingga termasuk *voiced.*

Hasil pengujian dengan sampel suara "kubu"



Gambar 3.25 Tampilan sinyal asli sampel suara "kubu"

Tabel 3.8 Hasil pengujian dengan sampel suara "kubu"

Sinyal	Interval waktu (milidetik)	Nilai *pitch* (Hz)	Suara	Keterangan
	**121** - 160 161 - 200 201 - **240**	0 0 0	Mewakili konsonan k	*Unvoiced*
	**241** - 280 281 - **320**	242.4242 242.4242	u	*Voiced*
	**401** - 440 441 - 480 481 - **520**	0 0 0	Mewakili konsonan b	*Unvoiced*
	**521** - 560 561 - 600 601 - 640 641 - 680 681 - 720 721 - **760**	242.4242 228.5714 210.5263 205.1282 205.1282 94.1176	u	*Voiced*

Tabel 3.8 menunjukkan bahwa sinyal suara "kubu" pada saat interval waktu antara 141 – 240 dan 401 - 520 milidetik mempunyai nilai *pitch* nol (0) dan

menyuarakan mewakili konsonan "k" dan "b", sehingga termasuk *unvoiced.* Sedangkan pada interval waktu antara 241 – 320 dan 521 – 760 milidetik mempunyai nilai *pitch* antara 94.1176 – 242.4242 Hz dan menyuarakan vokal "u", sehingga termasuk *voiced.*

Bab 4
Pengolahan Suara dalam Kawasan Waktu

Tujuan akhir pengolahan sinyal ucapan adalah untuk mendapatkan representasi yang lebih berguna dari informasi yang dibawa oleh sinyal ucapan. Representasi tepat yang kita diperlukan diberikan oleh bagian-bagian informasi dalam sinyal ucapan. Contohnya, tujuan dari pengolahan sinyal mungkin untuk memudahkan penentuan apakah suatu bagian bentuk gelombang berkorespondensi dengan ucapan atau tidak. Kita dapat mengklasifikasikan bagian suatu sinyal menjadi 3 yaitu ucapan yang disuarakan (*voiced speech*), ucapan tidak yang disuarakan (*unvoiced speech*), atau kesunyian (*silence/background noise*). Dalam beberapa kasus, suatu representasi harus menghilangkan informasi yang tidak relevan dan menempatkan informasi yang diinginkan daripada representasi yang mendetail namun menahan semua informasi yang melekat padanya. Dalam kasus lain (contohnya transmisi digital) bisa membutuhkan representasi yang paling akurat dari sinyal ucapan yang dapat diperoleh dengan satu set pembatas yang diberikan.

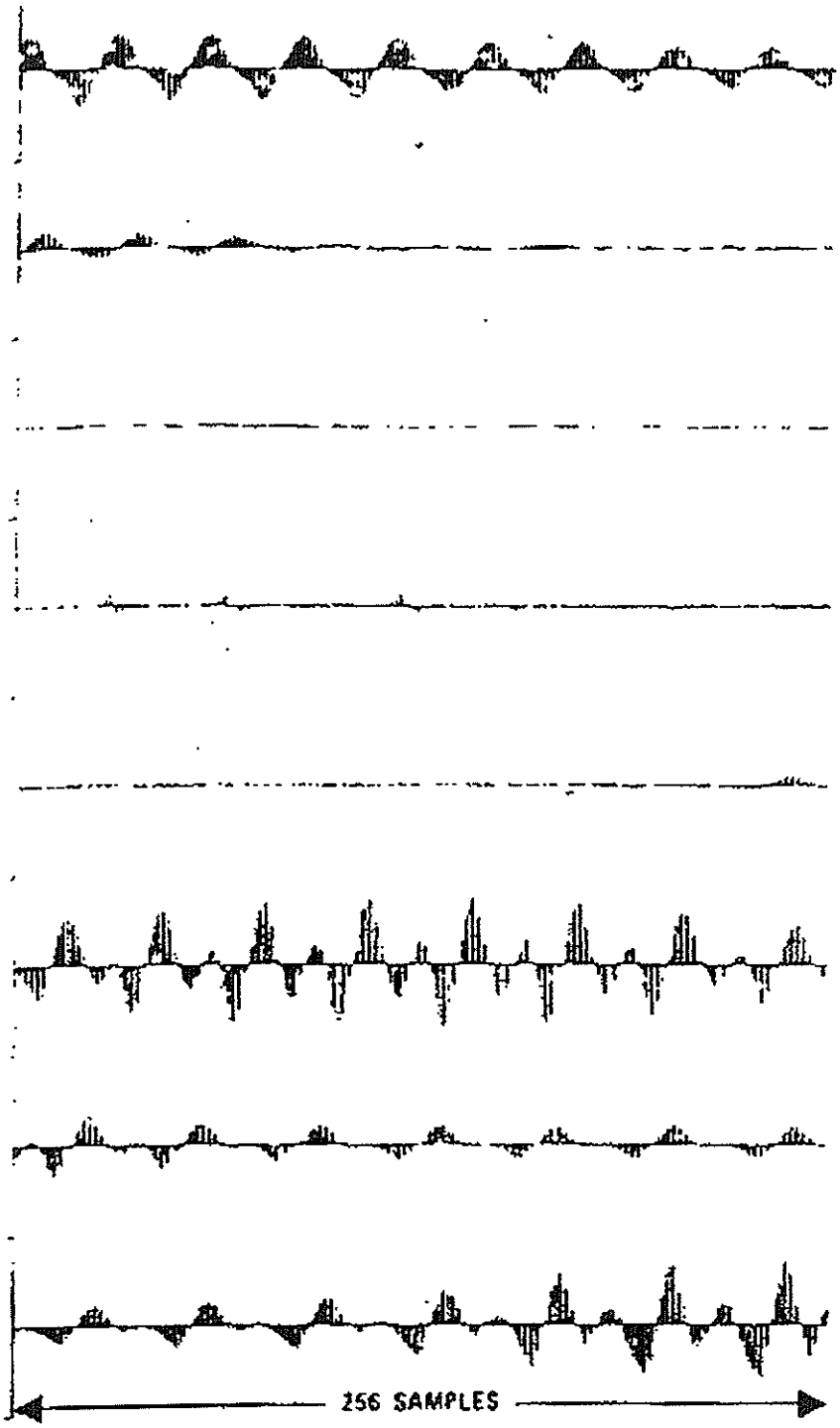
Dalam bab ini kita memperhatikan suatu set teknik pengolahan dengan metode *time-domain*. Dari sini kita paham bahwa metode pengolahan melibatkan secara langsung bentuk gelombang sinyal ucapan.

Beberapa contoh dari representasi sinyal ucapan dalam pengukuran *time-domain* mengandung kecepatan rata-rata *zero-crossing*, energi dan fungsi auto korelasi. Representasi ini sangat menarik karena pengolahan digital yang diperlukan sangat sederhana untuk diimplementasikan dan hasil representasinya menyajikan suatu basis yang berguna untuk mengkalkulasi sifat/cirri sinyal ucapan.

Beberapa metode penting akan dibahas sebagai contoh dari representasi *time-domain* umum dan pendekatan pemrosesan/pengolahan. Akhirnya kita membahas beberapa skema untuk mengkalkulasi sifat bentuk gelombang ucapan seperti klasifikasi *voiced/ unvoiced*, *pitch* (tingi-rendah nada ucapan) dan intensitas dari representasi *time-domain*.

4.1 Pengolaha ucapan dalam ranah waktu

Gambar 4.1 menunjukkan sebuah urutan dari beberapa sample (8000sample/detik) yang merepresentasikan sebuah sinyal ucapan. Hal ini jelas menunjukkan bahwa sifat dari sinyal ucapan berubah dalam waktu. Contohnya, eksitasinya berubah antara –*voiced* dan *unvoiced speech*, ada variasi yang signifikan pada amplitudo puncak sinyal dan ada variasi yang jelas pada frekuensi dasar dalam *voiced regions*. Pada kenyataannya, variasi ini sangat jelas dalam plot bentuk gelombang yang menjelaskan bahwa teknik pemrosesan kawasan waktu sederhana harus dapat menyediakan representasi yang bermanfaat dari sifat sinyal seperti intensitas, excitation mode, pitch, dan mungkin juga parameter vocal tract.



Gambar 4.1 Sampel dari sinyal suara (frekuensi sampling 8 kHz)

Asumsi yang perlu diperhatikan dalam speech processing adalah sifat-sifat sinyal yang relative berubah terhadap waktu secara perlahan. Asumsi ini mengarah pada metode pemrosesan "short-time' dalam segmen pendek dari suara yang tersustain. Hal ini adalah sering berulang (periodik) sesuai yang diharapkan. Segmen pendek ini sering kali saling meliputi dengan yang lainnya dimana kadang disebut frame analisis. Hasil proses pada tiap frame dapat dalam satu

angka atau satu set angka. Maka dari itu, pemrosesan seperti itu menghasilkan urutan waktu independent yang baru yang dapat menyediakan representasi sinyal suara.

Kebanyakan teknik proses short-time yang akan dibahas adalah seperti representasi short-time Fourier, yaitu

$$Q_n = \sum_{m=-\infty}^{\infty} T[x(m)]w(n-m) \tag{4.1}$$

Sinyal suara yang telah difilter untuk mendapatkan frekuensi yang diharapkan adalah subyek transmisi, $T[\ ]$, dapat linier atau nonlinier, dan dapat bergantung pada beberapa parameter tambahan. Hasil urutan kemudian dikalikan dengan urutan jendela yang terletak pada waktu corresponding sampai indeks sample n. Hasil kali kemudian dijumlahkan dengan semua nilai nonzero. Biasanya urutan jendela akan berupa durasi tertentu, walau ini bukan kasus yang sering. Nilai Q_n adalah urutan dari nilai bobot local rata-rata dari urutan $T[x(m)]$.

Energi short-time dari sinyal adalah contoh yang sederhana dimana mengilustrasikan ide yang dibahas di atas. Energi sinyal diskrit didefinisikan sebagai

$$E = \sum_{m=-\infty}^{\infty} x^2(m) \tag{4.2}$$

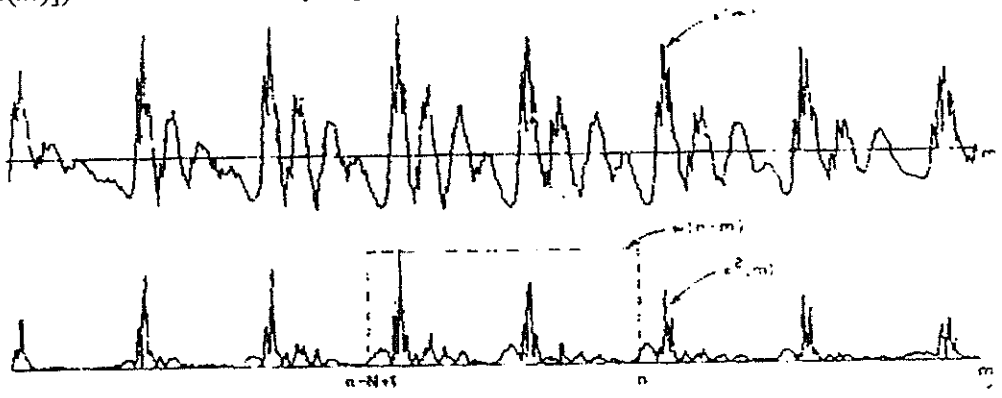
Beberapa kuantitas memiliki arti kecil atau utilitas untuk ucapan jika hal itu memberikan informasi yang sedikit tentang sifat time-independent dari sinyal ucapan. Definisi sederhana energi shot-time adalah

$$E_n = \sum_{m=n-N+1}^{n} x^2(m) \tag{4.3}$$

Dimana energi short-time pada sample ke-n merupakan penjumlahan kuadrat N sample $n-N+1$ melalui sampai n.

$$\begin{aligned} w(n) &= 1 \qquad 0 \le n < N\text{-}1 \\ &= 0 \qquad n \text{ yang lain} \end{aligned} \tag{4.4}$$

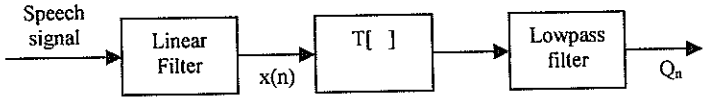
Gambar 4.2 menggambarkan perhitungan urutan energi short-time. Perhatikan bahwa jendela bergeser sepanjang urutan dari nilai kuadrat (pada umumnya, $T[x(m)]$) memilih interval yang digunakan untuk perhitungan.



Gambar 4.2 pengilustrasian dari perhitungan energi short-time

Persamaan 4.1 terlihat dalam bentuk konvolusi diskrit dari jendela $w(n)$ dengan urutan $T[x(n)]$. Lalu Q_n dapat dijelaskan sebagai keluaran dari sistem linier

time-invarian dengan respon impuls $h(n) = w(n)^2$. Ini digambarkan dalam gambar 4.3.



Gambar 4.3 gambaran umum dari prinsip analisis short-time

4.2 Energi short-time dan magnitudo rata-ratanya

Kita telah amati bahwa amplitudo sinyal ucapan bervariasi terhadap waktu. Amplitudo segment unvoiced umumnya lebih rendah dari amplitudo segment voiced. Energi short-time sinyal ucapan menggambarkan variasi amplitudo ini. Definisi energi short-time adalah:

$$E_n = \sum_{m=-\infty}^{\infty} [x(m)w(n-m)]^2 \tag{4.5}$$

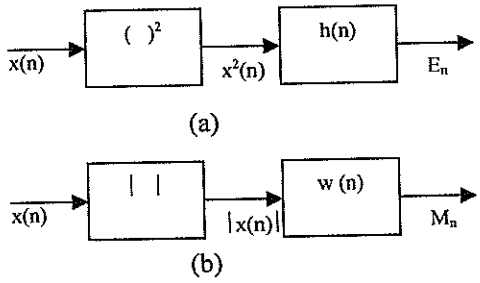
Persamaan ini bisa ditulis sebagai

$$E_n = \sum_{m=-\infty}^{\infty} x^2(m) \cdot h(n-m) \tag{4.6}$$

dimana

$$h(n) = w^2(n) \tag{4.7}$$

Persamaan 4.6 dapat diinterpretasikan seperti yang digambarkan oleh gambar 4.4a. Dimana, sinyal $x^2(n)$ difilter oleh filter linear dengan respon impuls $h(n)$ seperti yang diberikan oleh persamaan (4.7)



Gambar 4.4 Blok diagram (a) energi short-time (b) magnitudo rata-rata short-time

Pemilihan respon impuls $h(n)$ menentukan sifat dasar dari representasi energi short-time. Untuk melihat pengaruh jendela pada energi short-time, kita amati bahwa jika $h(n)$ pada persamaan 4.6 sangat panjang maka E_n akan sedikit mengalami perubahan terhadap waktu. Suatu jendela harus ekivalen dengan tapis lolos lebar pita yang sangat kecil. Namun keluarannya bisa tidak konstan.

Efek dari jendela pada representasi energi yang terikat waktu dapat diilustrasikan dengan membahas sifat dari dua jendela representatif, yaitu jendela persegi.

$$\begin{aligned} h(n) &= 1 \qquad 0 \le n \le N-1 \\ &= 0 \qquad \text{untuk yang lain} \end{aligned} \tag{4.8}$$

dan jendela Hamming

$$h(n) = 0.54 - - 0.46\cos(2\pi n/(N-1)), 0 \le n \le N-1$$

$$= 0 \qquad \text{untuk yang lain} \quad (4.9)$$

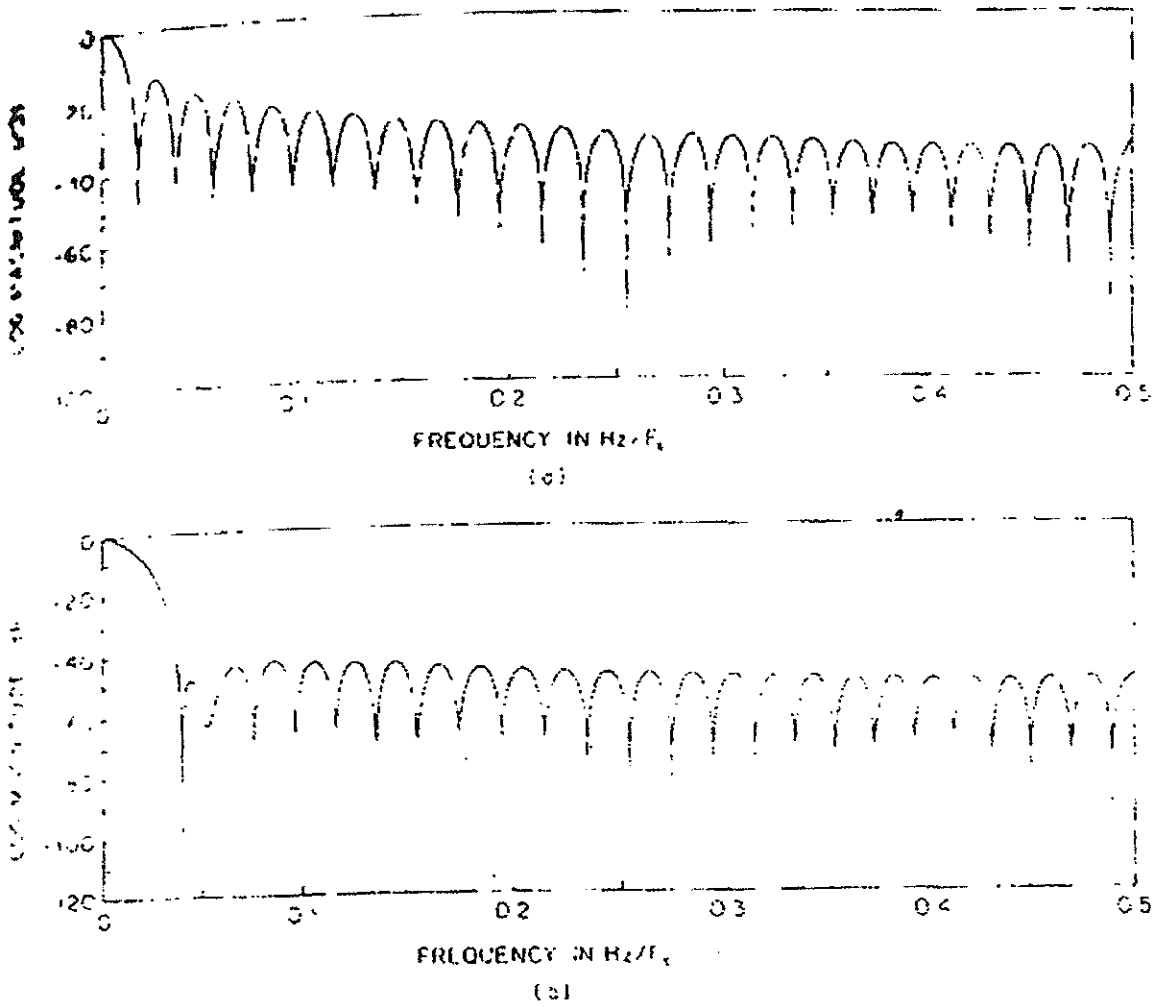
Jendela persegi pada persamaan 4.3 berkorespondensi untuk menggunakan bobot yang equal dengan sample pada interval (n – N + 1) sampai n. Respon frekuensi jendela persegi (respon impuls pada persamaan 4.8) ditunjukkan dengan

$$H(e^{j\Omega T}) = \frac{\sin(\Omega NT/2)}{\sin(\Omega T/2)} e^{-j\Omega T(N-1)/2} \quad (4.10)$$

Logaritma magnitude dari respon ini ditunjukkan dalam gambar 4.5a untuk 51 sample window ($N = 51$). Catat bahwa zero pertama persamaan 4.10 terjadi pada frekuensi analog

$$F = F_s/N \quad (4.11)$$

Dimana F_s = 1/T adalah frekuensi sampling. Secara nominal nilai ini adalah frekuensi cuttoff dari LPF yang berkorespondensi dengan jendela persegi. Respon frekuensi dari 51 titik jendela Hamming ditunjukkan dalam gambar 4.5b. Terlihat bahwa bandwidth dari Hamming window dua kali bandwidth sebuah jendela persegi pada panjang yang sama. Hamming window juga memberikan atenuasi yang lebih besar di luar passband daripada jendela persegi. Atenuasi kedua jendela ini tidak terikat pada durasinya. Memperbesar panjangnya, N, akan menurunkan banwidth. Jika N terlalu kecil maka E_n akan berfluktuasi sangat cepat. Jika N terlalu besar, maka E_n akan berubah secara lambat dan tidak akan cukup merefleksikan perubahan sifat sinyal ucapan. Namun durasi periode pitch bervariasi dari 20 sample (pada frekuensi sampling 10kHz) untuk wanita atau anak, sampai 250 sample untuk suara pria paling rendah. Dengan demikian nilai N yang praktis sekitar 100-200 untuk sampling 10kHz (yaitu durasi 10-20 ms).

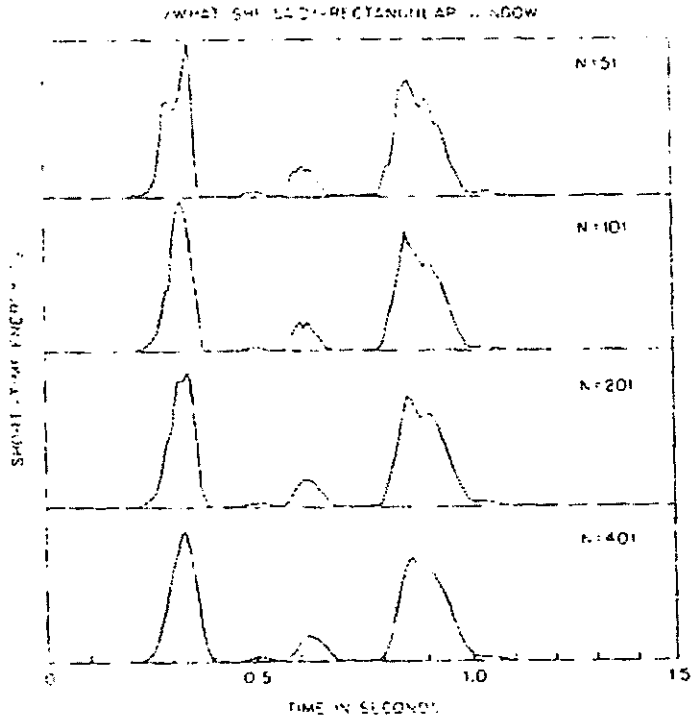


Gambar 4.5 Transformasi Fourier (a) rectangular window (b) Hamming window

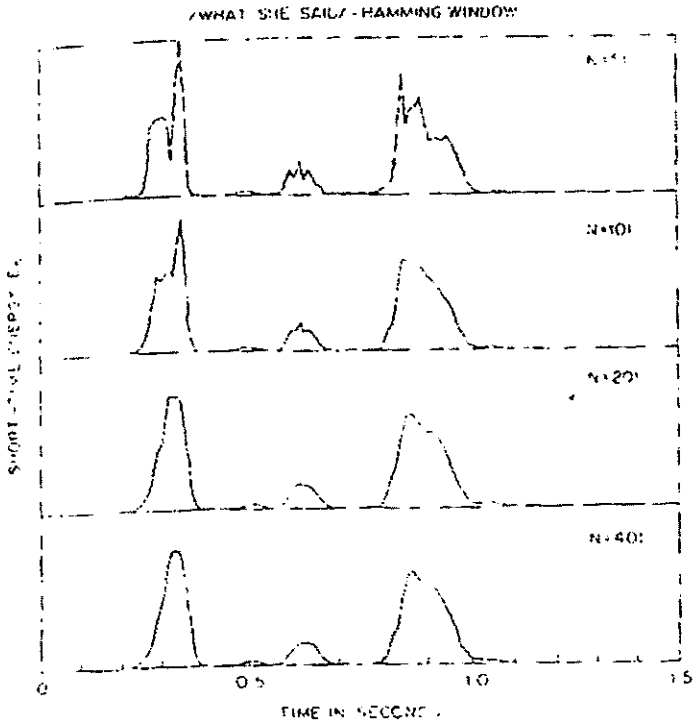
Gambar 4.6 dan 4.7 menunjukkan efek dari pengubahan durasi window (untuk rectangular dan Hamming window) pada perhitungan energi dari ucapan si pembicara. Terlihat bahwa jika N naik maka energi menjadi lebih halus untuk kedua jendela tsb.

$$M_n = \sum_{m=-\infty}^{\infty} |x(m)| w(n-m) \tag{4.12}$$

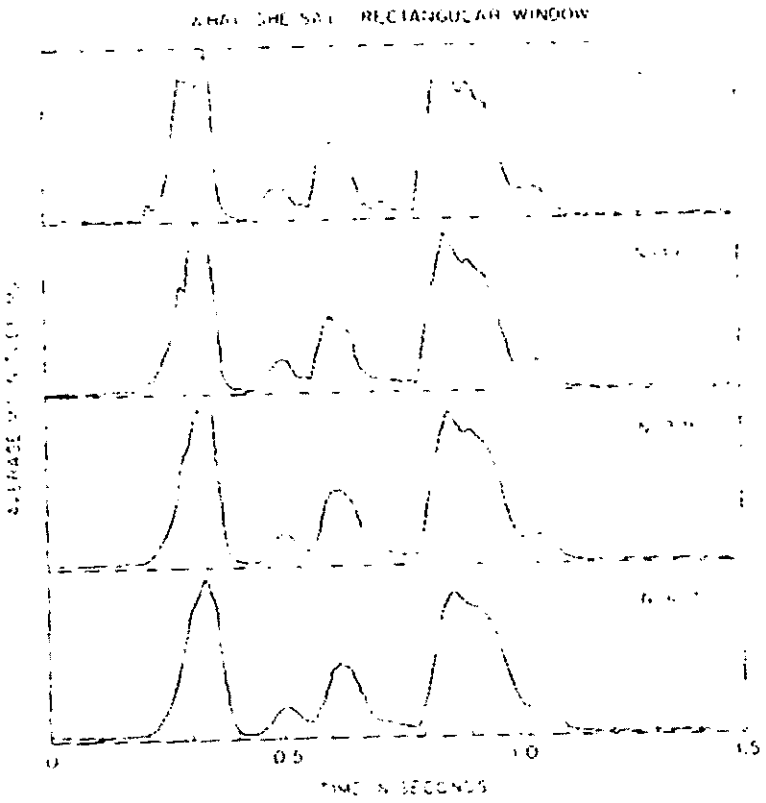
dimana bobot jumlah nilai mutlak sinyal dikomputasi dari jumlah kuadrat. Gambar 4.4b menunjukkan bagaimana persamaan 4.12 dapat diimplementasikan sebagai operasi filter linier pada $|x(n)|$.



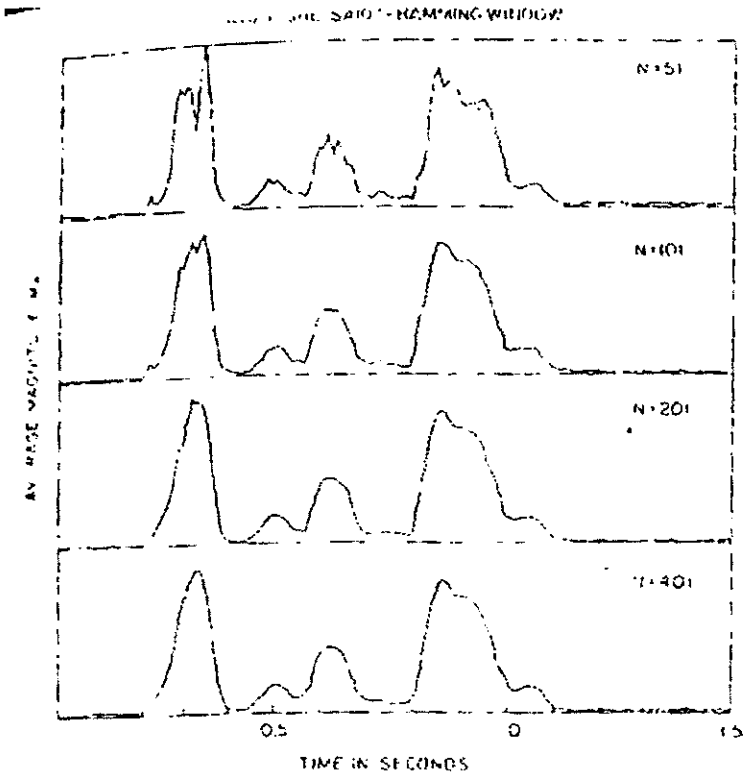
Gambar 4.6 Fungsi energi short-time untuk panjang rectangular windows yang bervariasi.



Gambar 4.7 Fungsi energi short-time untuk panjang Hamming windows yang bervariasi.



Gambar 4.8 Fungsi magnitude rata-rata untuk rectangular window dengan panjang yang bervariasi



Gambar 4.9 Fungsi magnitude rata-rata untuk Hamming window dengan panjang yang bervariasi

Gambar 4.8 dan 4.9 menunjukkan magnitude rata-rata yang berhubungan dengan gambar 4.6 dan 4.7. Untuk penghitungan magnitude rata-rata dari persamaan 4.12, range dinamis (rasio dari maksimum dan minimum) merupakan akar range dinamis untuk perhitungan standar dari energi. Lalu perbedaan level antara voiced dengan unvoiced regions tidak ditegaskan seperti energi short-time.

Bandwidth fungsi energi dan magnitude rata-rata adalah bandwidth LPF, maka perlu disampling dengan frekuensi sinyal ucapan.Contoh, untuk durasi window 20 ms, maka cukup disampling dengan kecepatan 100 sample/detik. Artinya banyak informasi yang hilang dalam memperoleh representasi short-time ini.

Diperlukan filter digital untuk penghalusan. Kita dapat merancang LPF dengan filter FIR atau IIR. Filter FIR memiliki keuntungan dengan keluaran yang mudah dikomputasi pada kecepatan sampling yang lebih rendah dari masukannya dengan menggeser window lebih dari 1 sample di antara penghitungannya. Contoh, jika sinyal ucapan disampling pada 10000 sample/detik, dan durasi window 20 ms (200 sample), energi shrt-time dapat dikomputasi pada kecepatan sampling sekitar 100 sampling/detik atau setiap 100 sample pada kecepatan samplind masukkannya. Contoh sederhana dari sebuah jendela

$$\begin{aligned} h(n) &= a^n \qquad n \geq 0 \\ &= 0 \qquad n < 0 \end{aligned} \tag{4.13}$$

Nilai 0 < a < 1memberikan window dimana durasi efektifnya dapat disesuaikan. Korespondensi transformasi-z dari window adalah

$$H(n) = \frac{1}{1 - az^{-1}} \qquad |z| > |a| \tag{4.14}$$

Frekuensi responnya $H(e^{jwT})$, memiliki sifat lowpass yang diinginkan. Filter seperti ini dapat diimplementasikan dengan persamaan sederhana.

$$E_n = aE_{n-1} + x^2(n) \tag{4.16}$$

Dan magnitude rata-rata

$$M_n = aM_{n-1} + |x(n)| \tag{4.16}$$

Untuk menggunakan persamaan 4.15 dan 4.16, kuantitas E_n dan M_n harus dikomputasikan pada tiap sample dari sinyal masukan, bahkan sampling yang rendah sudah cukup. Bagaimanapun juga ketika sampling yang rendah sudah cukup, metode nonrekursif hanya sedikit melibatkan aritmatika. Jendela pada persamaan 4.8 dan 4.9 berkorespondensi dengan filter kausal. Kerena keduanya simetris, keduanya memiliki fasa linier dengan delay (N-1)/2 samples.

4.3 Short-time average zero-crossing rate

Dalam konteks sinyal diskrit, zero-crossing terjadi jika sample suksesif memiliki perbedaan simbol yang bersifat aljabar. Contohnya, sinyal sinus dengan frekuensi F_0 yang disampling dengan frekuensi F_s memiliki F_s / F_0 sample per siklusnya. Tiap siklus memliki 2 pemotongan nilai nol (zero crossing) sehingga

$$Z = 2\,F_0/F_s \qquad \text{crrossings/sample} \tag{4.17}$$

Lalu, kecepatan rata-rata zero crossing memberikan jalan untuk memperkirakan frekuensi sinyal sinus.

Sinyal ucapan merupakan sinyal pita lebar dan penafsiran dari kecepatan rara-rata zero-crossingnya kurang lebih tepat. Bagaimanapun, perkiraan yang kasar mengenai sifat spektral sinyal dapat diperoleh dengan representasi berdasarkan kecepatan rata-rata zero-crossing short-time.

$$Z_n = \sum_{m=-\infty}^{\infty} |\text{sgn}[x(m)] - \text{sgn}[x(m-1)]| w(n-m) \qquad (4.18)$$

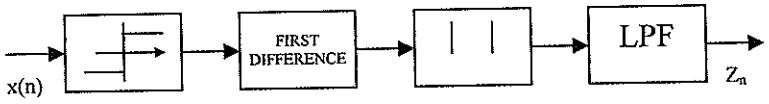
dimana

$$\begin{aligned} sgn[x(n)] &= 1 \qquad x(n) \geq 0 \\ &= -1 \qquad x(n) < 0 \end{aligned} \qquad (4.19)$$

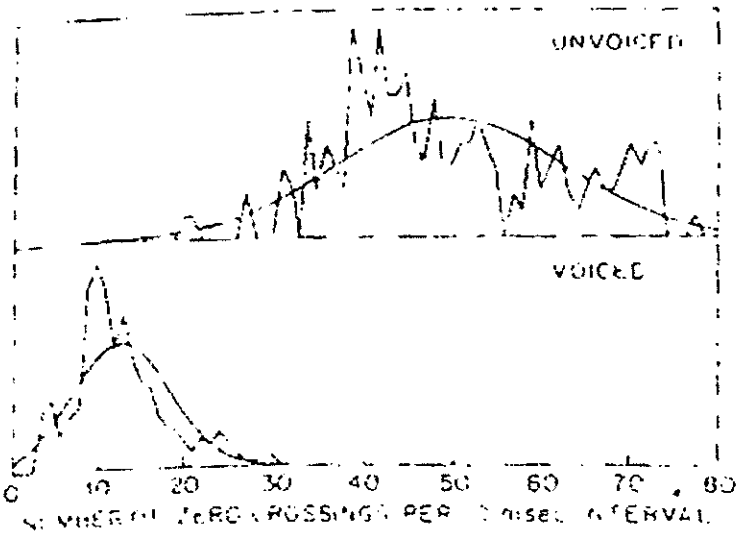
dan

$$\begin{aligned} w(n) &= \frac{1}{2N} \qquad 0 \leq n \leq N\text{-}1 \\ &= 0 \qquad \text{untuk yang lain} \end{aligned} \qquad (4.20)$$

Gambar 4.10 menunjukkan operasi dalam persamaan 4.18. Hal ini menunjukkan bahwa kecepatan rata-rata zero-crossing short-time memiliki sifat umum yang sama dengan energi dan magnitudo rata-rata short-time. Bagaimanapun, persamaan 4.18 dan gambar 4.10 membuat perhitungan Z_n terlihat lebih kompleks dari kenyataannya. Hal ini memerlukan pengecekan sample dalam pasangan untuk menjelaskan bahwa terjadi zero-crossing dan rata-ratanya dihitung di atas N sample yang berurutan. Delay yang ada dapat diabaikan.



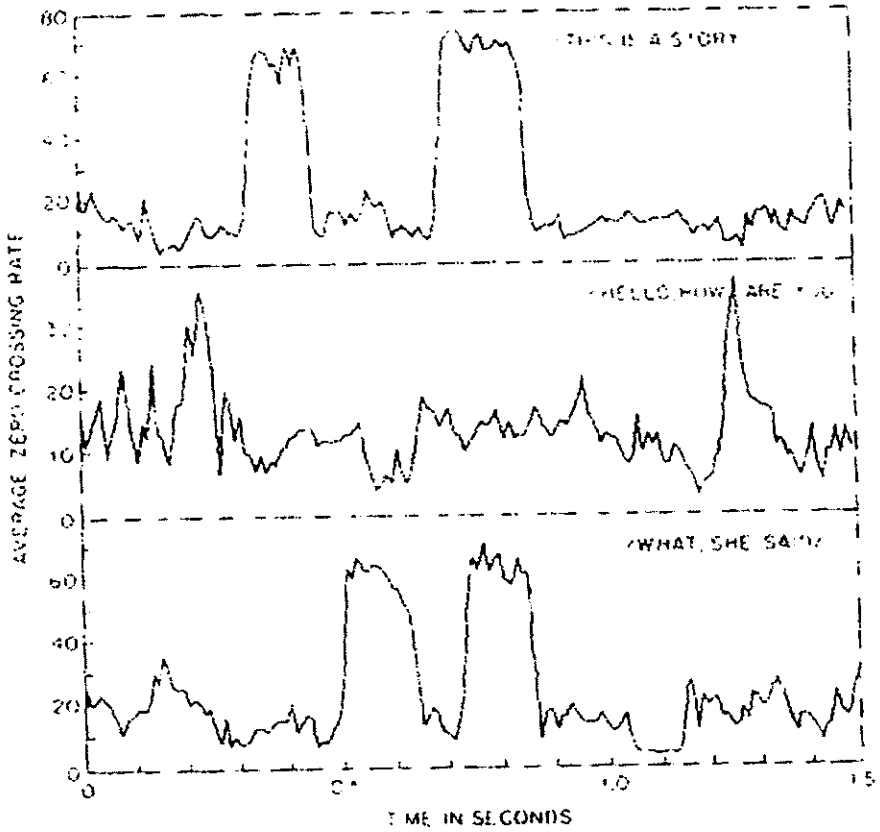
Gambar 4.10 Blok diagram dari rata-rata zero-crossing short-time



Gambar 4.11 Distribusi zero-crossing untuk unvoiced dan voiced speech

Model dari pembentukan ucapan adalah sbb; energi dari voiced speech terkonsentrasi di sekitar 3 kHz karena spektrum dari fall-off diperkenalkan oleh

gelombang glottai, sebagaimana untuk unvoiced speech, sebagian besar energi terdapat pada frekuensi tinggi. Karena frekuensi tinggi menyebabkan jumlah zero-crossing yang besar dan frekuensi rendah menyebabkan jumlah zero-crossing yang kecil pula, maka terdapat korelasi antara kecepatan zero-crossing dengan distribusi energi dalam frekuensi. Jika jumlah zero-crossing tinggi maka sinyal ucapannya adalah unvoiced, dan jika zero-crossing rendah, maka sinyal ucapannya voiced. Gambar 4.11 menunjukkan histogram jumlah rata-rata dari zero crossing (di atas 10 ms) unvoiced speech dan voiced speech. Catat bahwa kurva Gaussian memiliki distribusi yang baik. Nilai rata-rata jumlah zero crossing short-time adalah 49 per 10 ms untuk unvoiced dan 14 per 10 ms untuk voiced speech.



Gambar 4.12 Jumlah rata-rata zero-crossing untuk tiga pengucapan kata yang berbeda

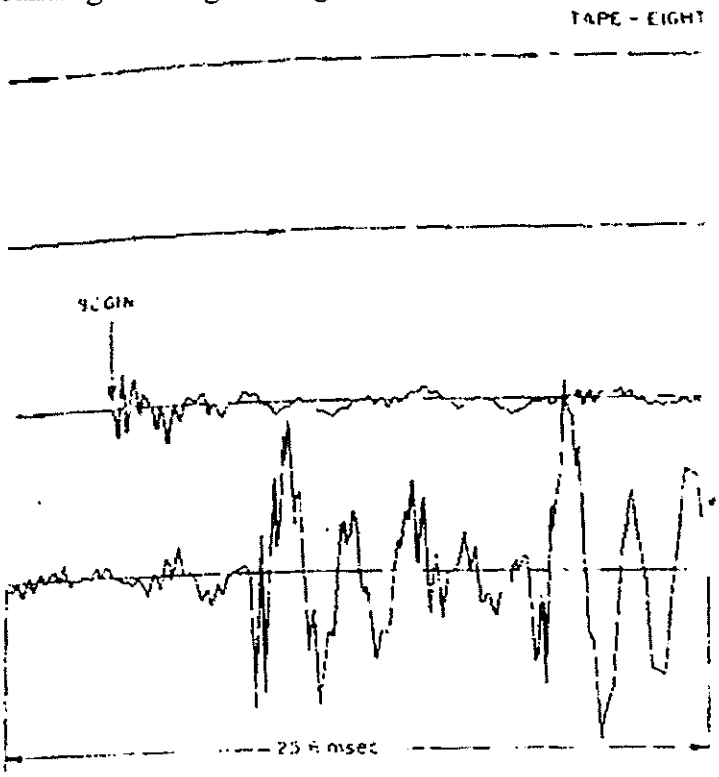
Gambar 4.12 menunjukkan contoh dari pengukuran jumlah rata-rata zero-crossing. Terdapat 150 sample pada frekuensi sampling 10kHz dan keluarannya dihitung 100 kali/detik (window bergerak tiap 100 sample). Jumlah zero-crossing sangat dipengaruhi oleh dc offset pada konverter analog ke digital. Suara dengung 60 Hz dan noise yang dapat muncul dalam proses pengubahan ke digital. Contohnya BPF lebih sering dipakai daripada LPF sebagai filter anti aliasing untuk menghilangkan komponen DC dan 60 hz pada sinyal ucapan.

4.4 Speech VS Perbedaan Keheningan dengan Menggunakan Energi dan Zero-Crossing

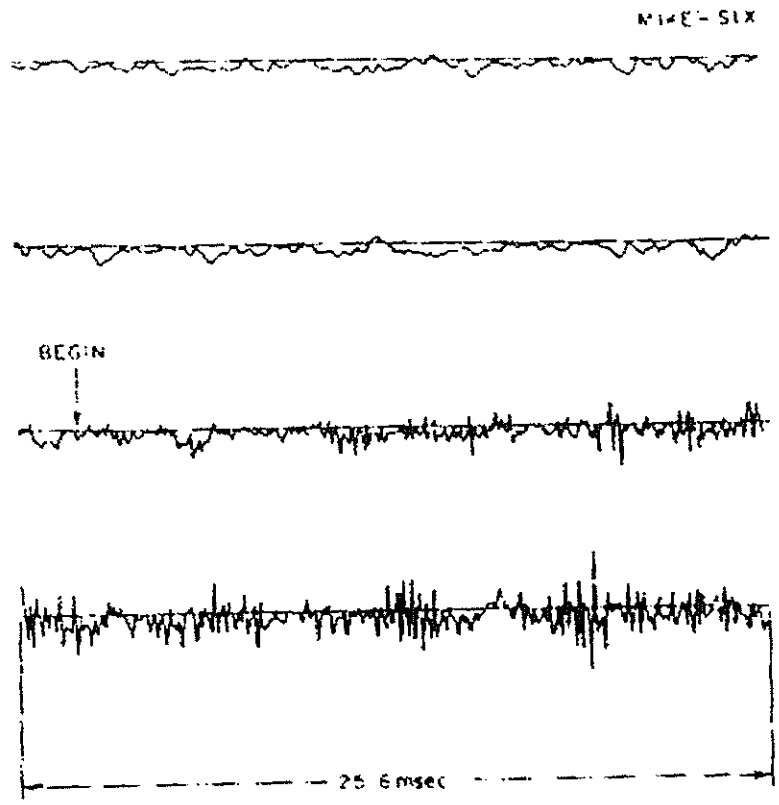
Penentuan awal dan akhir sinyal dalam background noise adalah sangat pentingdalam pengolahan suara. Dalam pengenalan kata secara otomatis sangat

penting untuk memisahkan daerah-daerah sinyal yang berkorespondensi dengan kata demi kata yang diucapkan oleh seseorang. Skema pengalokasian awal dan akhir sinyal dapat digunakan untuk mengeliminasi perhitungan yang signifikan pada sistem nonreal-time dengan membuatnya menjadi mungkin untuk memproses bagian sinyal yang hanya berhubungan dengan kata/ucapan.

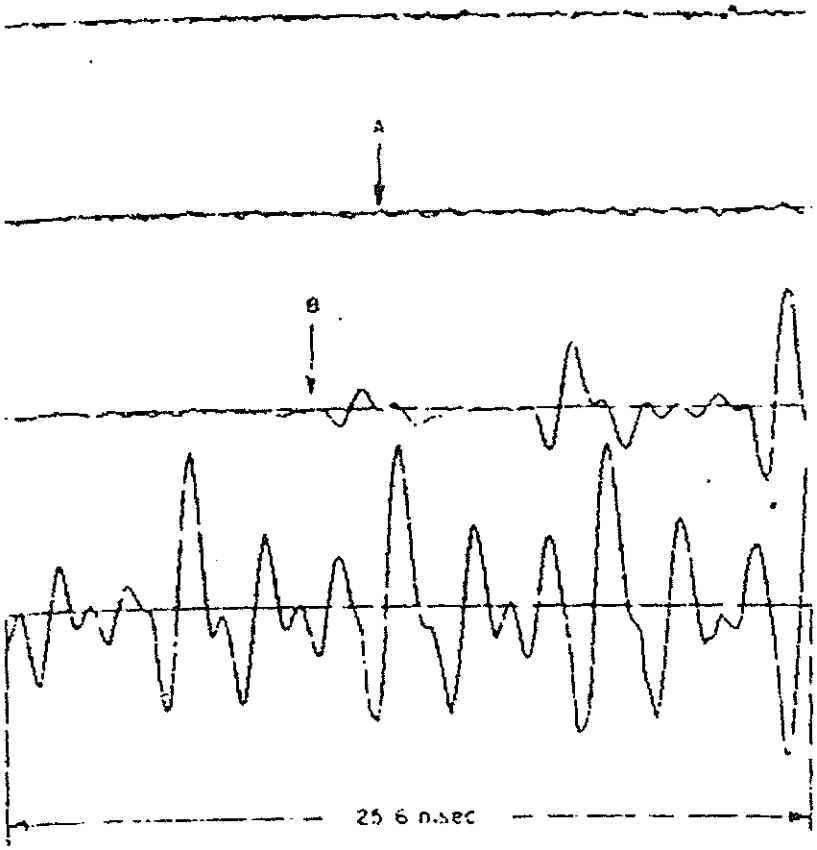
Membedakan ucapan dengan noise merupakan masalah yang tidak sepele. Kecuali dalam lingkungan yang memiliki SNR yang tinggi (contoh dalam dapur rekaman). Untuk lingkungan spt ini, energi dari level sinyal terendah pun jauh lebih tinggi dibandingkan dengan energi latar noisenya.



Gambar 4.13 Bentuk gelombang awal sinyal kata "eight"



Gambar 4.14 Bentuk gelombang awal sinyal kata "six"



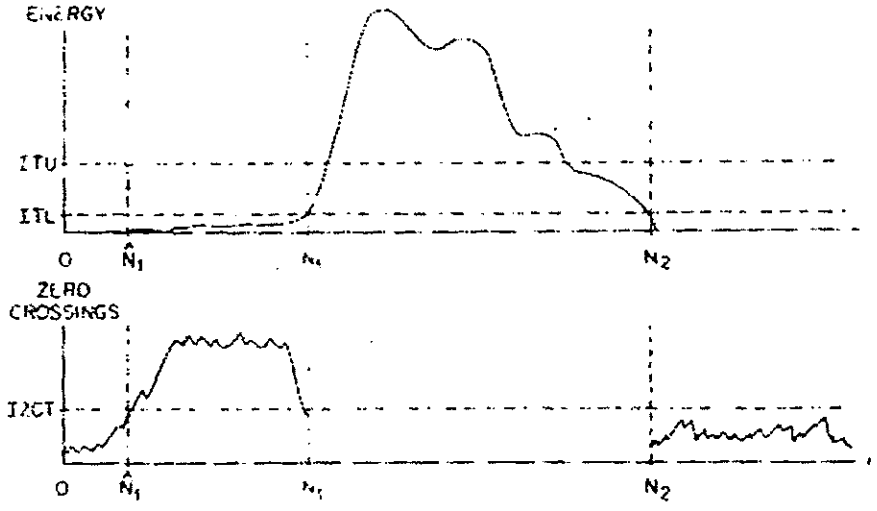
Gambar 4.15 Bentuk gelombang awal sinyal kata "four"

Algoritma yang dibahas pada bagian ini berdasarkan 2 pengukuran kawasan waktu yang sederhana yaitu jumlah energi dan zero-crossing. Gambar 4.13 memperlihatkan suatu contoh (awal dari ucapan "eight") dimana noise latarnya mudah dihilangkan. Dalam kasus ini perubahan radikal bentuk gelombang energi noise latar dengan ucapan menjadi tanda awal urutan sinyal ucapan. Gambar 4.14 menunjukkan contoh lain (awal dari ucapan "six") dimana sangat mudah untuk melokasikan awal ucapannya. Isi dari frekuensi ucapan sangat berbeda dengan noise latarnya yang ditunjukkan dengan kenaikan yang tajam dari zero-crossing bentuk gelombangnya.

Gambar 4.15 memberi contoh sinyal dimana sulit untuk menentukan awal ucapan dari kata "four". Karena awal ucapannya dimulai dengan energi yang rendah dari bunyi desah /f/. Dengan pembesaran gambar didapat titik B merupakan awal ucapan, namun sebenarnya titik awalnya adalah A. Berikut ini merupakan bunyi desah yang sulit untuk menentukan awal dan akhir ucapan:

1. Bunyi desah lemah (/f/, /th/, /h/) pada awal atau akhir ucapan
2. Weak plosive bursts (/p/, /t/, /k/) pada awal atau akhir ucapan
3. Bunyi sengau di akhir ucapan
4. Bunyi desah yang melemah di akhir kata
5. Suara vocal yang membekas di akhir ucapan

Sehubungan dengan kesulitan di atas, representasi jumlah energi dan zero-crossing dapat dikombinasikan untuk membentuk algoritma untuk menentukan awal dan akhir suatu sinyal ucapan. Suatu algoritma khusus telah digunakan oleh Rabiner dan Sambur dalam konteks suatu sistem pengenalan ucapan kata terisolasi . Dalam sistem ini, speaker mengucapkan sebuah kata dalam perekaman dengan waktu yang sudah ditentukan. Keseluruhan interval disampling dan disimpan untuk pemrosesan. Tujuan algoritma ini adalah untuk mencari awal dan akhir ucapan sehingga noise latar dapat diabaikan.



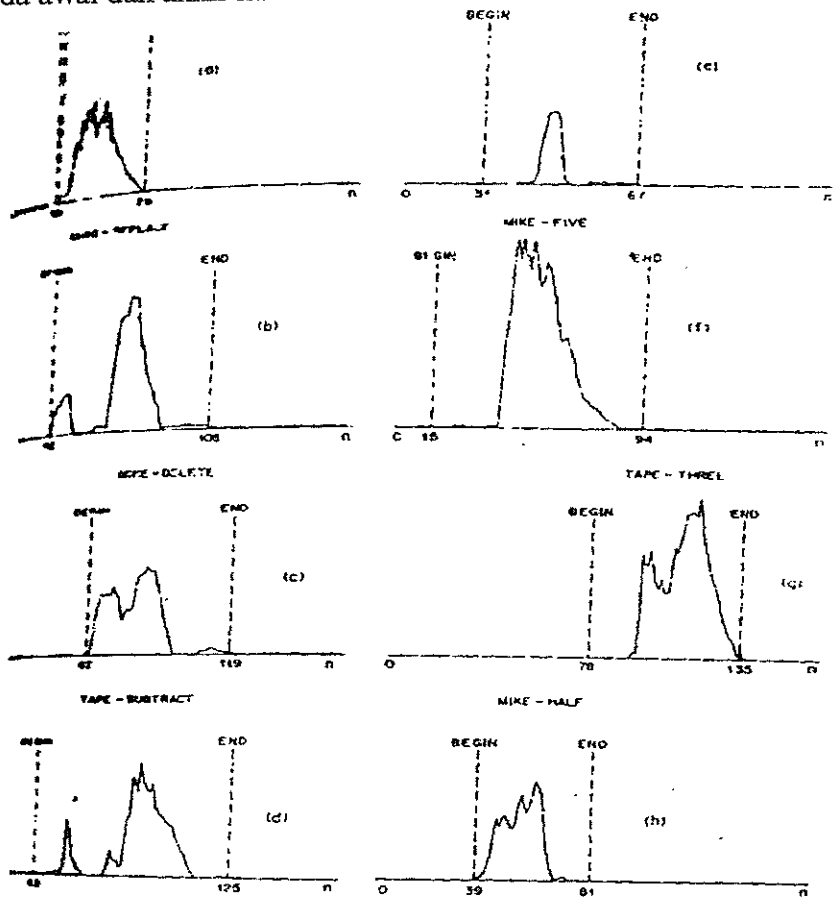
Gambar 4.16 Contoh pengukuran rata-rata magnitude dan zero-crossing untuk sebuah kata dengan bunyi desah yang kuat pada awalnya.

Algoritma ini dapat dilihat pada gambar 4.16. Jumlah zero-crossing dibingkai tiap 10 ms (Pers. 4.18) dan magnitude rata-rata dihitung dengan jendela 10 ms (Pers. 4.12). Kedua fungsi ini dihitung untuk keseluruhan interval perekaman dengan kecepatan 100 kali/detik. Pada 100 ms pertama, interval tidak

mengandung ucapan. Karakteristik noise latar diperoleh dari perhitungan rerata dan standar deviasi dari nilai rata-rata magnitude dan zero crossing. Data statistic yang diperoleh digunakan untuk mendapatkan nilai zero-crossing dan ambang energi. Standar ITU dan ITL memiliki titik awal dan akhir ucapan yang berbeda.Titik N_1 dan N_2 merupakan titik awal dan akhir sinyal.

Gambar 4.17 menunjukkan bagaimana algoritma ini bekerja pada kata terisolasi. Pada gambar ini terdapat 8 buah plot fungsi magnitude rata-rata dari 8 kata yang berbeda (untuk 2 sumber yang berbeda). Beberapa kata direkam di dalam ruangan computer yang bising (diberi tanda-Mike) dan yang lainnya direkam pada pita analog (diberi tanda-Tape). Pada gambar 4.17a (word sembilan), nilai ambang rata-rata magnitude cukup untuk meletakkan titik batas. Pada gambar 4.17b (kata "replace"), algoritma zero-crosisng digunakan untuk menentukan titik akhir ucapan yang mengandung bunyi desah /s/. Walaupun /s/ memiliki rata-rata magnitude yang besar, standar ini tdak dapat digunakan untuk mencari titik akhir yang sebenarnya karena ambang rerata magnitude diset secara konservatif. Pada gambar 4.17c, akhiran /t/ pada kata "delete" adalah tepat karena nilai zero-crosing di atas 70 ms lenyap saat /t/ dilepaskan.

Pada gambar 4.17d terdapat contoh dimana nilai zero crossing signifikan di kedua titik pada awal kata "subtract". Algoritma ini berhasil menentukan titik awal dan akhir kata. Gambar 4.17e-4.17h menunjukkan cohtoh kata dengan bunyi desah pada awal dan akhir kata.



Gambar 4.17 Urutan dari magnitude rata-rata yang menunjukkan algoritma pencarian titik akhir.

4.5 Perkiraan periode Pitch dengan pendekatan pengolahan secara parallel

Perkiraan periode nada (pitch) atau frekuensi fundamental merupakan salah satu masalah terpenting dalam speech processing. Detektor pitch digunakan dalam vocoder, sistem identifikasi dan verifikasi speaker dan alat Bantu bagi penyandang cacat.Karena hal ini penting, banyak solusi bagi masalah ini yang sudah diajukan.

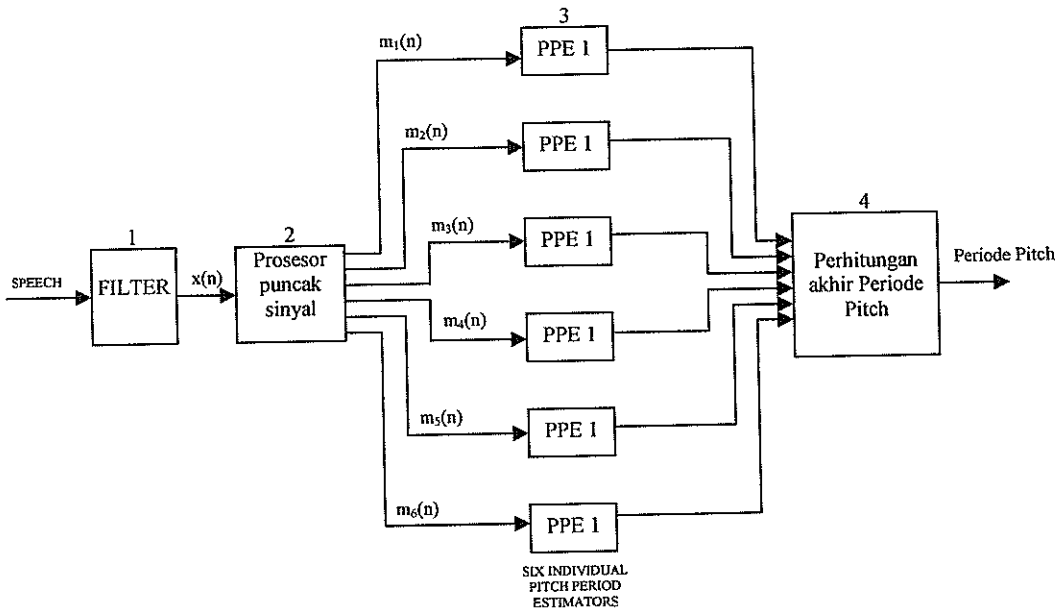
Pada bagian ini kita bahas tentang skema pendeteksian pitch pertama yang diajukan oleh Gold yang kemudian dimodifikasi oleh Gold dan Rabiner. Alasan pembahasan ini adalah: (1) telah banyak digunakan dengan berhasil dalam aplikasi yang luas, (2) berdasarkan pada proses dalam kawasan waktu (3) dapat diimplementasikan untuk pengoperasian secara cepat pada komputer dengan tujuan umum atau dapat dengan mudah dikonstruksikan pada hardware digital dan (4) mengilustrasikan kegunaan prinsip dasar dari parallel processing.

Prinsip dasar skem ini adalah:

1. Sinyal ucapan diproses untuk menghasilkan deretan impuls yang menahan periode sinyal asli dan menghilangkan sifat yang tidak relevan dengan proses pendeteksian nada.
2. Proses ini menggunakan detektor nada yang amat sederhana untuk memperkirakan periode tiap deretan impuls.
3. Perkiraan beberapa detektor nada secara logical terkombinasi untuk menyimpulkan periode bentuk gelombang ucapan.

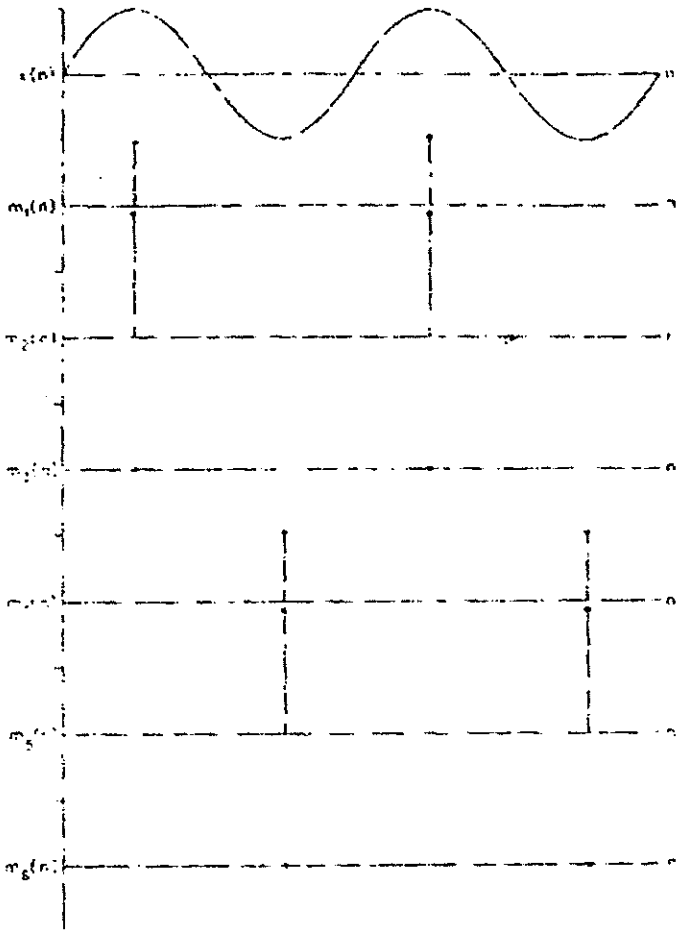
Skema yang diajukan oleh Gold dan Rabiner digambarkan pada gambar 4.18. Sinyal ucapan dicuplik dengan kecepatan yang cukup untuk menghasilkan resolusi yang baik; contoh 10 kHz dengan periode $T = 10^{-4}$ detik. Sinyal di tapis dengan tapis lolos rendah (LPF) dengan frekuensi cutoff sekitar 900 Hz untuk menghasilkan gelombang yang halus. Sebuah tapis lolos pita (BPF) yang meloloskan frekuensi 100 Hz sanpai 900 Hz dibutuhkan untuk menahan noise berfrekuensi 60 Hz. (Penapisan ini dapat dilakukan dengan filter analog sebelum sampling atau dengan filter digital sesudah sampling.)

Setelah penapisan, puncak dan lembah sinyal dapat dilokasikan dan dari lokasi serta amplitudo, beberapa deret impuls (6 pada gambar 4.18) diperoleh dari sinyal tertapis. Setiap deret impuls terdiridari impuls positif yang muncul pada lokasi dari puncak maupun lembah sinyal input. 6 kasus yang dipakai oleh Gold dan Rabiner adalah:



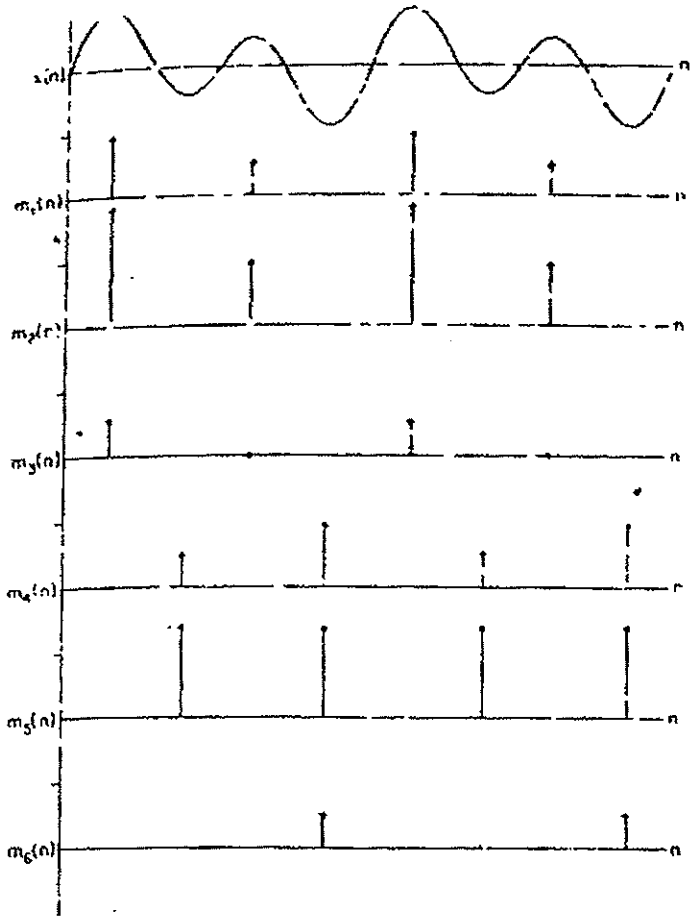
Gambar 4.18 Blok diagram detektor periode pitch parallel dalam kawasan waktu.

1. m_1 (n): sebuah impuls yang equal dengan amplitudo puncak yang muncul pada lokasi puncak.
2. m_2 (n): sebuah impuls yang equal dengan selisih di antara amplitudo puncak dengan ampiltudo lembah terdahulu yang muncul pada tiap puncak.
3. m_3 (n): sebuah impuls yang equal dengan selisih di antara amplitudo punack dengan amplitudo puncak terdahulu yang muncul pada tiap puncak. (Jika selisih ini negative maka impulsnya diset nol).
4. m_4 (n): sebuah impuls yang equal dengan amplitudo negative pada satu lembah yang terjadi pada tiap lembah.
5. m_5 (n): sebuah impuls yang equal dengan amplitudo negative pada satu lembah ditambah amplitudo pada puncak terdahulu yang terjadi pada tiap lembah.
6. m_6 (n): sebuah impuls yang equal dengan amplitudo negative pada satu lembah ditambah amplitudo local minimum yang terdahulu yang terjadi pada tiap lembah. (Jika selisih ini negative maka impulsnya diset nol).

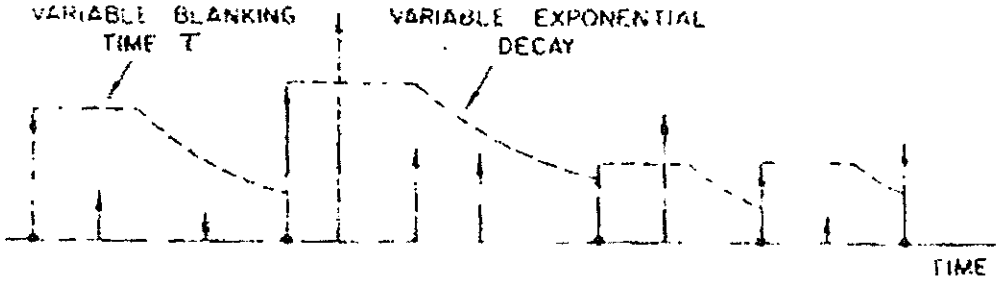


Gambar 4.19 Input (sinus) dan deretan impus yang dibangkitkan dari puncak dan lembah sinyal input.

Gambar 4.19 dan 4.20 menunjukkan dua contoh – sinyal sinus murni dan komponen fundamental lemah ditambah harmonisa kuat kedua – bersamaan dengan deretan impuls yang telah didefinisikan di atas. Secara jelas, deretan impuls memiliki periode fundamental yang sama dengan sinyal input asli, walaupun m_5 (n) pada gambar 4.20 hampir menjadi periodic dengan setengah periode fundamental. Tujuan dari pembangkitan deretan impuls ini untuk memudahkan perkiraan periode pada basis short-time. Operasi sederhana dari pitch period estimators ditunjukkan pada gambar 4.21. Setiap deretan impuls diproses dengan sistem nonlinier – time varying. Saat sebuah impuls dari amplitudo dideteksi pada input, outputnya direset menjadi nilai impuls tersebut dan kemudian ditahan untuk interval kosong, τ (n) — selama tidak ada pulsa yang dapat dideteksi. Pada akhir interval kosong, output mulai menghilang secara eksponensial. Ketika sebuah impuls melebihi level output yang menghilang ini maka proses akan terulang. Jumlah interval yang menghilang dan kosong ini tergantung dari peride pitch yang diperkirakan paling akhir. Hasilnya berupa semacam penghalusan deretan impuls, menghasilkan quasi-periodic sequence dari pulsa pada gambar 4.21. Panjang dari tiap pulsa merupakan periode pitch yang diperkirakan. Periode pitch diperkirakan secara periodic (contoh 100 kali/detik) dengan mengukur panjang dari pulsa yang merentang sepanjang interval sampling.

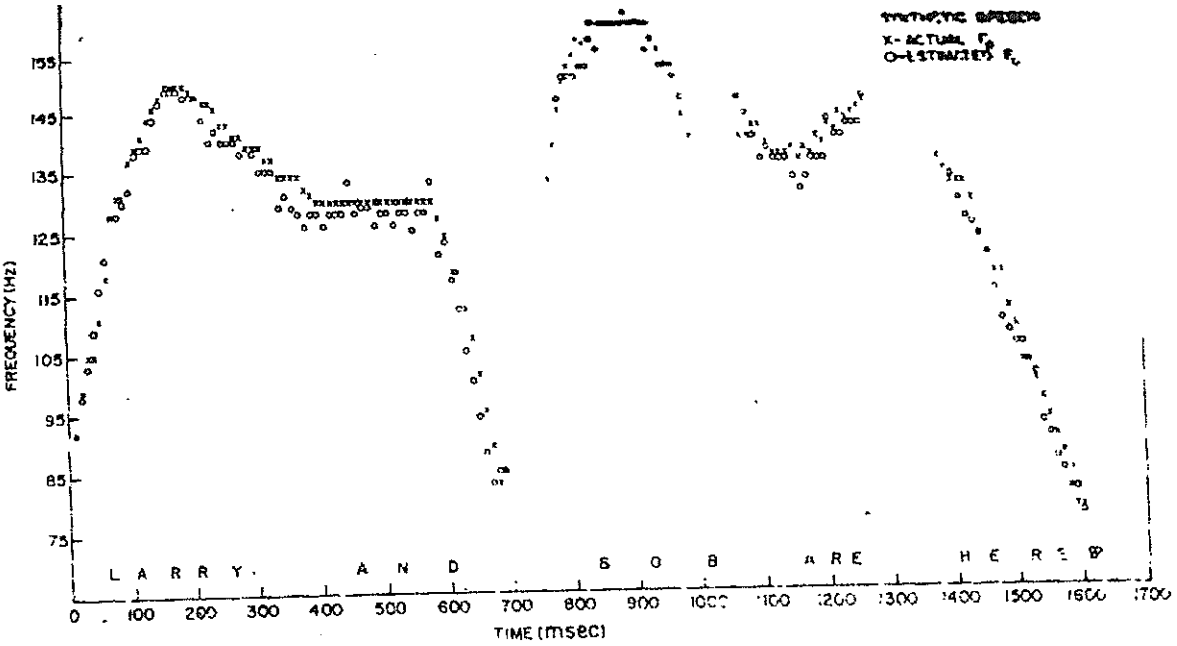


Gambar 4.20 Input (fundamental lemah dan harmonisa kedua) dan deretan impuls yang dibangkitkan dari puncak dan lembah.



Gambar 4.21 Operasi dasar tiap pitch period estimator dari pitch detektor kawasan waktu.

Unjuk kerja dari skema deteksi ini diiliutrasikan pada gambar 4.22 yang menunjukkan keluaran dari contoh sinyal ucapan sintetis. Keunutngan menggunakan sinyal sintetis adalah periode pitch diketahui secara akurat dan keakuratan pengukuran sangat baik. Kerugiannya adalah sinyal sintetis tidak dapat menunjukkan sifat sinyal alamiah.



Gambar 4.22 Perbandingan antara frekuensi nada sinyal alami dengan sintetis.

4.6 Fungsi Autokorelasi dari Short-Time

Fungsi Autokorelasi dari sinyal diskrit kawasan waktu didefinisikan sebagai

$$\phi(k) = \sum_{m=-\infty}^{\infty} x(m)x(m+k) \tag{4.21}$$

Jika sinyal adalah acak dan periodic maka definisi yang tepat adalah

$$\phi(k) = \lim_{N\to\infty} \frac{1}{(2N+1)} \sum_{m=-N}^{N} x(m)x(m+k) \tag{4.22}$$

Pada kasus lain, representasi fungsi autokorelasi sinyal adalah jalan yang tepat untuk menunjukkan sifat-sifat tertentu dari sinyal. Contohnya, jika sebuah sinyal dengan periode P samples, maka dapat dengan mudah ditunjukkan bahwa:

$$\phi(k) = \phi(k+P) \tag{4.23}$$

yaitu, fungsi autokorelasi dari sinyal periodic juga merupaka fungsi periodic dengan periode yang sama. Sifat lain yang penting dari fungsi autokorelasi ini adalah:

1. Merupakan fungsi genap; yaitu $\phi(k) = \phi(-k)$.
2. Mencapai nilai maksimum pada $k = 0$; yaitu $\mid F(k) \mid \leq F(0)$ untuk semua k.
3. Jumlah $F(0)$ equal dengan energi (Pers. 4.2) untuk sinyal terdeterminasi atau daya rata-rata acak atau periode sinyal.

Dari sifat 1 dan 2 kita lihat bahwa untuk sinyal periodic, fungsi autokorelasi mencapai nilai maks. pada sample 0, $\pm P$, $\pm 2P$, ... Periode dapat diperkirakan dengan mencari lokasi maksimum pertama pada fungsi autokorelasi.

Dengan memakai pndekatan yang sama untuk mendefinisikan short-time representation yang baru kita diskusikan, kita definisikan fungsi auto korelasi short-time sebagai:

$$R_d(k) = \sum_{m=-\infty}^{\infty} x(m)w(n-m)x(m+k)w(n-k-m) \tag{4.24}$$

Persamaan ini dapat diartikan sebagai: pertama sebuah segmen dari ucapan dipilih dengan perkalian dengan window; kemudian definisi deterministic dari autokorelasi persamaan 4.21 diaplikasikan pada winnowed segment dari ucapan. Maka:

$$R_n(-k) = R_n(k) \tag{4.25}$$

Dengan memakai persamaan ini, kita dapat mengekspresikan $R_n(k)$ pada pers 4.10. Pertama kita catat bahwa:

$$\begin{aligned} R_n(k) &= R_n(-k) \\ &= \sum_{m=-\infty}^{\infty} x(m)x(m-k)[w(n-m)w(n+k-m)] \end{aligned}$$

(4.26)

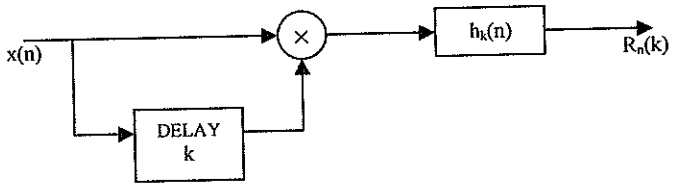
Jika kita definisikan

$$h_k(n) = w(n)w(n+k) \tag{4.27}$$

Maka pers. 4.26 dapat ditulis sebagai:

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(m)x(m-k)h_k(n-m) \tag{4.28}$$

Nilai autokorelasi ke-k "lag" pada waktu n didapat dengan penapisan urutan $x\ (n)\ x\ (n\ -\ k)$ dengan tapis yang memiliki respon impuls $h_k\ (n)$. Hal ini di gambarkan pada gambar 4.23.



Gambar 4.23 Diagram blok representasi dari short-time autocorrelation

Perhitungan fungsi autokorelasi short-time biasanya dilakukan dengan menggunakan persamaan 4.24 setelah menulis ulang bentuk

$$R_n(k) = \sum_{m=-\infty}^{\infty} [x(n+m)w'(m)][x(n+m+k)w'(k+m)] \tag{4.29}$$

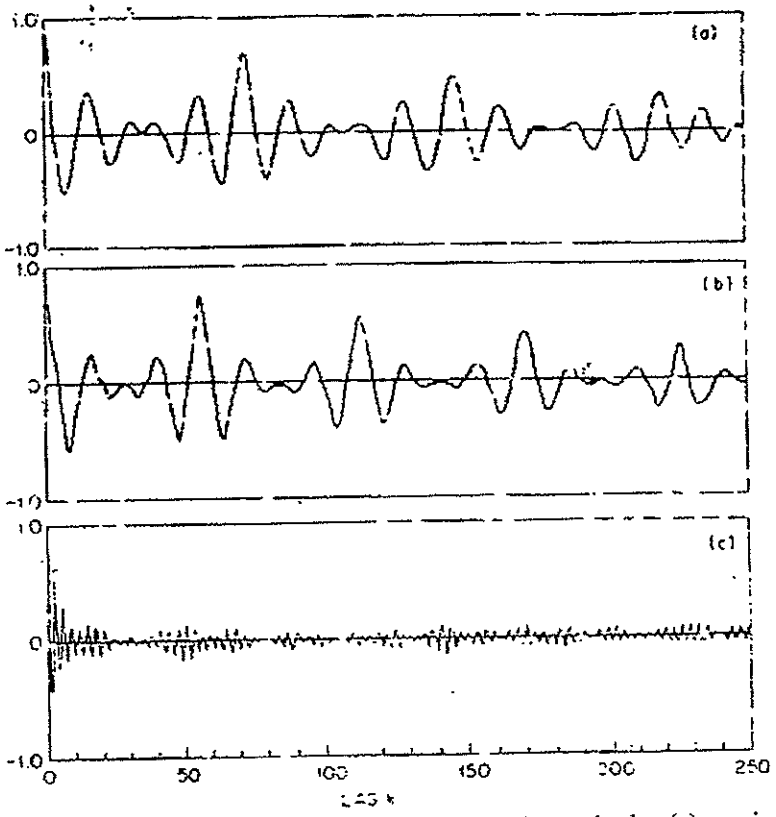
dimana $w'(n) = w(-n)$. Persamaan 4.29 menetapkan bahwa waktu origin dari urutan input diubah ke sample n, kemudian dikalikan dengan window w' untuk memilih segmenpendek pada ucapan. Jika window w' merupakan durasi terbatas seperti pada persamaan 4.8 dan 4.9 maka hasil urutan, $x\ (n + m)\ w'(n)$ menjadi durasi terbatas dan pers 4.29 menjadi

$$R_n(k) = \sum_{m=0}^{N-1-k} [x(n+m)w'(m)][x(n+m+k)w'(k+m)] \tag{4.30}$$

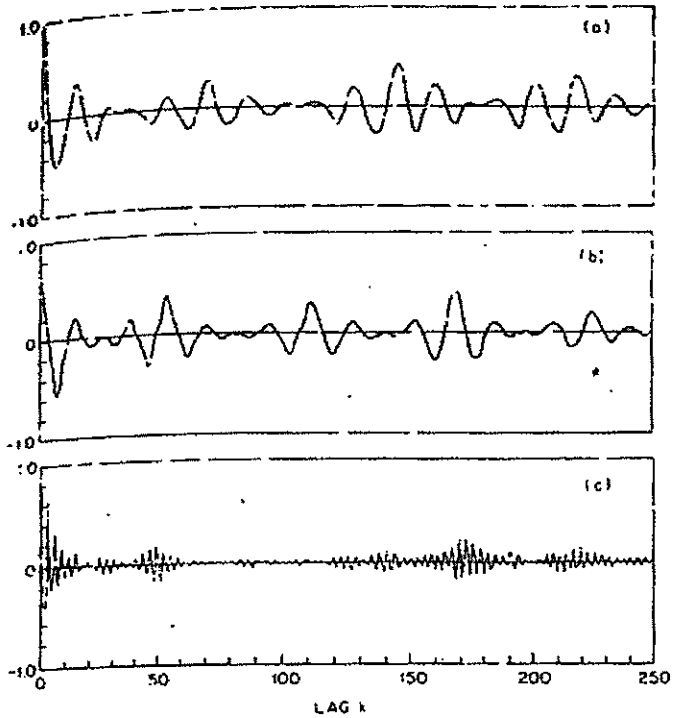
Catat bahwa ketika jendela persegi atau Hamming pada pers 4.8 dan 4.9 digunakan untuk w' pada persamaan 4.30 mereka berhubungan dengan sistem nonkausal dalam pers. 4.28. Untuk panjang window berhingga, posisi ini tidak masalah jika delay yang cocok dapat digunakan dalam proses, walau pada aplikasi rel-time sekalipun.

Perhitungan autokorelasi tertnggal ke-k dengan menggunakan pers.4.30 akan membutuhkan N perkalian untuk menghitung $x\ (n + m)\ w'(m)$, dan $(N - k)$ perkalian dan penambahan untuk perhitungan jumlah perkalian yang tertinggal. Perhitungan dari banyak lag diperlukan dalam memperkirakan keperiodikan yang memerlukan banyak aritmatika. Ini dapat dikurangi dengan menggunakan keuntungan dari sifat unik pers. 4.30.

Alternatif lain dari pers.4.30 yang berguna jika hanya memerlukan sedikit lag pada pers. 4.28. Jika window W_n dipilih seperlunya, maka $R_n(k)$ dapat dihitung secara rekursif.



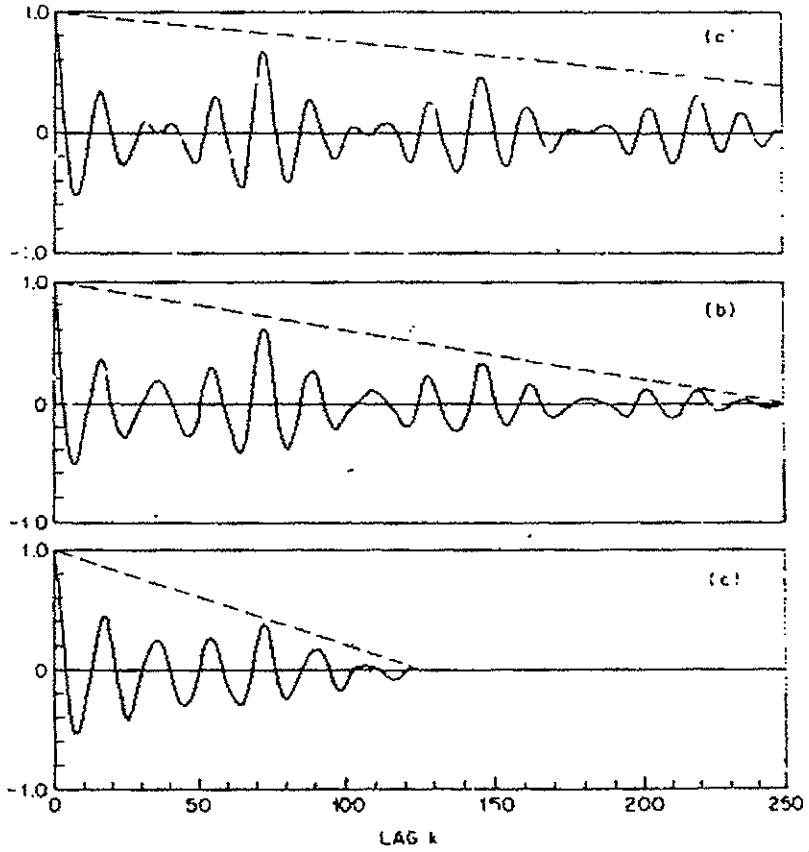
Gambar 4.24 Fungsi autokorelasi untuk (a) dan (b) voiced speech; dan (c)unvoiced speech, dengan rectangular window $N = 401$.



Gambar 4.25 Fungsi autokorelasi untuk (a) dan (b) voiced speech; dan (c)unvoiced speech, dengan Hamming window $N = 401$.

Gambar 4.24 menunjukan 3 contoh perhitungan fungsi autokorelasi untuk ucapan yang disample pada 10 kHz dengan menggunakan persamaan 4.30 dengan $N = 401$. Tampak bahwa autokorelasi dievaluasi untuk lag $0 \leq k \leq 250$. Dua kasu spertama merupakan segmen voiced speech dan yang ketiga untuk segmen unvoiced speech. Untuk segmen pertama, puncak terjadi kira-kira pada perkalian 72 yang mengindikasikan periode 7,2 ms atau frekuensi fundamentalnya kira-kira 140 Hz. Untuk bagian voiced yang kedua, periodenya kira-kira 5,8 ms. Untuk unvoiced tidak terdapat periode autokorelasiyang kuat, fungsi auto korelasi untuk unvoiced terlihat seperti bentuk gelombang noise berfrekuensi tinggi.

Gambar 4.25 menunjukkan contoh yang sama dengan menggunakan Hamming window. Dengan membandingkannya dengan gambar 4.24, rectangular window menghasilkan indikasi yang lebih terlihat dari pada Hamming window.



Gambar 4.26 Fungsi autokorelasi untuk voiced speech dengan rectangular window (a) $N = 401$; (b) $N = 251$; (c) $N = 125$.

Gambar 4.26 mengilustrasikan pengaruh rectangular window dengan panjang yang berbeda. Garis putus-putus merupakan plot dari persamaan

$$R(k) = 1 - K/N, \qquad |k| < N \tag{4.31}$$

yang merupakan fungsi autokorelasi dari rectangular window. Secara jelas, makin besar N maka puncak semakin jauh dari garis. Penggunaan panjang window yang lebih pendek dapat memodifikasi persamaan fungi autokorelasi menjadi:

$$\hat{R}_n(k) = \sum_{m=-\infty}^{\infty} x(m) w_1(n-m) x(m+k) w_2(n-k-m) \tag{4.32}$$

Persamaan ini dapat ditulis sebagai

$$\hat{R}_n(k) = \sum_{m=-\infty}^{\infty} x(n+m)\,\hat{w}_1(m)\,x(n+m+k)\,\hat{w}_2(m+k) \tag{4.33}$$

dimana

$$\hat{w}_1(m) = w_1(-m) \tag{4.34a}$$

dan

$$\hat{w}_2(m) = w_2(-m) \tag{4.34b}$$

Kita pilih wndow $\hat{w}_2$ untuk mengikutsertakan sample di luar interval non zero dari window $\hat{w}_1$. Maka:

$$\hat{w}_1(m) = 1 \qquad 0 \le m \le N-1$$

$$= 0 \qquad \text{untuk yang lain} \tag{4.35a}$$

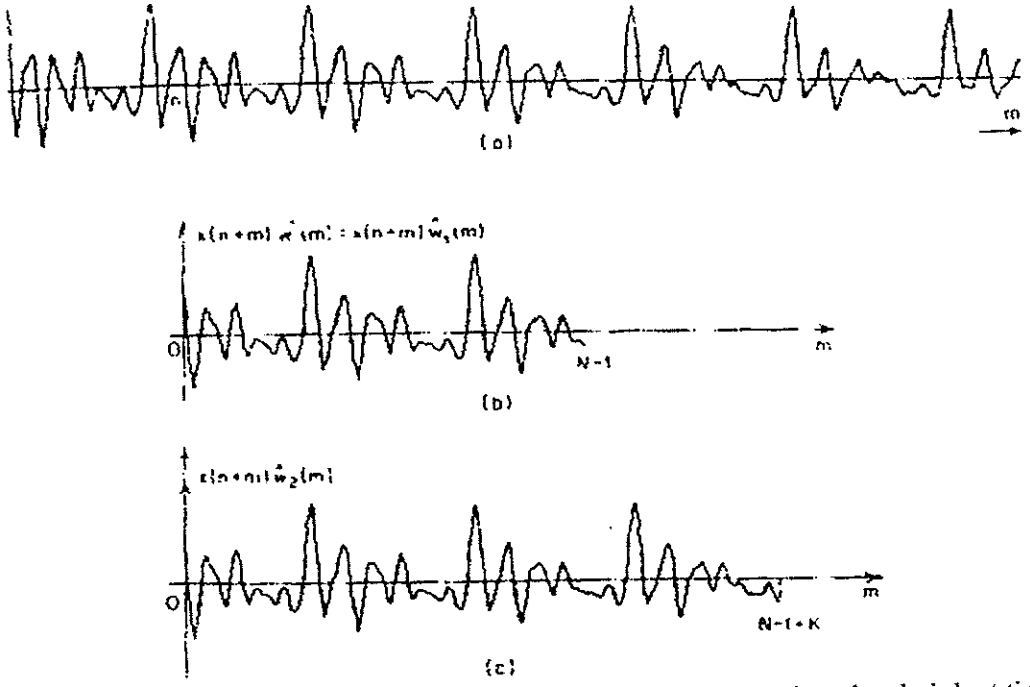
dan

$$\hat{w}_2(m) = 1 \qquad 0 \le m \le N-1+K$$
$$= 0 \qquad \text{untuk yang lain} \tag{4.35b}$$

dimana K merupakan lag terbesar dari interest. Lalu pers.4.33 dapat ditulis sebagai

$$\hat{R}_n(k) = \sum_{m=0}^{N-1} x(n+m)x(n+m+k) \qquad 0 \le k \le K \tag{4.36}$$

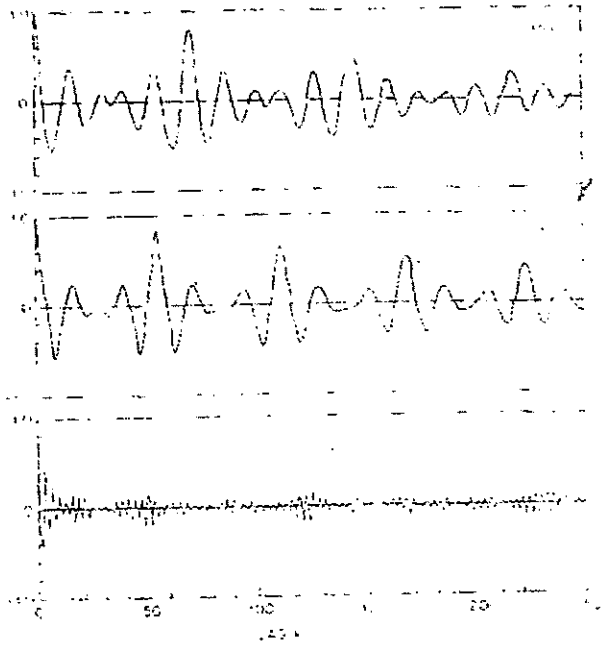
yaitu, rata-ratanya selalu lebih dari N sample, dan sample dari luar interval n ke $n + N$ -1 digunakan dalam perhitungan. Perbedaan dalam data yang digunakan pada perhitungan pers.4.30 dan 4.36 ditunjukkan pada gambar 4.27. Gambar 4.27a menunjukkan bentuk gelombang ucapan dan gambar 4.27b menunjukkan segmen dari N sample yang dipilih oleh rectangular indow. Untuk rectangular window, segmen ini akan dipakai pada ungkapan di pers.4.30 dan pada ungkapan $x(n + m)$ $\hat{w}_1$ (m) pada pers.4.36. Gambar 4.27c menunjukkan ungkapan lain di pers. 4.36. Catat bahwa K sample tambahan diikutsertakan.



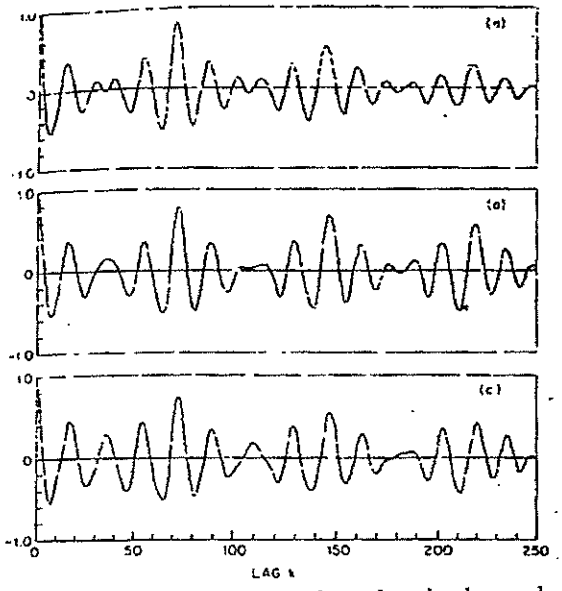
Gambar 4.27 Ilustrasi dari sample yang terlibat dalam perhitungan fungsi autokorelasi short-time.

Persamaan 4.36 akan direferensikan sebagai fungsi autokorelasi short-time yang dimodifikasi. Segmen panjang ucapan berhingga $x(n + m)$ $\hat{w}_1(m)$ dan $x(n + m)$ $\hat{w}_2(m)$ merupakan fungsi korelasi-silang. Maka $\check{R}_n(k)$ memiliki sifat fungsi korelasi silang, bukan fungsi auto korelasi. Sebagai contoh, $\check{R}_n(-k) \neq \check{R}_n(k)$. Namun $\check{R}_n(k)$ akan mendisplaykan puncak pada perkalian periode dari sinyal periodik dan tidak akan menampilkan kejatuhan amplitudo pada nilai k yang besar. Gambar 4.28 menunjukkan fungsi autokorelasi termodifikasi yang berhubungan dengan contoh pada gambar 4.24. Karena untuk N = 401 pengaruh dari variasi bentuk gelombang mendominasi efek "meruncing" pada gambar 4.24. Perbandingan gambar 4.29 dan gambar 4.26 menunjukkan bahwa perbedaannya lebih nyata

untuk nilai N yang lebih kecil. Sangat jelas bahwa puncak pada gambar 4.29 lebih sedikit dari puncak pada $k = 0$ karena deviasi yang dari keperiodikan di atas interval n ke $n + N - 1 + K$ yang terlibat dalam perhitungan pers. 4.36.



Gambar 4.28 Fungsi autokorelasi termodifikasi untuk segmen ucapan pada gambar 4.24 dengan $N = 401$



Gambar 4.29 Fungsi autokorelasi termodifikasi untuk voiced speech pada gambar 4.26

4.7 Fungsi selisih dari magnitude rata-rata short-time

Seperti yang sudah kita bahas, perhitungan fungsi autokorelasi melibatkan banyak aritmatika, walaupun sudah disederhanakan. Teknik yang menghilangkan kebutuhan perkalian adalah berdasarkan ide bahwa untuk input periodik murni dengan periode P, urutannya adalah

$$d(n) = x(n) - x(n-k) \tag{4.37}$$

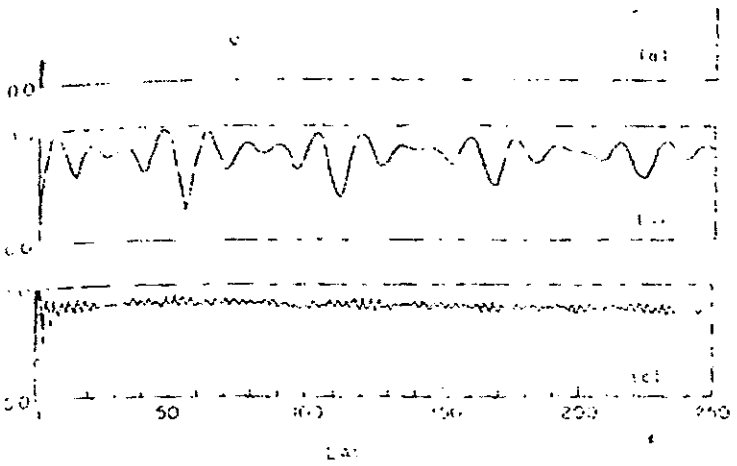
akan menjadi nol untuk $k = 0, \pm P, \pm 2P,...$ Untuk segmen pendek voiced speech adalah wajar untuk mengharapkan bahwa $d(n)$ akan mengecil pada perkalian dengan periode, namun tidak nol. Magnitudo rata short-time dari $d(n)$ yang merupakan fungsi k harus mengecil ketika k mendekati periode. Fungsi selisih magnitude rata short-time (AMDF) didefinisikan sebagai

$$\gamma_n(k) = \sum_{m=-\infty}^{\infty} |x9n+m)w_1(m) - x(n+m-k)w_2(m-k)| \tag{4.38}$$

Jelas bahwa jika $x(n)$ hampir periodik di sepanjang interval yang dijangkau oleh window, $\gamma_n(k)$ harus menurun tajam untuk $k = P, 2P. \ldots$ Catat bahwa lebih rasional jika menggunakan window rektagular. Jika keduanya memiliki panjang yang sama, kita dapatkan fungsi yang mirip dengan fungsi autokorelasi termodifikasi pada pers. 4.36. Dapat ditunjukkan bahwa:

$$\gamma_n(k) \approx \sqrt{2}\beta(k)[\hat{R}(0) - \hat{R}_n(k)]^{1/2} \tag{4.39}$$

$\beta(k)$ pada pers.4.39 bervariasi antara 0.6 dan 1.0 dengan segmen ucapan berbeda, namun tidak berubah secara cepat dengan k untuk segmen ucapan tertentu.



Gambar 4.30 Fungsi AMDF (nrmalisasi sampai 1.0) untuk segmen ucapan yan g sama seperti gambar 4.24 dan 4.28

Gambar 4.30 menunjukkan fungsi AMDF untuk segmen ucapan pada gambar 4.24 dan 4.28 dengan panjang jendela yang sama. Fungsi AMDF diimplementasikan dengan operasi pengurangan, penambahan dan nilai mutlak, secara jelas untuk operasi penambahan dan perkalian dalam fungsi autokorelasi. AMDF memiliki keuntungan sehingga banyak digunakan pada sistem pengolaha suara secara real-time.

4.8 Perkiraan Periode Pitch dengan Fungsi auto korelasi

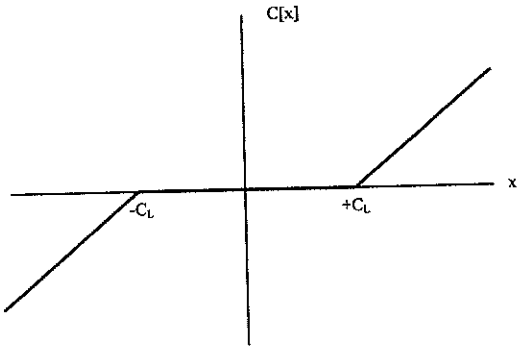
Salah satu dari pembatasan utama dari representasi fungsi autokorelasi menyediakan sebuah representasi yang tepat sebagai dasar untuk menentukan periode pitch sebagai fungsi waktu. Sebagai hasil, kita dapatkan pada gambar

4.26, sebagai contoh bahwa fungsi autokorelasi memiliki banyak puncak. Sebagian besar dari puncak-puncak ini dapat dihubungkan dengan osilasi lembah dari respon vocal tract yang responsive terhadap bentuk tiap periode sinyal ucapan. Pada gambar 4.26a dan 4.26b, puncak pada periode pitch memiliki amplitudo terbesar; bagaimanapun, pada gambar 4.26c, puncak pada saat $k = 15$ sebenarnya lebih tinggi dari puncak pada saat $k = 72$. Hal ini karena windownya lebih pendek dari periode pitch. Maka puncak autokorelasi untuk respon vocal tract lebih besar dari pada untuk periode eksitasi vocal, prosedur sederhana untuk mengambil puncak terbesar pada fungsi autokorelasi akan gagal.

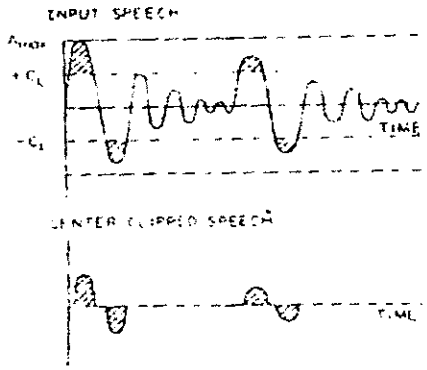
Untuk menghindarinya maka proses sinyal ucapan diperlukan untuk membuat periode lebih nyata bersamaan dengan penghilangan sifat lain yang mengganggu. Tenik yang digunakan sering disebut sebagai "pemerata spectrum" dengan tugas untuk menghilangkan efek dari fungsi transfer pada vocal tract, dengan demikian akan membawa tiap harmonisa pada level amplitudo yang sama sebagaimana dalam kasus sebuah periode deretan impuls. Banya teknik perataan spectrum yang telah diajukan, namun tekinik "center clipping" atau "pemotongan tengah" terlihat menguntungkan.

Pada skema yang diusulkan oleh Sondhi, sinyal ucapan terpotong tengah diperoleh transformasi nonlinier dimana $C[\ ]$ ditunjukkan pada gambar 4.31.

$$y(n) = C[x(n)] \tag{4.40}$$



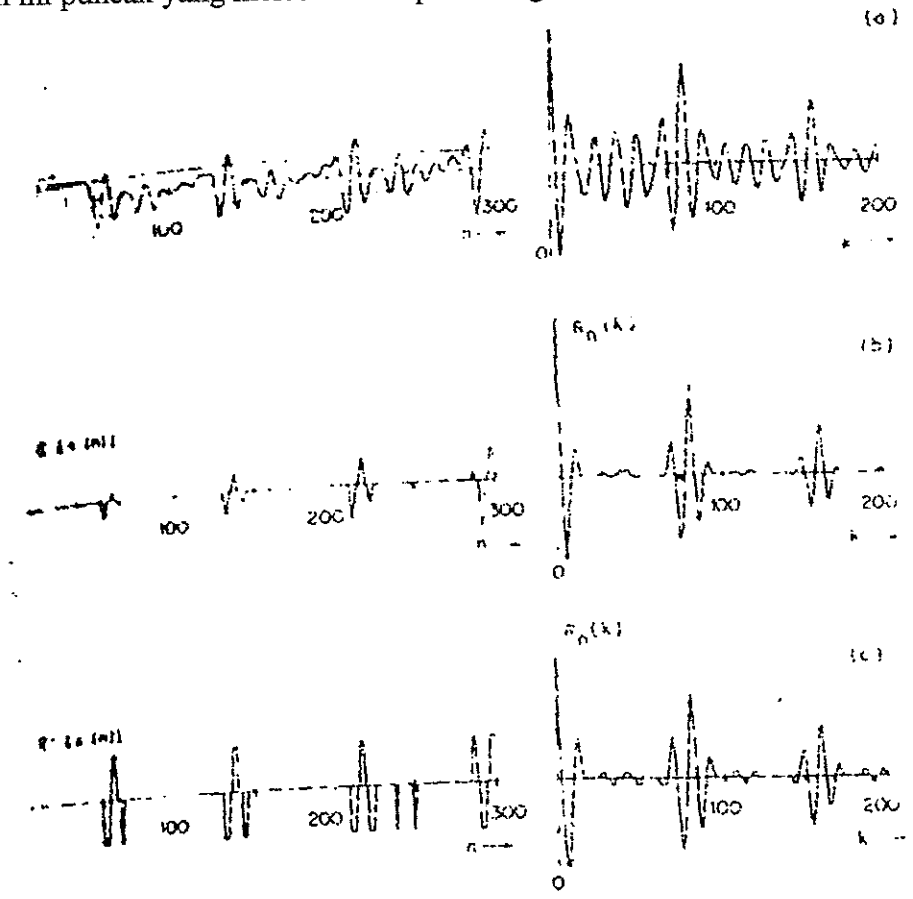
Gambar 4.31 Fungsi pemotongan nilai tengah



Gambar 4.32 Contoh yang memperlihatkan pengaruh pemotongan tengah pada sinyal ucapan

Operasi pemotong nilai tengah ditunjukkan pada gambar 4.32. sebuah segmen ucapan yang akan dipakai dalam perhitungan fungsi autokorelasi ditunjukkan pada Plot bagian atas. Untuk segmen ini amplitudo maksimum A_{maks}

didapatkan dan pada level pemotongan C_L dapat dilihat bahwa untuk sample di atas C_L, keluaran pemotong tengah sama dengan masukan minus level pemotongan. Untuk sample dibawah level pemotongan ,keluarannya adalah nol. Plot bagian bawah pada gambar 4.32 menunjukkan keluaran untuk masukan di atas. Berbeda dengan bagian 4.5 dimana puncak dikonversi menjadi impuls, pada bagian ini puncak yang melebih level pemotongan diubah menjadi pulsa setempat.

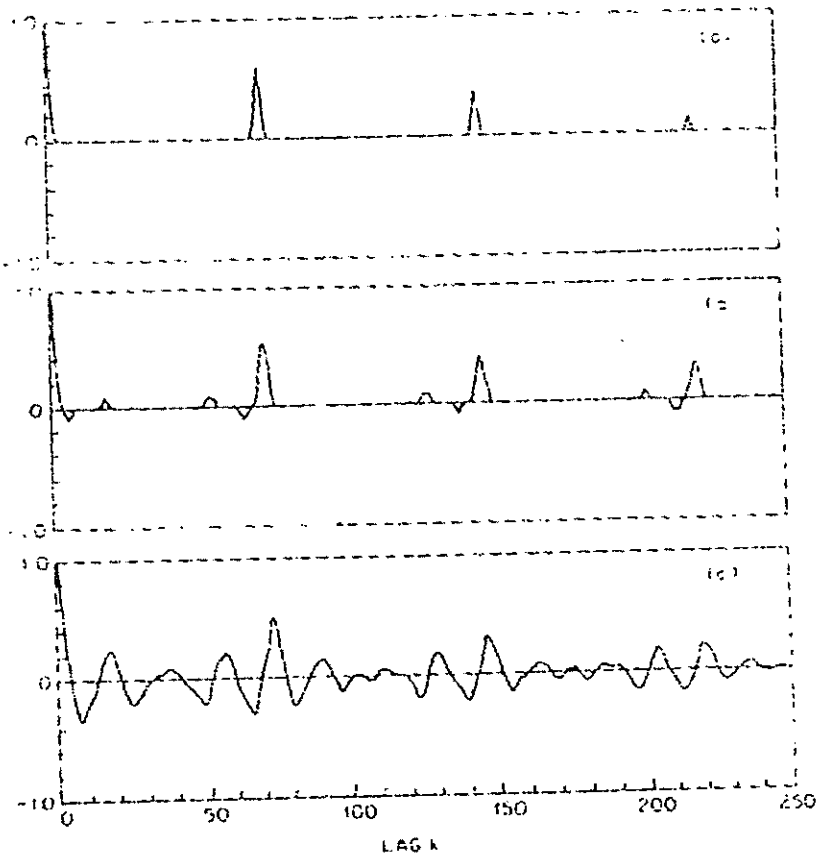


Gambar 4.33 Contoh bentuk gelombang dan fungsi korelasi; 9a) tanpa pemotongan; (b) terpotong tengah; (c) pemotongan tengah 3-level

Gambar 4.33 menggambarkan pengaruh dari operasi pemotongan tengah pada perhitungan fungsi autokorelasi. Gambar 4.33a menunjukka 300 sample segmen (F_s = 10 kHz) voiced speech. Terlihat adanya puncak yang kuat pada bagian kanan. Gambar 4.33b menunjukkan sinyal terpotong tengah dimana level pemotongan telah diatur pada gambar 4.33a. (Pada kasus ini diset pada 68% dari magnitude maksimum pada 100 sample pertama).

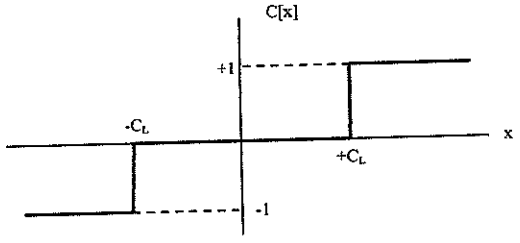
Untuk level pemotongan yang tinggi, sedikit puncak akan melampaui level pemotongan dan kemudian sedikit puncak akan muncul pada keluaran dan maka dari itu sedikit puncak yang tidak berhubungan juga akan muncul pada fungsi autokorelasi. Hal ini ditunjukkan oleh gambar 4.34 yang memperlihatkan fungsi autokorelasi pada segmen ucapan gambar 4.26a, untuk penurunan level pemotongan. Saat level pemotongan menurun, banyak puncak yang akan lolos dari pemotong (clipper) dan kemudian fungsi autokorelasi menjadi leih kompleks. (Catat bahwa level pemotongan untuk nol berkorespondensi dengan gambar 4.26a) Semakin tinggi level pemotongan maka indikasi periode semakin jelas.

Mamun ada kesulitan pada pemakaian level pemotongan yang tinggi yaitu semua amplitudo bias hilang. Untuk alasan ini Sondhi menggunakan level pemotongan 30% dari amplitudo maksimum.



Gambar 4.34 Fungsi autokorelasi untuk ucapan terpotong tengah dengan $N = 401$; (a) C_L 80% dari maksimum; (b) 64%; (c) 48% (Segmen ucapan pada Gambar 4.26a)

Puncak yang tidak berhubungan dengan fungsi autokorelasi dapat dihilangkan dengan pemotongan tengah. Modifikasi sederhana dari fungsi pemotongan tengah menghasilkan penyederhanaan perhitungan fungsi autokorelasi tanpa mempengaruhi pendeteksian pitch. Modifikasi ini ditunjukka pada gambar 4.35. Seperti yang terlihat, keluaran clipper akan +1 jika $x(n) > C_L$ dan -1 jika $x(n) < -C_L$. Selain itu keluaran akan nol. Fungsi ini disebut dengan 3-level center clipper. Gambar 4.33c menunjukkan keluaran 3-level center clipper dari masukan pada gambar 4.33a.



Gambar 4.35 Fungsi pemotongan tengah 3 tingkat

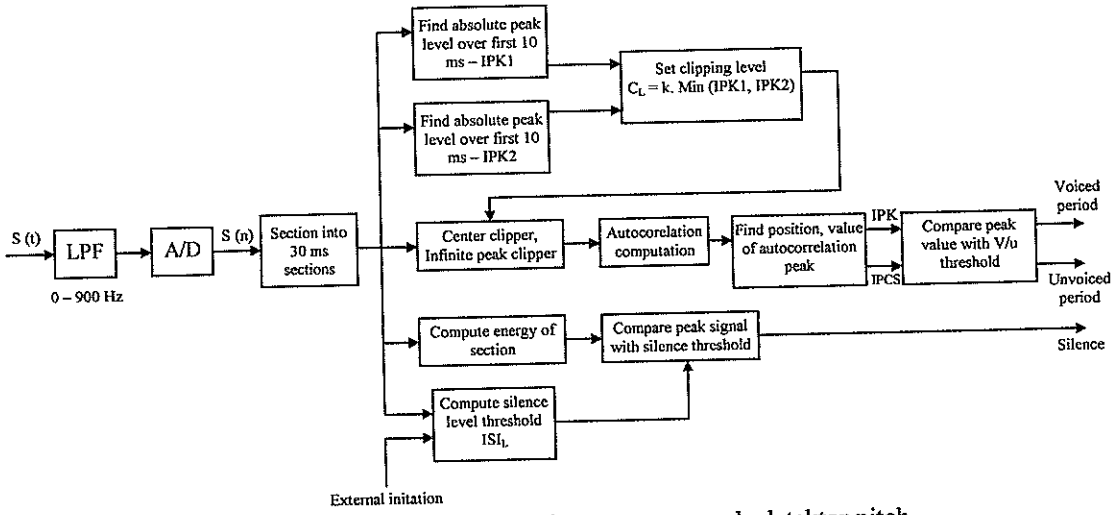
Perhitungan fungsi pemotongan tengah 3 tingkat juga sederhana. Jika kita denotasikan keluaran pemotongan tengah 3 tingkat sebagai $y(n)$ maka keluaran dari $y(n+m)\,y(n+m+k)$ pada fungsi autokorelasi

$$R_n(k) = \sum_{m=0}^{N-k-1} y(n+m)y(n+m+k) \tag{4.41}$$

hanya dapat memiliki 3 nilai yang berbeda

$$\begin{aligned} y(n+m)\,y(n+m+k) &= 0 \quad \text{jika} \quad y(n+m) = 0 \quad \text{atau} \quad y(n+m+k) = 0 \\ &= +1 \quad \text{jika} \quad y(n+m) = y(n+m+k) \\ &= -! \quad \text{jika} \quad y(n+m) \neq y(n+m+k) \end{aligned} \tag{4.42}$$

Maka pada hardware dibutuhkan beberapa logika kombinasional sederhana dan up-down counter untuk mengakumulasikan nilai autokorelasi untuk tiap k.

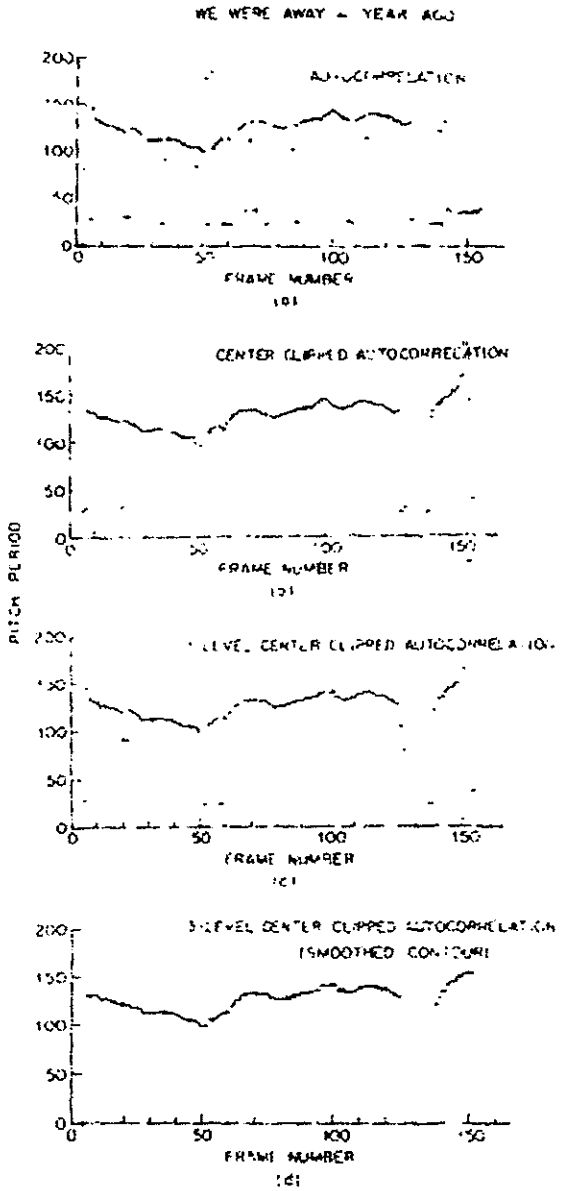


Gambar 4.36 Blok diagram korelasi pemotongan pada detektor pitch

Gambar 4.36 menunjukkan implementasi hardware digital pendeteksi periode pitch yang algoritmanya adalah sbb:

1. Sinyal ucapan ditapis dengan LPF 900 Hz dan disampling dengan 10 kHz.
2. Segmen dengan panjang 30 ms (300 sample) dipilih pada interval 10 ms. Jadi segmen overlap dengan 20 ms.
3. Magnitudo rata-rata pers 4.12 dihitung dengan 100 sample rectangular window. Puncak sinyal di tiap frame dibandingkan dengan mengukur puncak sinyal untuk 50 ms dari noise latar. Jika puncak di atas ambang maka segmen tsb adalah speech. Jika tidak maka sebagai silence.
4. Menentukan level pemotongan dengan persentase tetap (misal 68%) dari minimum pada nilai mutlak maksimum pada 100 sample egmen speech pertama dan terakhir.
5. Dengan level pemotongan ini, sinyal ucapan diproses dengan pemotong tengah 3-level dan fungsi korelasi dihitung sepanjang periode pitch yang diharapkan.
6. Puncak terbesar dari fungsi autokorelasi ditempatkan dan nilai puncak dibandingkan dengan ambang tetap (misal 30% dari $R_n(0)$). Jika puncak jauh dibawah ambang maka segmen merupakan unvoiced dan jika di atas, periode pitch didefinisikan sebagai lokasi dari puncak tertinggi.

Gambar 4.37 menujukkan keluaran (kontur pitch) dari algoritma di atas. Gambar 4.37a adalah kontur pitch yang diperoleh dengan menggunakan autokorerai sinyal ucapan tanpa pemotongan. Catat bahwa hamburan titik yang error mengacu pada fakta bahwa puncak pada lag pendek lebih besar dari pucak pada period pitch. Periode pitch rata-rata pada sample 100 sampai 150 menimbulkan pelemahan yang signifikan. Puncak pada fungsi autokorelasi sehubungan dengan respon vocal tract lebih besar dari sehubungan dengan periditas. Gambar 4.37b dan 4.37c menggunakan pemotongan tengah 3-level dengan fungsi autokorelasi. Sebagian besar error telah dihilangkan dengan pemotongan. Sedikit error terdapat di kedua kontur pitch yang dapat dihilangkan dengan metode penghalusan nonlinier yang akan dibahas pada bagian berikut. Sebuah contoh ditunjukkan pada gambar 4.37d.



Gambar 4.37 Keluaran pitch detektor autokorelasi; (a) tanpa pemotongan (b); dengan pemotongan (gambar 4.31); (c) dengan pemotongan 3-level (gambar 4.35); (d) Output dari (c) yang dihaluskan secara nonlinier.

4.9 Penghalusan Median dan Pengolahan Ucapan

Pada kebanyakan aplikasi pengolahan sinyal, penghalus linier (atau filter linier) umumnya dipakai untuk menghilangkan noise seperti komponen sinyal. Untuk beberapa aplikasi penghalus linier ini tidak cukup untuk menghaluskan. Contohnya dapat dilihat pada gambar 4.37c yang masih mengandung error. Penghalus linier juga dapat mendistorsi kontur pada daerah peralihan antara voiced dan unvoiced speech. Maka diperlukan penghalus nonlinier, namun tidak ada penghalus nonlinear ideal. Suatu penghalus nonlinier dengan menggunakan kombinasi perubahan median dan penghalus linier (diajukan oleh Tukey) dapat menghasilkan keluaran yang kita inginkan.

Konsep dasar dari penghalus linier adalah pemisahan sinyal berdasarkan isi frekuensi yang tidak overlap. Untuk penghalus nonlinier pemisahan sinyal berdasarkan bagaimana sinyal dapat dianggap halus atau kasar (noise). Jadi suatu sinyal $x(n)$ dapat dianggap berbentuk:

$$x(n) = S\,[x(n)] + R\,[x(n)] \tag{4.43}$$

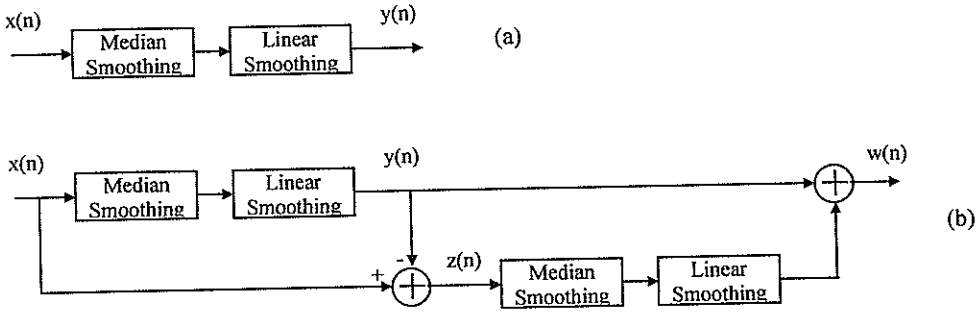
dimana $S\,[x]$ adalah bagian halus dari sinyal x, dan $R\,[x]$ adalah bagian kasarnya. Suatu nonlinieritas yang mampu memisahkan $S\,[x(n)]$ dari $R\,[x(n)]$ adalah nilai median $x(n)$ yang terus berjalan. Keluaran dari penghalus dengan median berjalan ini, $M_L[x(n)]$, secara sederhana merupakan jumlah L, $x(n)$, . . . , $x(n - L + 1)$. Median berjalan dari panjang L memiliki sifat yang kita inginkan sbb:

1. $M_L[\alpha x(n)] = \alpha\, M_L\,[x(n)]$
2. Median tidak akan menimbulkan diskontinuitas pada sinyal jika sinyal tidak memiliki diskontinuitas lain dengan $L/2$ sample.
3. Median akan cenderung mengikuti polynomial orde rendah pada sinyal. Median berjalan harus diempasiskan kembali karena tidak mematuhi sifat superposisi seperti algoritma proses algoritma lain. Yaitu

$$M_L[\alpha x_1(n) + \beta x_2(n)] \neq \alpha M_L[x_1(n)] + \beta M_L[x_2(n)] \tag{4.44}$$

Walau median berjalan umumnya mempertahankan diskontinuitas yang tajam pada sinyal, mereka sering gagal untuk menyediakan penghalusan yang cukup dari noise yang tidak diharapkan seperti komponen sinyal. Suatu kesepakatan yang bagus menggunakan algoritma penghalusan berdasarkan konbinasi dari media berjalan dan penghalusan linier. Karena median berjalan menghasilkan beberapa penghalusan, penghalus linier dapat berupa sistem orde kecil. Umumnya filter linier berupa filter FIR simetris yang delaynya dapat diimbangi. Sebagai contoh, sebuah filter Hamming dengan respon impuls yang umumnya cukup.

$$\begin{aligned} h(n) &= \tfrac{1}{4} \qquad n = 0 \\ &= \tfrac{1}{2} \qquad n = 1 \\ &= \tfrac{1}{4} \qquad n = 2 \end{aligned} \tag{4.45}$$



Gambar 4.38 Blok diagram sistem penghalus nonlinear.

Gambar 4.38a menunjukkan blok diagram penghalus kombinasi berdasarkan median berjalan dan penghalusan linear. Sinyal $y(n)$ pada keluaran penghalus merupakan perkiraan dari sinyal $S\ [x(n)]$. Karena perkiraan ini tidaklah ideal, pelolosan kedua dari penghalusan nonliniear digabungkan dengan algoritma penghalusan pada gambar 4.38b. Karena

$$y(n) = S\ [x(n)] \tag{4.46}$$

maka

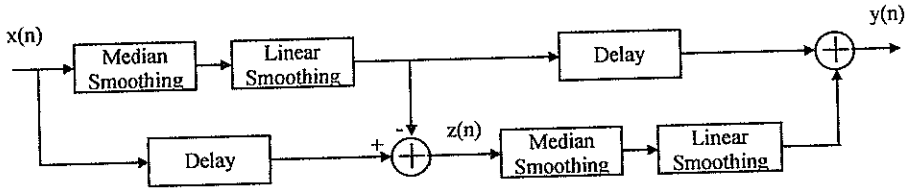
$$z(n) = x(n) - y(n) = R\ [x(n)] \tag{4.47}$$

Pelolosan kedua pada penghalusan nonliniear dari $z(n)$ menghasilkan sinyal pengkoreksi yang ditambahkan pada $y(n)$ untuk memberikan $w(n)$, perkiraan ulang dari $S\ [x(n)]$. Sinyal $w(n)$ membuktikan hubungan

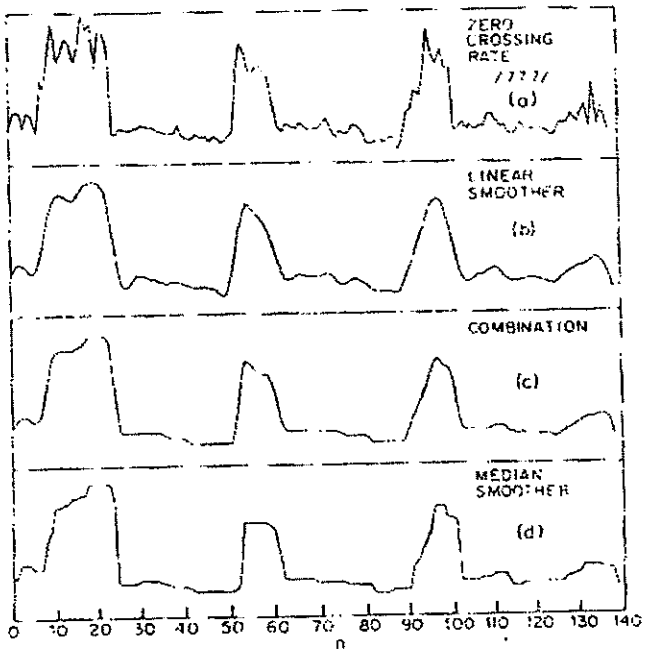
$$w(n) = S\ [x(n)] + S\ [R\ [x(n)]] \tag{4.48}$$

Jika $z(n) = R\ [x(n)]$ maka penghalus nonlinier adalah ideal, dan $S\ [R\ [x(n)]]$ akan sama dengan nol.

Untuk mengimplementasikan penghalus nonlinier pada gambar 4.38 untuk sistem nyata maka diperlukan delay pada tiap penghalus. Tiap penghalus median memiliki delay dari $(L - 1)/2$ sample dan tiap penghalus linier memiliki delay yang berkorespondensi dengan respon impuls yang digunakan. Gambar 4.39 menunjukkan blok diagram realisasi dari penghalus pada gambar 4.38b.

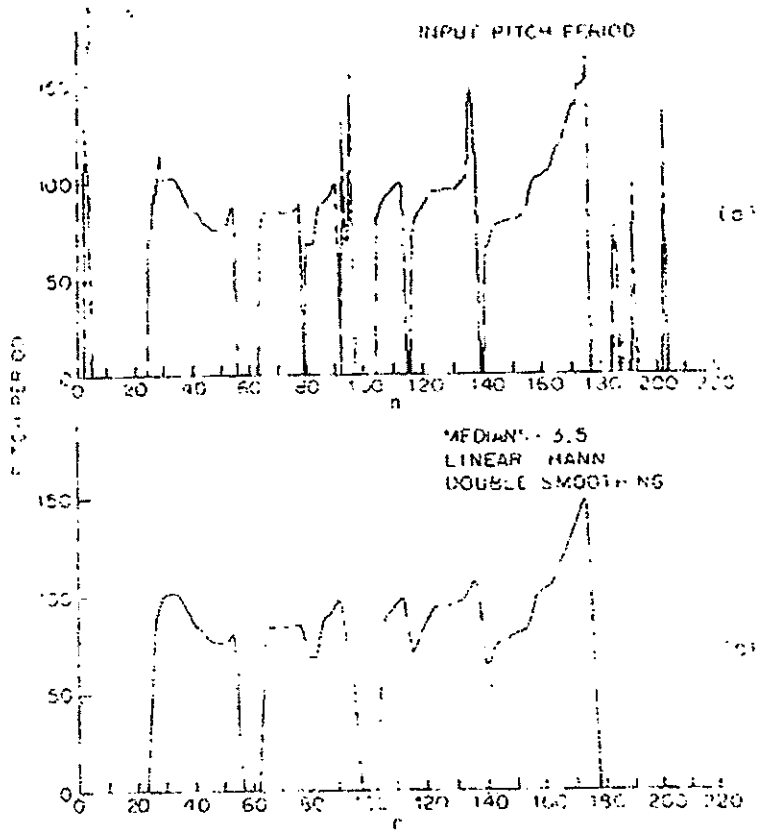


Gambar 4.39 Blok diagram sistem penghalus nonlinear dengan kompensasi delay.



Gambar 4.40 Contoh dari penghalusan nonlinier yang diaplikasikan pada representasi zero-crossing

Gambar 4.40 menunjukkan hasil dengan penghalus yang berbeda-beda pada representasi zero-crossing sinyal ucapan. Sinyal input (gambar 4.40a) adalah kasar. Dapat dilihat pada gambar 4.40d bahwa keluaran dari penghalus median (5 titik median diikuti 3 titik median) memiliki blok seperti efek pada kehadiran komponen frekuensi tinggi pada keluaran yang telah dihaluskan. Keluaran dari penghalus linier (19 titik LPF FIR), yang ditunjukkan pada gambar 4.40b, tercoreng ketika sinyal input berubah secara cepat. Keluaran dari kombinasi penghalus (sebuah median dari 5 diikuti sebuah median dari 3 diikuti oleh 3 titik Hamming window) yang ditunjukkan oleh gambar 4.40c terlihat mengikuti perubahan sinyal input.



Gambar 4.41 Contoh penghalusan nonlinier dari sebuah kontur pitch.

Gambar 4.41 menunjukkan contoh dimana kombinasi penghalus dipakai untuk menghaluskan kontur periode pitch dengan beberapa error yang ada pada perkiraan. Sifat penting dari penghalusan median adalah dapat mengoreksi kesalahan terselubung pada data sementara menggabungkan operai ini dengan penghalusan yang diinginkan. Seperti terlihat pada gambar 4.41, kombinasi penghalus dapat menghilangkan banyak error dan cukup baik dalam menghaluskan sinyal.

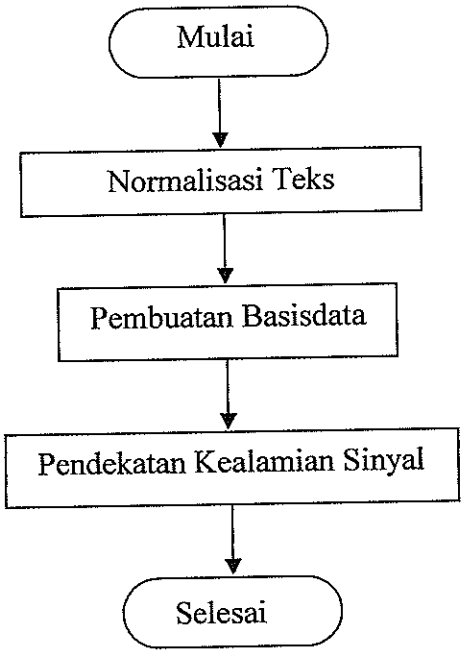
Bab 5
Penerapan Teknik Pengolahan Suara Digital

5.1 Pengubah Tulisan-ke-Ucapan

Pengambilan sampel fonem sinyal ucapan dalam Bahasa Indonesia dari beberapa penutur melalui peralatan perekam suara pada komputer dan menyimpannya dengan format file WAV. Untuk itu diperlukan sebuah perangkat komputer yang telah dilengkapi sound blaster dan sebuah mikropon dengan spesifikasi yang memadai. Dalam pembuatan program simulasi pembangkit ucapan ini perancangan sistem merupakan tahapan yang penting. Perancangan bertujuan agar dalam pembuatannya dapat berjalan secara sistematis dan terstruktur, sehingga hasil program dapat optimal serta berjalan sesuai dengan apa yang dikehendaki. Aspek-aspek dalam perancangan meliputi kemungkinan pengembangan di masa depan, efektifitas dan efisiensi, kemampuan program, dan kemudahan untuk dipahami pengguna (*user friendly*) yang diwujudkan dalam tampilan grafis (*Graphical User Interface*).

5.1.2 Sistem Pengubahan Teks ke Ucapan

Secara umum sistem pengubahan teks ke ucapan ditunjukkan pada Gambar 5.1. Sistem tersebut terdiri dari normalisasi teks, dilanjutkan dengan pembuatan basisdata dan normalisasi sinyal.



Gambar 5.1 Alur sistem pengubahan teks ke ucapan.

Normalisasi Teks

Normalisasi teks Bahasa Indonesia yang dilakukan hanya pada normalisasi huruf dan angka. Basisdata yang digunakan dalam program simulasi pembangkit ucapan adalah suku kata dan huruf yang merupakan bagian dari *diphone* karena ucapan dalam Bahasa Indonesia sebagian besar dibentuk dari suku kata dan huruf. Suku kata yang dipakai adalah gabungan dua huruf konsonan dan vokal. Huruf konsonan yang dibaca mati digunakan untuk konsonan yang berada di akhir kata dan huruf konsonan yang muncul berurutan dalam satu kata. Huruf besar yang muncul lebih dari satu kali dianggap sebagai singkatan dan akan dibaca sebagai huruf yang berdiri sendiri. Teks masukan yang berupa angka akan dibaca sebagai rangkaian suku kata yang merupakan penyusun bunyi angka itu sendiri.

Pembuatan Basisdata

Setelah suku kata dan huruf ditentukan sebagai basisdata, pembuatan basisdata merupakan langkah selanjutnya. Pembuatan basisdata untuk program simulasi pembangkit ucapan meliputi proses perekaman, segmentasi, dan pembentukan `sounds.res`.

1. Proses Perekaman
   Perekaman dilakukan dengan komputer pribadi (PC) menggunakan Goldwave 5.06 dengan memperhatikan hal-hal berikut:
   a. Suku kata yang direkam adalah kombinasi satu konsonan dan satu vokal. Konsonan tersebut adalah (b), (c), (d), (f), (g), (h),(j), (k), (l), (m), (n), (p), (r), (s), (t), (w), (y), (z) dan vokalnya adalah (a), (i), (u), (e), (o).
   b. Bunyi huruf (q) dianggap sama dengan bunyi huruf (k), bunyi huruf (v) dianggap sama dengan bunyi huruf (f), dan huruf (x) jarang sekali digunakan dalam kata-kata Bahasa Indonesia.
   c. Basisdata huruf yang diperlukan adalah (a), (i), (u), (e), (o), (b), (c), (d), (f), (g), (h), (j), (k), (l), (m), (n), (p), (r), (s), (t), (z) dengan menganggap huruf w, y jarang sekali menjadi huruf mati. Basisdata ini didapat dengan melakukan segmentasi sinyal hasil rekaman suku kata yang memuatnya.
   d. Teks dalam huruf besar akan dibaca sebagai singkatan, sehingga basisdata untuk huruf besar akan berbeda dengan huruf kecil.
2. Proses Segmentasi
   Proses ini dilakukan menggunakan Goldwave 5.06 untuk menyempurnakan bunyi masing-masing basisdata. Seluruh basisdata yang diperlukan disimpan dalam format `.wav` dengan nama file sesuai dengan bunyinya, misal `ba.wav` untuk /ba/. Khusus untuk basisdata huruf besar diberi nama file dengan dua huruf, misal `AA.wav` untuk basisdata huruf besar (A), membedakannya dengan file `a.wav`.
3. Melakukan *compiling* file `sounds.rc` menjadi file `sounds.res`
   Langkah membuat file `sounds.res` adalah sebagai berikut:
   a. Membuat file *notepad* yang didalamnya terdapat basisdata yang digunakan dengan format tulisan seperti di bawah ini.

      [nama RES] [format file] [nama file RES]

      Contoh : a wave a.wav

      Contoh di atas menunjukkan bahwa penulisan (a) pada program simulasi akan menyebabkan munculnya bunyi sinyal dengan nama file

`a.wav`. seluruh basisdata yang diperlukan harus tertulis dalam file *notepad* tersebut.

b. Memberi nama file notepad tersebut dengan `sounds.rc` dan menyimpannya dalam direktori yang sama dengan tempat disimpannya seluruh basisdata.
c. Mengambil file `brcc32.exe` dari Delphi\Bin\brcc32.exe.
d. Melakukan *compiling* `sounds.rc` menjadi `sounds.res` dengan membuka *command prompt* dan menuliskan perintah berikut

[brcc32] [sounds.rc]

e. File `sounds.res` akan secara otomatis terbentuk di direktori yang sama dengan `sounds.rc`.
f. Memasukkan file `sound.res` tersebut didirektori program simulasi.

Program simulasi pembangkit ucapan ini masih merupakan program dengan basisdata yang cukup kecil, karena basisdata yang digunakan hanya basisdata tertentu saja belum menyeluruh sesuai dengan karakter Bahasa Indonesia. Normalisasi akronim, tanggal, waktu, karakter khusus, dan simbol diabaikan dalam program simulasi ini.

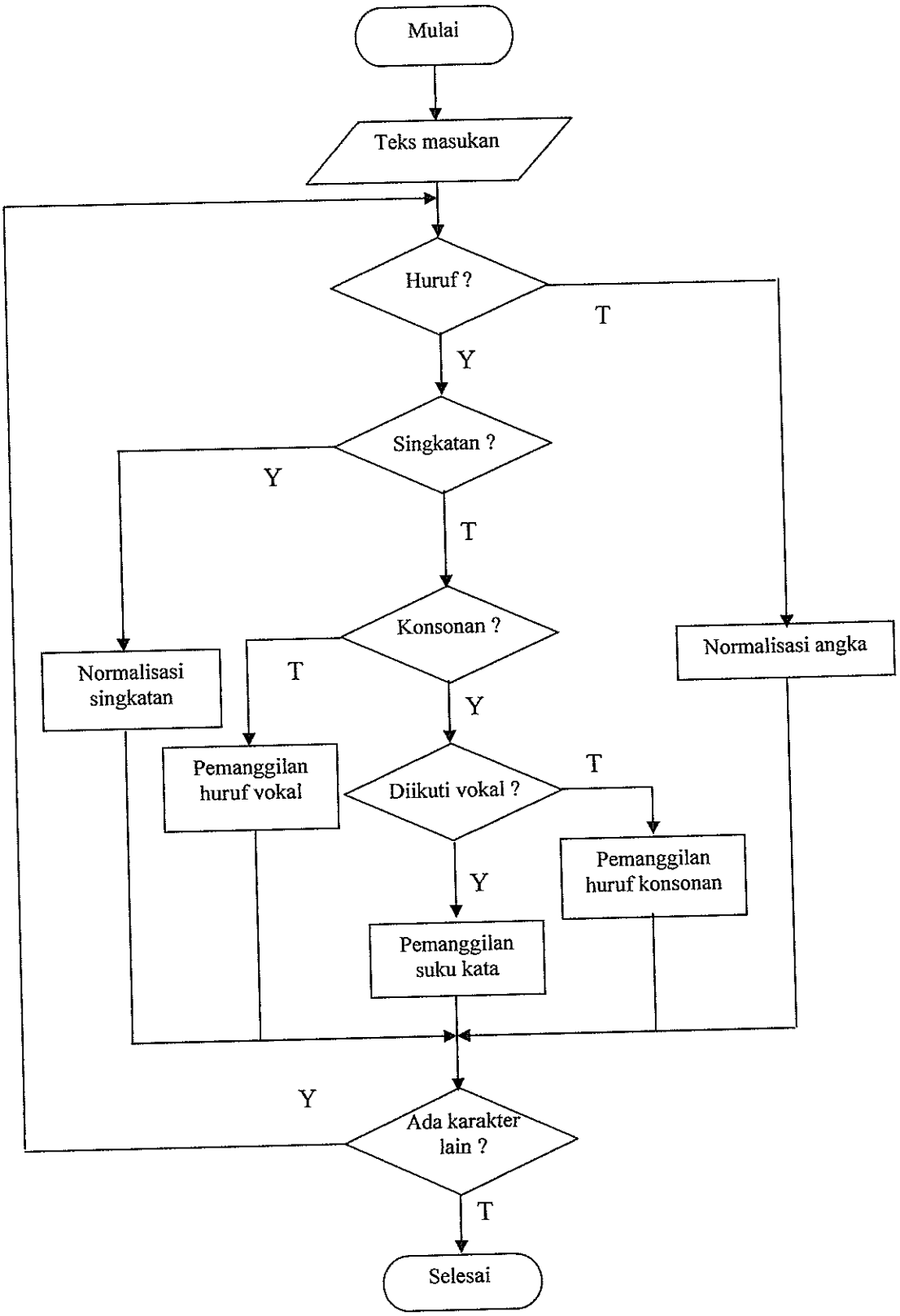
Pendekatan Kealamian Sinyal Ucapan

Program simulasi menggabungkan sinyal basisdata sesuai teks masukan, sehingga sinyal keluarannya masih mengandung *silence*. Sinyal ucapan asli tidak memiliki *silence* pada tiap *diphone* yang ada. Misal, kata [bapak] dibaca dengan /ba/ dan /pak/, bagian sinyal yang berbunyi /pak/ tidak mengandung *silence*. Kata [bapak] akan dibaca oleh program simulasi dengan menggabungkan sinyal /ba/, /pa/, dan /k/. Bentuk sinyal antara /pa/ dan /k/ merupakan *silence*.

5.1.2 Perancangan Bagian TEXT TO VOICE

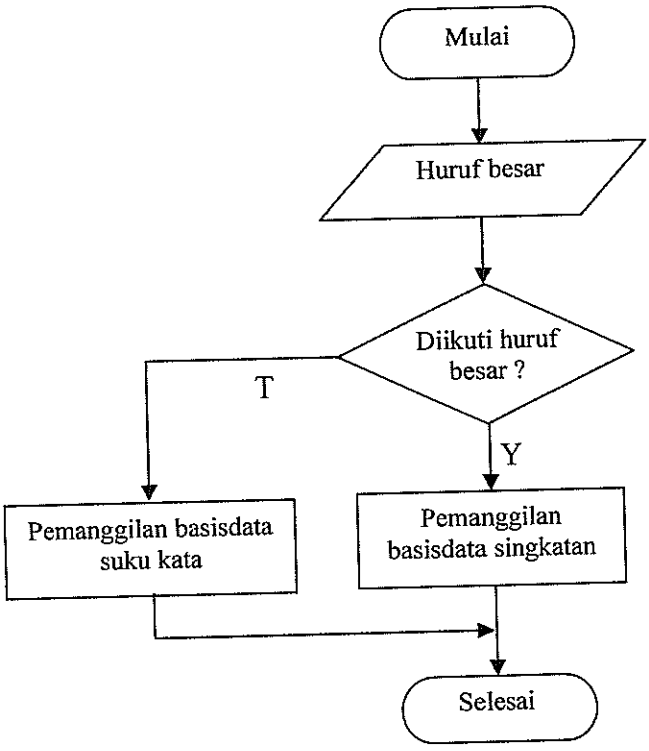
TEXT TO VOICE merupakan bagian yang melakukan proses menyuarakan teks masukan. Bagian ini terdiri dari layar sebagai tempat teks, tombol **baca**, tombol **stop** untuk menghentikan pembacaan, dan tombol **hapus** untuk menghapus teks yang ada di layar. Teks yang dapat dimasukkan adalah teks dengan tipe file `.txt`. Tombol **baca** akan menyuarakan teks yang dimasukkan dengan memanggil basisdata yang ada sesuai dengan bagan alir pada Gambar 5.2.

Teks masukan akan di deteksi apakah merupakan huruf atau bukan, jika merupakan huruf maka akan dideteksi lagi apakah huruf tersebut konsonan atau vokal. Teks masukan berupa huruf besar yang muncul lebih dari dua kali dianggap sebagai singkatan. Apabila teks masukan berupa huruf konsonan dan ternyata diikuti oleh huruf vokal maka basisdata yang dipanggil berupa suku kata, sedangkan untuk konsonan yang diikuti konsonan itu sendiri atau konsonan lainnya, konsonan tersebut akan dianggap sebagai huruf mati. Huruf vokal baik yang diikuti vokal maupun konsonan akan dianggap huruf vokal tersendiri. Jika teks masukan bukan termasuk huruf berarti termasuk dalam angka. Teks masukan berupa angka akan dinormalisasi menjadi teks sesuai bunyinya terlebih dahulu, baru disuarakan.



Gambar 5.2 Alur pembacaan teks masukan program simulasi pembangkit ucapan.

Normalisasi singkatan dilakukan dengan menganggap huruf besar yang muncul lebih dari satu kali merupakan singkatan dan dibaca sesuai basisdata untuk singkatan. Bagan alir normalisasi singkatan ditunjukkan oleh Gambar 5.3.



Gambar 5.3 Alur pembacaan normalisasi singkatan.

Normalisasi angka hanya dilakukan untuk angka satuan dari 0 sampai dengan 9. Teks angka diubah menjadi gabungan suku kata dan huruf yang membentuk bunyi angka tersebut. Pembacaan angka sama untuk semua kondisi. Angka 0 sebagai kata [nol], 1 sebagai kata [satu], 2 sebagai kata [dua], 3 sebagai kata [tiga], 4 sebagai kata [empat], 5 sebagai kata [lima], 6 sebagai kata [enam], 7 sebagai kata[tujuh], 8 sebagai kata [delapan], dan 9 sebagai kata [sembilan].

Proses pemanggilan suku kata dan huruf, baik vokal maupun konsonan, hampir sama. Teks masukan yang berupa konsonan yang diikuti vokal akan memanggil basisdata suku kata yang sesuai yang sudah ada di `sounds.res`.

Pemanggilan konsonan disebabkan oleh adanya teks berupa konsonan yang diikuti konsonan atau konsonan yang ada di akhir kata. Konsonan yang bertemu konsonan mempunyai bunyi yang sama dengan konsonan yang ada di akhir kata.

Teks huruf vokal akan memanggil file basisdata vokal yang bersesuaian. Basisdata vokal juga dipakai untuk singkatan yang berupa huruf vokal karena bunyi vokal sebagai singkatan sama dengan bunyi vokal yang bukan bagian suku kata. Bunyi /a/ pada kata [aku] sama dengan bunyi /A/ pada singkatan /ABC/.

Pembangkitan ucapan untuk singkatan /K/ dilakukan dengan memanggil basisdata dengan nama file `kk.wav` menurut senarai program berikut:

```
'K':Playsound('kk',hInstance,snd_resource or snd_NoStop);
```

Teks masukan yang bukan merupakan huruf akan dianggap sebagai angka yang memerlukan normalisasi. Normalisasi angka dilakukan dengan mengubah angka menjadi teks sesuai dengan bunyi angka tersebut. Pembacaan angka /5/mengikuti senarai program berikut:

```
'5':
    begin
    Playsound('li',hInstance,snd_resource or snd_NoStop);
    Playsound('ma',hInstance,snd_resource or snd_NoStop);
end;
```

Apabila teks berupa huruf konsonan besar bertemu dengan huruf kecil bukan merupakan singkatan. Proses pemanggilan huruf vokal dan huruf konsonan mempunyai bentuk senarai program yang hampir sama. Senarai program untuk huruf (a) adalah sebagai berikut:

```
'a','A':Playsound('a',hInstance,snd_resource or
snd_NoStop);
```

Senarai program untuk pembacaan suku kata /ca/ menggunakan senarai program berikut:

```
'c':
   begin
   c1:=Txt[i+1];
   if (c1='a') Then begin
          PlaySound('ca',hInstance,snd_resource or snd_NoStop);
    i:=i+1;
   end
   else if (c1='i')  Then begin
          PlaySound('ci',hInstance,snd_resource or snd_NoStop);
    i:=i+1;
   end
   else if (c1='u') Then begin
          PlaySound('cu',hInstance,snd_resource or snd_NoStop);
    i:=i+1;
   end
   else if (c1='e') Then begin
          PlaySound('ce',hInstance,snd_resource or snd_NoStop);
    i:=i+1;
   end
   else if (c1='o') Then begin
          PlaySound('co',hInstance,snd_resource or snd_NoStop);
    i:=i+1;
   end
   else
          Playsound('c',hInstance,snd_resource or snd_NoStop);
end;
```

Senarai program di atas juga melukan pemanggilan konsonan /c/ apabila teks masukan tidak termasuk dalam suku kata yang diawali huruf (c). Selang waktu antar kata dalam teks adalah 200 milidetik menurut senarai program berikut:

```
' ': sleep(200);
```

Pembacaan teks dilakukan dengan memanggil basisdata yang sudah ada dalam **sounds.res**. Pemanggilan basisdata dapat dilakukan dengan penulisan senarai program pada **iif.dpr**.

```
program iif;
uses
  Forms,
  iip in '..\program\iip.pas' {Form1};
{$R *.RES}
```

```
{$R Sounds.Res}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

5.1.3 Perancangan Bagian SILENCE REMOVAL

Sinyal keluaran bagian TEXT TO VOICE masih mengandung *silence* dan penggabungan masing-masing basisdatanya masih terlihat jelas, sehingga berbeda dengan sinyal ucapan asli. Untuk memperbaiki hal tersebut, dilakukan penghilangan *silence*. Pengolahan sinyal ucapan lebih lanjut harus dilakukan perekaman terlebih dahulu. Hasil perekaman akan dibunyikan ulang, sehingga menghasilkan sinyal ucapan yang tidak mengandung *silence*.

Untuk melakukan proses diatas, diperlukan komponen dalam bahasa pemrograman Delphi 7 untuk merekam dan memainkan kembali sinyal suara. Komponen yang digunakan adalah *MMTools* yang diunduh dari secara gratis. Program simulasi pembangkit ucapan ini mempunyai beberapa kelemahan, yaitu harus dijalankannya program Delphi 7 pada saat file `.exe` dijalankan.

Pembacaan teks pada bagian TEXT TO VOICE bersamaan dengan proses perekaman. Pada bagian SILENCE REMOVAL disediakan tombol untuk membaca sekaligus merekam yaitu tombol **rekam**, berbeda dengan tombol pembacaan pada bagian sebelumnya. Tombol pembacaan teks atau tombol **baca** pada bagian TEXT TO VOICE hanya membaca teks dan menghasilkan sinyal ucapan yang masih mengandung *silence*. Sinyal yang telah dibaca dan direkam akan menghasilkan sinyal ucapan yang tidak mengandung *silence*.

Komponen *MMTools* digunakan untuk merekam dan membunyikan hasil rekaman tersebut. Pada saat perekaman, tombol **rekam** dan tombol **play** tidak aktif, sedangkan tombol untuk **stop** aktif.

```
procedure TForm1.btnRecordClick(Sender: TObject);
begin
   WaveIn.Start;
   form1.timer1.Enabled := true;
end;
```

Senarai program untuk procedure WaveIn.Start sendiri adalah sebagai berikut:

```
procedure TForm1.WaveInStart(Sender: TObject);
begin
MemStream.SetSize(wioTimeToBytes(WaveIn.PWaveFormat,WaveIn.MaxReco
rdTime));
   MemStream.Position := 0;
   BytesRecorded := 0;
   btnRecord.Enabled := False;
   btnPlay.Enabled := False;
   btnStop.Enabled := True;
end;
```

Proses menyuarakan hasil rekaman menurut senarai program berikut:

```
procedure TForm1.btnPlayClick(Sender: TObject);
begin
   WaveOut.PWaveFormat := WaveIn.PWaveFormat;
   BytesPlayed := 0;
WaveOut.Start;
end;
```

5.2 Perancangan Program Pembaca Email

Perancangan program bantu ini dimaksudkan sebagai alat bantu untuk mempermudah dalam menggunakan aplikasi pensintesa suara berbahasa Indonesia sebagai pembaca email. Perancangan program aplikasi menggunakan pendekatan terstruktur dengan diagram alir yang menjeslakan uruta-urutan proses yang terjadi pada aplikasi. Disamping itu juga terdapat penjelasan mengenai algoritma program. Hasil yang diharapkan adalah berupa sebuah aplikasi yang dapat memberitahu pengguna jika ada email yang masuk dan mengubah isi pesan email menjadi ucapan berbahasa Indonesia.

Algoritma aplikasi pensintesa ucapan berbahasa Indonesia sebagai pembaca email adalah sebagai berikut:

1. Proses inisialisasi basis data, konfigurasi *account* email dan koneksi internet.
2. Program aplikasi menghubungkan diri dengan penyedia email dalam periode tertentu.
3. Setelah terhubung, aplikasi memeriksa isi dari kotak masuk penyedia *email.*
4. Jika pada kotak masuk ada email masuk atau email yang belum terbaca, dan proses pemeriksaan aplikasi berjalan, email masuk tersebut akan diambil dan diproses, kemudian aplikasi memutuskan hubungan dengan penyedia email.
5. Jika tidak ada email yang masuk dan belum terbaca, aplikasi akan memutuskan diri dari penyedia email.
6. Proses email yang masuk dilakukan dengan melakukan pemisahan bagian-bagian email seperti nama dan alamat email pengirim, tanggal masuk, subyek, dan isi email.
7. Data yang didapat akan disimpan dalam tabel basis data.
8. Aplikasi akan memberikan pemberitahuan kepada pengguna bahwa ada email yang masuk.
9. Apabila pengguna menghendaki email tersebut dibacakan, maka isi pesan email yang masuk tersebut akan diubah menjadi ucapan, jika tidak isi pesan akan ditampilkan dalam bentuk teks.

5.2.1 Perancangan *form* aplikasi

Perancangan *form* aplikasi ini terdiri dari tiga *form* utama, yaitu: *form* menu utama. *form* baca email, dan *form* tentang.

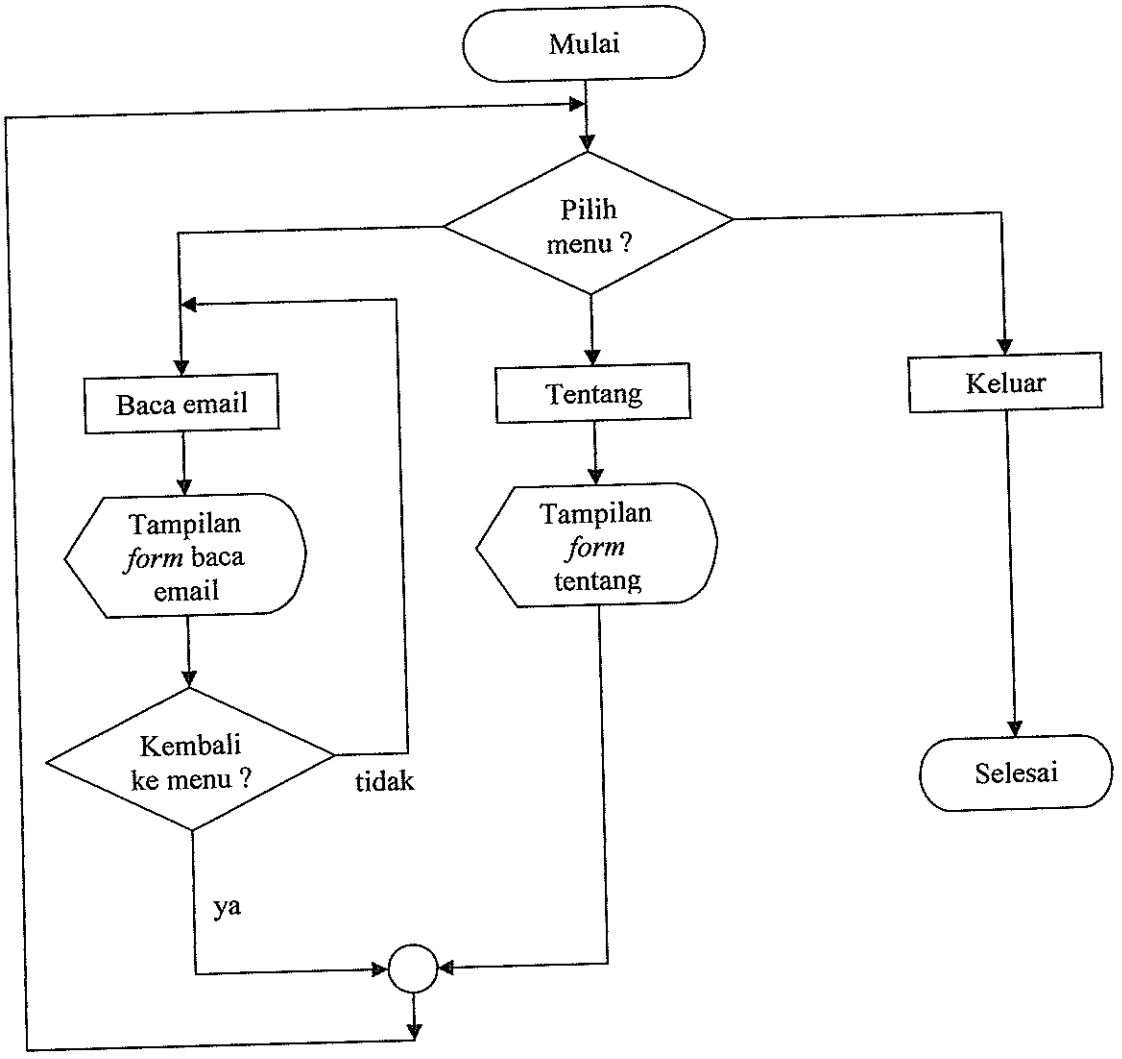
a. *Form* menu utama
   Form menu utama terdiri Judul Penelitian dan nama penyusun Penelitian, serta pilihan menu utama, yaitu : pemilihan menu baca email, pemilihan menu tentang, dan pemilihan untuk keluar dari aplikasi.
b. *Form* baca email
   Pada *form* ini akan ditampilkan isi pesan beserta nama pengirim pesan, alamat email pengirim, dan tanggal pengiriman. *Form* menu baca email ini terdiri dari lima sub pilihan, yaitu: sub pilihan mengirim email, sub pilihan kotak keluar, sub pilihan email terhapus, sub pilihan kotak masuk, dan sub pilihan konfigurasi *account* email.

c. *Form* tentang
Pada *form* ini akan ditampilkan biodata penulis Penelitian, judul Penelitian, dan nama pembimbing Penelitian

5.2.2 Perancangan diagram alir.

Perancangan diagram alir ini dimaksudkan untuk menggambarkan urutan dan tahap pembuatan aplikasi dan memperjelas penggunaannya. Perancangan diagram alir program bantu ini terdiri dari perancangan diagram alir menu utama, dan perancangan diagram alir aplikasi pembaca email.

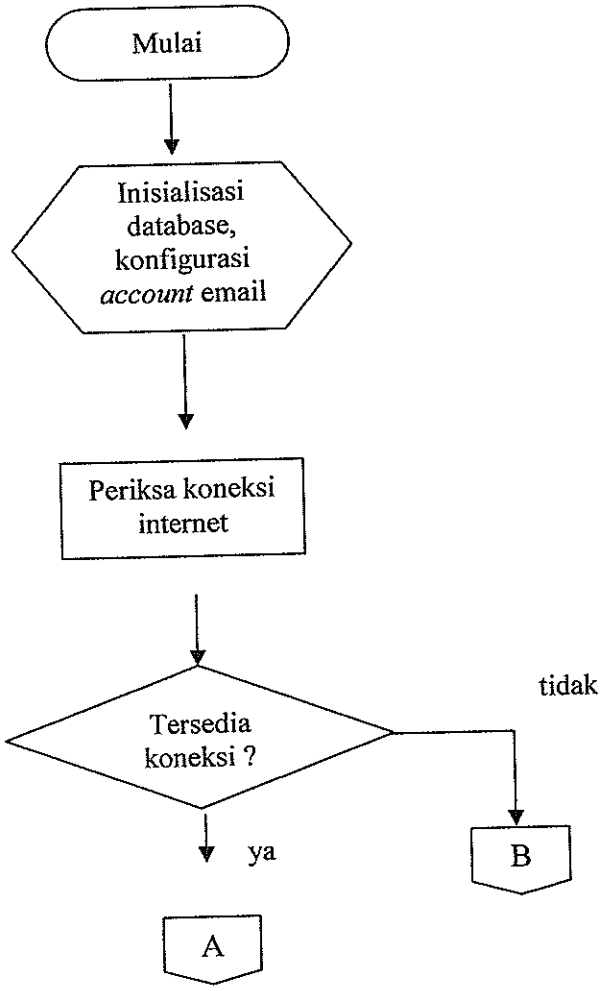
a. Perancangan diagram alir menu utama.
Diagram alir menu utama akan menggambarkan urutan penggunaan program bantu dengan menu-menu yang tersedia yang akan menghubungkan ke pilihan-pilihan yang akan ditampilkan. Diagram alir menu utama program bantu ditunjukkan pada gambar berikut ini.



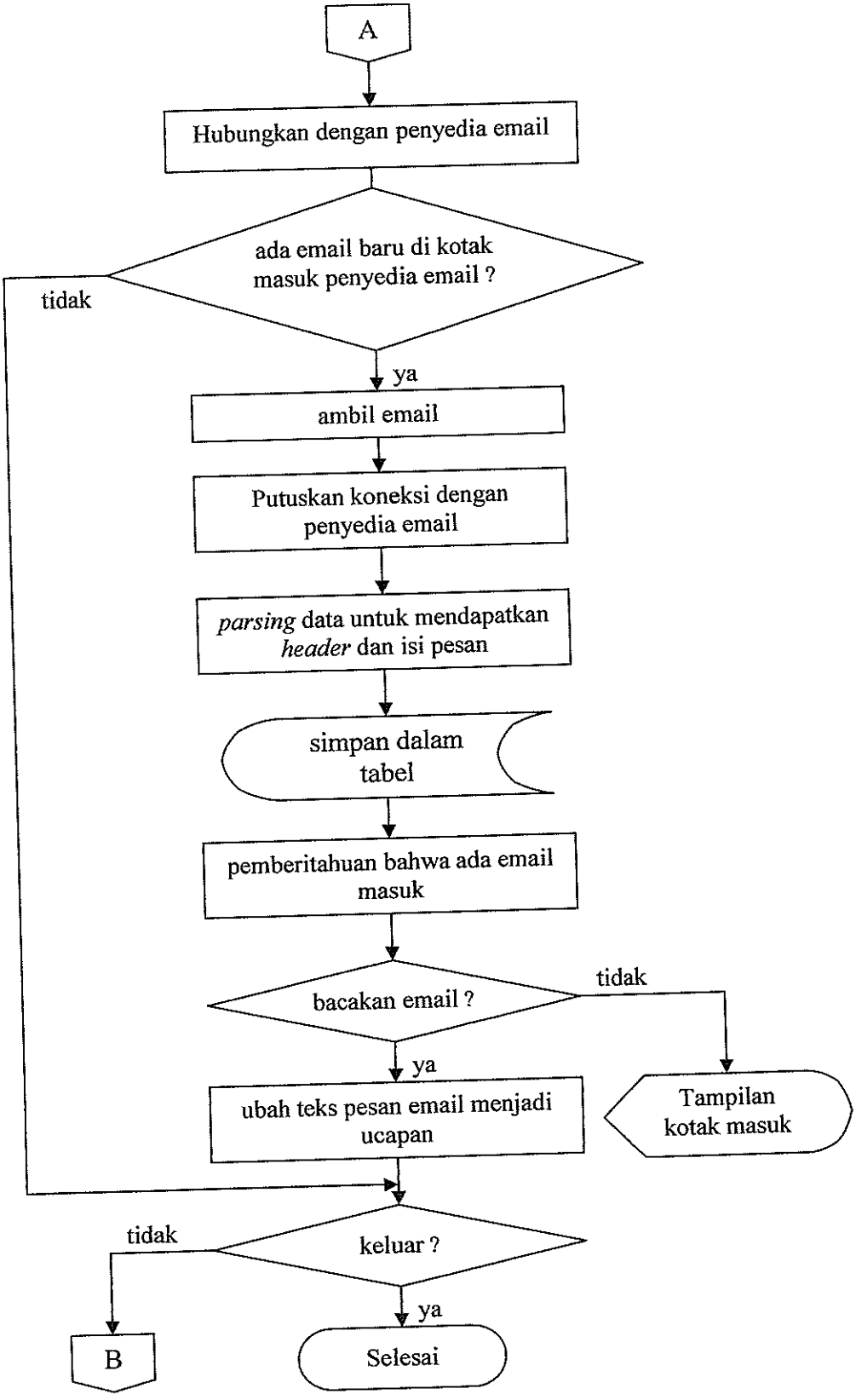
Gambar 5.4 Diagram alir menu utama.

Berdasarkan pada Gambar 5.4 dapat dijelaskan bahwa, pada saat aplikasi dijalankan akan ditampilkan menu utama pada *form* menu utama yang terdiri dari tiga pilihan menu yang bisa dipilih untuk ditampilkan. Piliha menu tersebut terdiri dari pilihan menu baca email yang merupakan aplikasi pensintesa ucapan berbahasa Indonesia sebagai pembaca email, pilihan tentang yang menampilkan tentang penulis Penelitian dan pilihan keluar untuk keluar dari aplikasi.

b. Perancangan diagram alir menu baca email.
Diagram alir menu baca email akan menggambarkan penggunaan *form* baca email yang terdiri dari lima sub pilihan. Diagram alir menu baca email ditunjukkan pada Gambar 5.5 berikut ini.

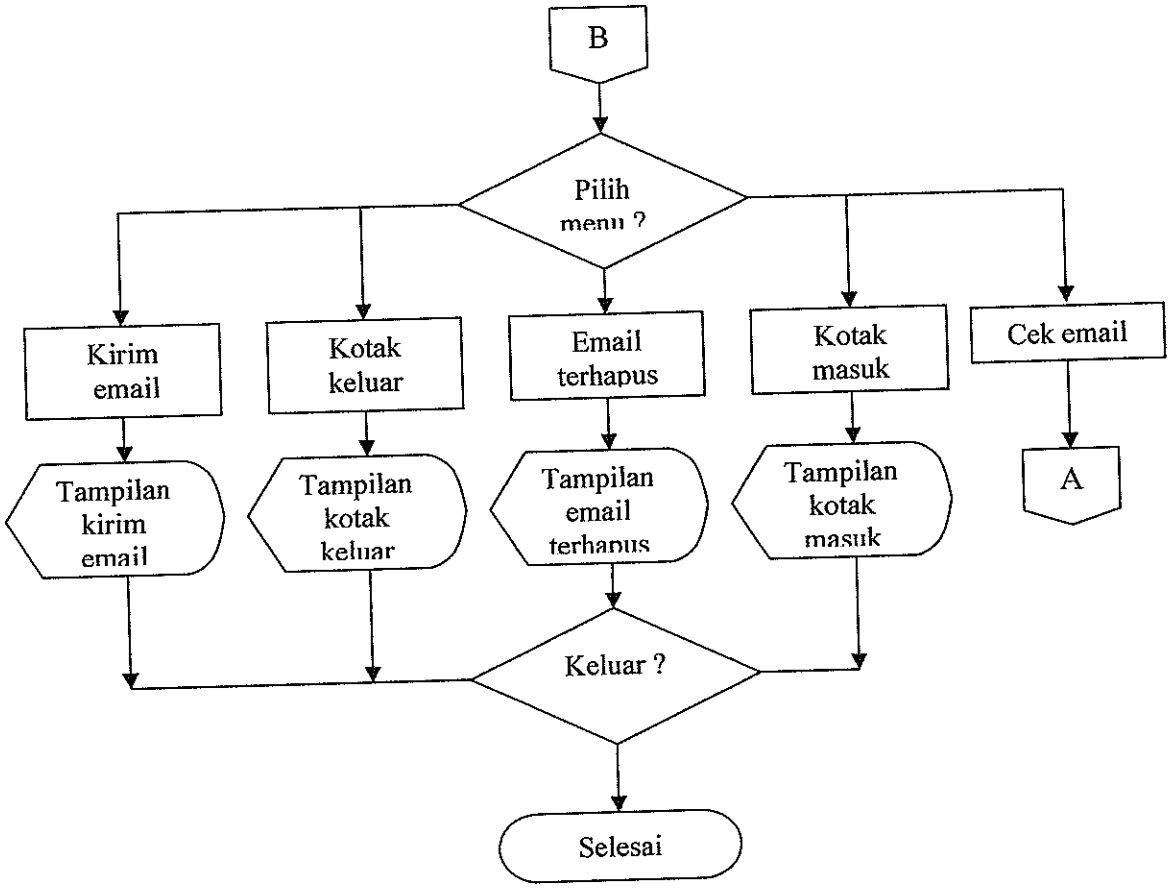


Gambar 5.5 Diagram alir menu baca email



Gambar 5.6 Diagram alir aplikasi pensintesa ucapan berbahasa Indonesia sebagai pembaca email

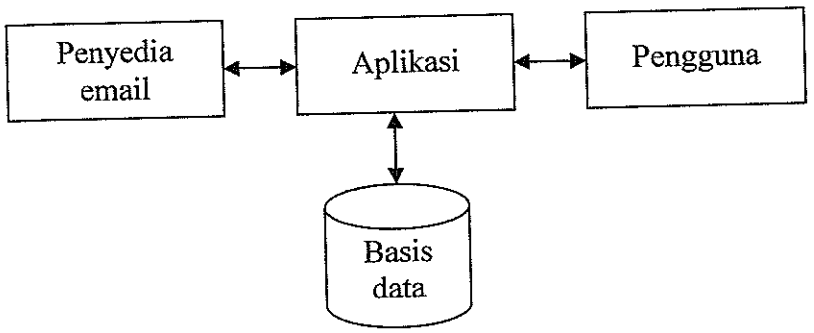
Dari gambar diagram alir diatas, urutan proses aplikasi dapat dijelaskan sebagai berikut: saat periode waktu telah tercapai, aplikasi akan menghubungkan diri ke penyedia email. Jika koneksi dengan penyedia email sudah terbentuk, aplikasi akan memerika kotak masuk di penyedia email. Jika ada email baru atau email yang belum terbaca, email akan diambil ke aplikasi, kemudian koneksi dengan penyedia email akan terputus. Email yang telah diambil tadi kemudian dipisah-pisahkan sesuai dengan tabel database yang telah tersedia, kemudian disimpan. Aplikasi akan memberitahu pengguna bahwa ada email baru yang masuk, jika pengguna menghendaki email dibacakan, maka teks pesan email akan diubah menjadi ucapan, jika tidak email akan ditampilkan dalm bentuk teks. Jika pilihan keluar dipilih, maka aplikasi akan selesai.



Gambar 5.7 Diagram alir menu baca email kondisi B

5.2.3 Implementasi

Tahap implementasi merupakan realisasi dari tahap perancangan aplikasi. Pada dasarnya aplikasi ini bekerja seperti pada Gambar 5.8 berikut ini.



Gambar 5.8 Sistem kerja aplikasi

Aplikasi bekerja dengan cara mengambil email pada kotak masuk penyedia email, kemudian teks email diubah menjadi ucapan. Pada prosesnya aplikasi juga menggunakan basis data untuk membantu kerja sistem. Aplikasi bekerja dalam tiga bagian, yaitu:

1. Inisialisasi
2. Mengambil data dari penyedia email
3. Mengubah teks email menjadi ucapan

5.2.4 Inisialisasi

Proses inisialisai aplikasi diperlukan untuk menghubungkan aplikasi dengan basis data. Proses inisialisasi basis data menggunakan senarai program berikut:

```
Set    mydb    =    DBEngine.Workspaces(0).OpenDatabase(App.Path    &
"\inbox.mdb")
Set myRs = _
   mydb.OpenRecordset("inboxall", dbOpenDynaset)
Set myrs1 = _
   mydb.OpenRecordset("outbox", dbOpenDynaset)
Set myrs2 = mydb.OpenRecordset("delete", dbOpenDynaset)
Set dbinc = New ADODB.Connection
 dbinc.Open "Provider=Microsoft.Jet.OLEDB.3.51;" & _
          "Data Source=" & App.Path & "\inbox.mdb;"
```

Senarai program diatas menggunakan objek ADODB untuk melakukan koneksi pada basis data. Basis data dibuat dalam format Microsoft Access untuk memudahkan hubungan dengan Visual Basic. Dari kode diatas, basis data yang digunakan adalah inbox.mdb.

5.2.5 Mengambil data dari penyedia email

Proses pengambilan email dilakukan dengan mengambil data dari kotak masuk penyedia email. Pengambilan email menggunakan senarai program dibawah ini:

```
MAPISession1.NewSession = True
MAPISession1.SignOn
MAPIMessages1.SessionID = MAPISession1.SessionID
```

storeall

Aplikasi menggunakan komponen MAPI (*Messaging Aplication Programming Interface*) untuk prose pengambilan email dari kotak masuk penyedia email. Komponen ini bekerja dengan menghubungkan aplikasi dengan *MAPI compliant-messaging system*, seperti Microsoft Exchanged, Microsoft Mail atau Outlook Express. Komponen MAPI mempunyai dua control yaitu *MAPIsession* dan *MAPImessage*. *MAPIsession* adalah kontrol yang digunakan untuk membangun hubungan dengan penyedia email, *MAPImessage* adalah kontrol yang digunakan untuk mengakses layanan dari penyedia email, seperti membaca dan mengirim pesan email.

Selanjutnya aplikasi memanggil prosedur storeall untuk mengambil email jika ada email baru di kotak masuk penyedia email. Senarai program prosedur storeall adalah sebagai berikut.

```
Public Sub storeall()
Dim a As Integer
MAPIMessages1.FetchUnreadOnly = True
MAPIMessages1.Fetch
Dim maxcount, i As Integer
While Not rsinc.EOF
rsinc.Delete
rsinc.MoveNext
Wend
While (Not MAPIMessages1.MsgIndex > MAPIMessages1.MsgCount - 1)
Call storeallMessage
If MAPIMessages1.MsgIndex = MAPIMessages1.MsgCount - 1 Then
GoTo lineend
Else
MAPIMessages1.MsgIndex = MAPIMessages1.MsgIndex + 1
End If
Wend
lineend:
If MAPIMessages1.MsgCount > 0 Then
With MAPIMessages1
For a = 0 To .MsgCount - 1
    .MsgIndex = a
    List1.AddItem .MsgOrigDisplayName
    List1.ItemData(List1.NewIndex) = a
Next
End With
End If

If MAPIMessages1.MsgCount = 0 Then
    Frame1.Visible = False
Else
    Frame1.Visible = True
    frameinline.Visible = True
    framshowinbox.Visible = False
    IndoTTS_Say "anda mempunyai email baru"
End If
End Sub
```

Setelah email baru pada kotak masuk penyedia email telah diambil, kemudian aplikasi memanggil prosedur storeallMessage, prosedur ini akan memisah bagian-bagian email dalam tabel. Senarai program prosedur storeallMessage adalah sebagai berikut:

```
Public Sub storeallMessage()
MyTime = Time
MySecond = Second(MyTime)
rsinc.AddNew
rsinc(0) = MAPIMessages1.MsgDateReceived & ":" & MySecond
rsinc(1)  =  MAPIMessages1.MsgOrigDisplayName  &  "       <"  &
MAPIMessages1.MsgOrigAddress & ">"
rsinc(3) = MAPIMessages1.MsgNoteText
rsinc(2) = MAPIMessages1.MsgSubject
rsinc.Update
End Sub
```

Pada tabel, bagian-bagian email dipisahkan dalam empat bagian yaitu waktu dan tanggal pesan dikirim, nama dan alamat email pengirim, isi pesan dan subyek pesan.

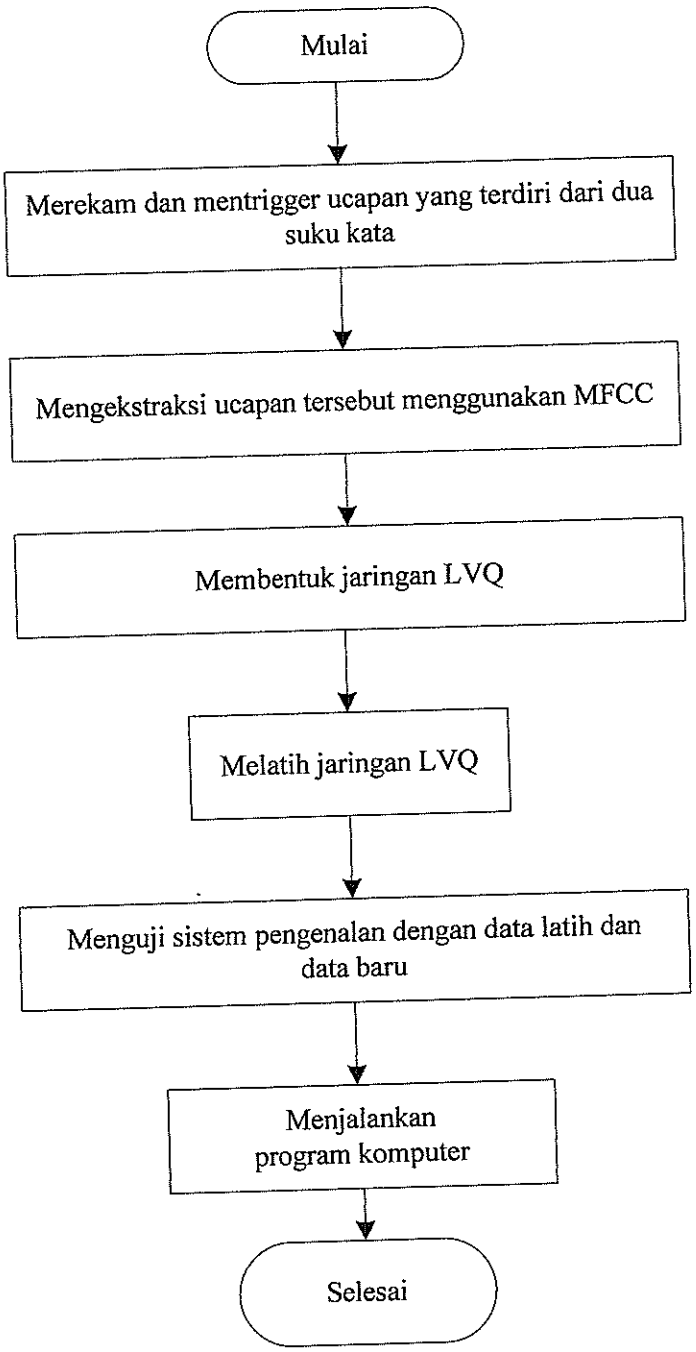
5.2.6 Mengubah teks email menjadi ucapan

Penelitian ini menggunakan MBROLA *Speech Engine* sebagai pensintesa ucapan dengan basis data diphone bahasa Indonesia yang telah tersedia. Senarai program untuk mengubah teks email menjadi ucapan adalah sebagai berikut.

```
Option Explicit
Declare Sub IndoTTS_Say Lib "ITTS_DLL.dll" Alias "IndoTTS_Say@4"
(ByVal TextToSay As String)
Declare Sub IndoTTS_Stop Lib "ITTS_DLL.dll" Alias "IndoTTS_Stop@0"
Declare   Sub   IndoTTS_ProsodyON   Lib   "ITTS_DLL.dll"   Alias
"IndoTTS_ProsodyON@0" ()
Declare   Sub   IndoTTS_ProsodyOFF   Lib   "ITTS_DLL.dll"   Alias
"IndoTTS_ProsodyOFF@0" ()
Declare   Sub   IndoTTS_SpeakON   Lib   "ITTS_DLL.dll"   Alias
"IndoTTS_SpeakON@0" ()
Declare   Sub   IndoTTS_SpeakOFF   Lib   "ITTS_DLL.dll"   Alias
"IndoTTS_SpeakOFF@0" ()
Declare  Sub  IndoTTS_SetPitchRatio  Lib  "ITTS_DLL.dll"  Alias
"IndoTTS_SetPitchRatio@4" (ByVal PRatio As Single)
Declare  Function  IndoTTS_isPlaying  Lib  "ITTS_DLL.dll"  Alias
"IndoTTS_IsPlaying@4" () As Boolean
Private Sub cmducap_Click()
 On Error GoTo errHandler
   IndoTTS_Say rt2.Text
      Exit Sub
errHandler:
   MsgBox Err.Number, vbOKOnly
   Command2.Enabled = True
End Sub
```

5.3 Pengenal Ucapan untuk Menjalankan Program Komputer

Dalam pembuatan program dibagi menjadi dua bagian utama, bagian pertama adalah proses perekaman dan mentrigger, proses pengambilan ciri sinyal, proses pengenalan dan pembelajaran, dan bagian terakhir adalah proses menjalankan komputer itu sendiri yang mana proses kesemuanya itu dilakukan dalam proses *real time*. Secara umum pembuatan program aplikasi dapat dilihat pada Gambar 5.9.



Gambar 5.9 Alur pembuatan program aplikasi pengenalan ucapan untuk menjalankan komputer

5.3.1 Perekaman dan Mentrigger Ucapan

Ucapan yang akan dijadikan objek dalam pembuatan program direkam sekaligus ditrigger sebagai data masukan bagi proses pengenalan dan pengujian. Data masukan ucapan diperoleh melalui *microphone* secara *real time*. Sinyal tersebut dengan frekuensi pencuplikan (*frequency sampling*) sebesar 11025 Hz, resolusi delapan bit dan waktu pentriggeran sebanyak satu detik (11025 sampel).

Ucapan-ucapan yang digunakan ada enam kata yaitu, "caatat', "tutup", "laguu", "tiitip", "keetiik", dan "totop", dengan masing-masing kata tersebut diucapkan dalam tempo yang sama. Untuk setiap ucapan diambil sampel ucapan sebanyak enam orang. Masing-masing kata diucapkan sebanyak lima kali setiap orang dengan perincian sebagai berikut:

- Caatat : 6 x 5 orang
- Tutup : 6 x 5 orang
- Laguu : 6 x 5 orang
- Tiitip : 6 x 5 orang
- Keetiik : 6 x 5 orang
- Totop : 6 x 5 orang

Jadi, jumlah data yang diambil adalah: 6 kata x 6 x 5 orang. Kemudian dalam program aplikasi ini akan dibuat dua macam data yaitu:

a. Data Pelatihan
   Data yang digunakan adalah data seluruhnya, dimana proses ini berjalan secara *real time* dengan rincian:
   6 kata x 6 x 5 orang
b. Data Uji / Data Baru
   Data yang digunakan sebanyak enam orang yang sama dengan data latih, dimana proses ini berjalan secara *real time* juga dengan rincian:
   6 kata x 6 x 5 orang

Ketika program eksekusi mulai dijalankan maka proses perekaman terhadap sinyal masukan berupa suara dilakukan secara terus menerus sekaligus ditampilkan bentuk sinyalnya secara *real time*. Akan tetapi data tersebut tidak tersimpan. Namun jika ada sinyal masukan dengan nilai amplitudo sebesar 0,7 dalam satuan ternormalisasi maka proses pentriggeran mulai. Untuk proses perekaman, instruksinya adalah sebagai berikut:

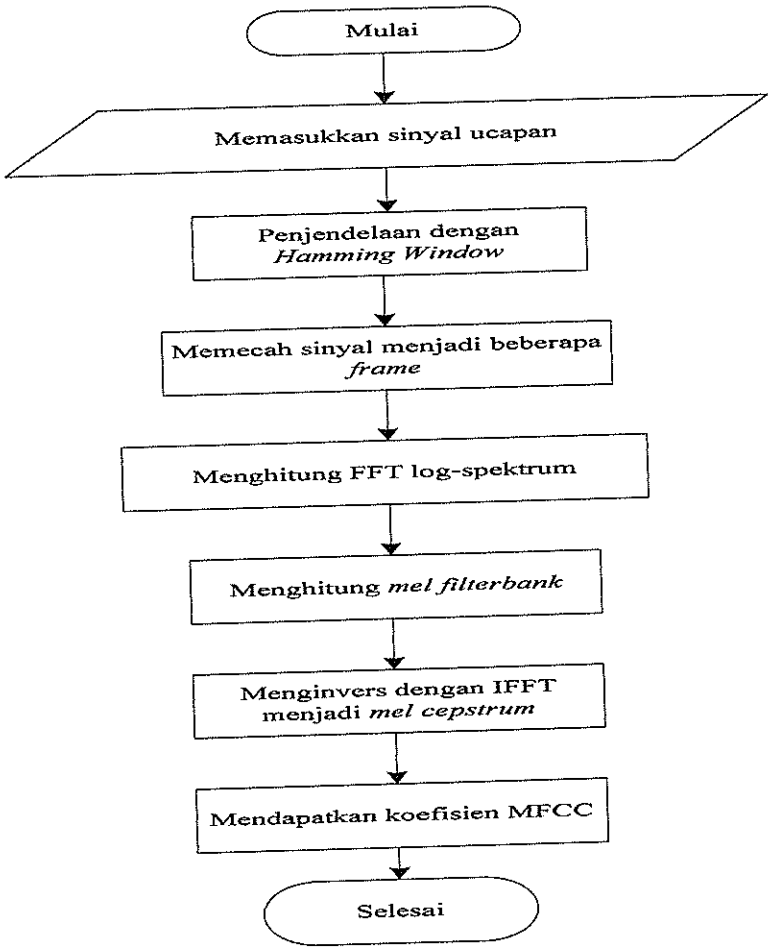
```
if BASS_RecordInit(-1) then begin
   BASS_RecordStart(11025,BASS_SAMPLE_8BITS or
   BASS_SAMPLE_MONO,RecordCallback,111);
end;
```

Sedangkan proses pentriggeran, instruksinya adalah

```
if (reclen=0) and (startrec<>0) and (now-startrec>1/(60*60*24)) then
begin
    fillchar(recbuff,sizeof(recbuff),0);
    cr:=rec;
    i:=0;
    repeat
      recbuff[i]:=buff[cr];
      inc(i);
      inc(cr);
        if cr>bufflen then cr:=0;
           until cr=curr;
        if i>10000 then reclen:=i;
end;
```

5.3.2 Analisis MFCC

Setelah sinyal ucapan direkam dan ditrigger, proses selanjutnya adalah mencari nilai koefisien-koefisien MFCC sinyal tersebut. Urutan pengolahan sinyal hingga diperoleh nilai koefisien-koefisien MFCC dapat dilihat pada Gambar 5.10. Langkah pertama adalah memasukkan sinyal ucapan, kemudian menuju langkah *frame blocking* dimana panjang *frame* menentukan jumlah segmen yang diperoleh dari pemecahan sinyal awal. Proses selanjutnya adalah melewatkan setiap segmen yang didapatkan dengan menggunakan *hamming window* dan kemudian di FFT log-spektrum. Hasil FFT log-spektrum ini diproses dengan filter bank segitiga skala mel dan yang terakhir diinverskan dengan IFFT sehingga diperoleh koefisien-koefisien yang disebut dengan koefisien MFCC. Koefisien-koefisien ini kemudian disimpan untuk digunakan pada proses berkutnya. Listing dari program pemrosesan sinyal ini merujuk pada aho.ch/fft.



Gambar 5.10 Alur ekstraksi ciri sinyal bunyi

5.3.3 Penentuan Koefisien-Koefisien MFCC

Pada saat perekaman ucapan secara tidak *real time* akan terjadi permulaan sinyal yang berbeda-beda untuk setiap ucapan. Padahal LVQ akan mengklasifikasikan sinyal berdasarkan vektor-vektor yang berdekatan, dan juga jumlah vektor harus sama. Karena proses perekaman secara *real time*, sistem akan

mentrigger sinyal ucapan saat sinyal masukan tersebut dengan nilai amplitudo tertentu dalam hal ini sebesar 0,7 dalam skala ternormalisasi sehingga tidak perlu proses pemotongan permulaan sinyal yang dianggap sebagai *noise*[22].

Setelah sinyal ucapan direkam dan ditrigger, kemudian sinyal tersebut diolah dengan ekstraksi fitur MFCC. Untuk mendapatkan koefisien-koefisien MFCC dipakai prosedur yang memiliki urut-urutan instruksi sebagai berikut:

```
procedure initialize(windowtype:string;il,ol:integer);
var freq,j,i :integer;
    sqrt2n,melcenter:single;
begin
     sampling:=11025;
     inputlength:=il;
     outputlength:=ol;
     pslength:=inputlength div 2;
     mellength:=outputlength;
     setlength(window,inputlength);
     setlength(filters,outputlength);
     setlength(filterstart,outputlength);

     if (windowtype = 'HANNING') then
        for i:=0 to inputlength-1 do
            window[i] :=0.5-0.5*cos((2*M_PI*i)/inputLength)
               else if (windowtype = 'HAMMING') then
                       for i:=0 to inputlength-1 do
                       window[i]:=0.54-0.46*cos((2*M_PI*i)/inputLength);

     setlength(tmpBuffer1,inputLength);
     setlength(tmpBuffer2CPLX,inputLength);
     setlength(tmpBuffer2,inputLength);
     high:=11025;
     low:=10;
     niquist := sampling / 2.0;
     highMel := 1000*log10(1+high/700)/log10(1+1000.0/700);
     lowMel  := 1000*log10(1+low/700)/log10(1+1000.0/700);
     setlength(centers,mellength+2);
        for i:=0 to mellength+1 do begin
                       melCenter := lowMel + i*(highMel-
       lowMel)/(melLength+1);
                          centers[i] := floor(0.5+
       psLength*700*(exp(melCenter*log10(1+1000.0/700.0)/1000)-
       1)/niquist);
       end;
       for i:=0 to mellength-1 do begin
           filterStart[i] := centers[i]+1;
           setlength(filters[i],centers[i+2]-centers[i]-1);
           j:=0;
            for freq:=centers[i]+1 to centers[i+1] do begin
                filters[i][j] := (freq-centers[i])/(centers[i+1]-
enters[i]);
                inc(j);
            end;
            for freq:=centers[i+1]+1 to centers[i+2]-1 do begin
                filters[i][j] := (centers[i+2]-freq)/(centers[i+2]-
                centers[i+1]);
                 inc(j);
            end;
        end;

      setlength(inputCopy,melLength);
      setlength(outputCopy,melLength);
      setlength(outputCopyCPLX,melLength);
      setlength(rNormalize,melLength);
      setlength(iNormalize,melLength);
```

```
      sqrt2n:=sqrt(2.0/inputLength);
      for i:=0 to melLength-1 do begin
          rNormalize[i]:=cos(M_PI*i/(2*melLength))*sqrt2n;
          iNormalize[i]:=-sin(M_PI*i/(2*melLength))*sqrt2n;
      end;
          rNormalize[0] := rNormalize[0]/sqrt(2.0);
end;

procedure calculate(inp:array of single);
var filtstart,filtersize,i,j:integer;
begin
      if (length(inp)<> inputLength) then exit;
      setlength(output,outputlength);
      for i:=0 to inputLength-1 do
          tmpBuffer1[i]:=inp[i]*window[i];
          FFT(tmpBuffer1, tmpBuffer2CPLX);
          RealOfCplx(tmpbuffer2CPLX,tmpbuffer2);
          tmpBuffer2[0]:=tmpBuffer2[0]*tmpBuffer2[0];
          for i:=1 to psLength-1 do
          tmpBuffer2[i] := tmpBuffer2[i]*tmpBuffer2[i]+
          tmpBuffer2[inputLength-i]*tmpBuffer2[inputLength-i];
      for i := 0 to length(filters)-1 do begin
           tmpBuffer1[i]:=0;
           filterSize := length(filters[i]);
           filtStart := filterStart[i];
                for j:=0 to filterSize-1 do
                    tmpBuffer1[i] := tmpBuffer1[i] +
                    filters[i][j]*tmpBuffer2[j+filtStart];
      end;
      j:=0;
      i:=0;
      while i<melLength-1 do begin
         inputCopy[j]:=log10(tmpBuffer1[i]+FLT_MIN);
         i:=i+2;
         j:=j+1;
      end;
      i:=mellength-1;
      while i>=0 do begin
         inputCopy[j]:=log10(tmpBuffer1[i]+FLT_MIN);
         i:=i-2;
         j:=j+1;
      end;

      FFT(inputCopy, outputCopyCPLX);
      RealofCplx(outputCopyCplx, outputCopy);
      for i:=1 to melLength div 2-1 do begin
           output[i]:=rNormalize[i]*outputCopy[i] -
           iNormalize[i]*outputCopy[melLength-i];
               output[melLength-i]:=rNormalize[melLength-i]*outputCopy[i]
           +   iNormalize[melLength-i]*outputCopy[melLength-i];
      end;
       output[melLength div 2] := outputCopy[melLength div
       2]*rNormalize[melLength div 2];
      end;
  end.
```

5.3.4 Pembuatan Jaringan Saraf Tiruan

Setelah melalui proses ekstraksi selanjutnya nilai-nilai yang dihasilkan dimasukkan kedalam jaringan saraf tiruan dengan menggunakan metode pembelajaran LVQ. Instruksi yang digunakan adalah:

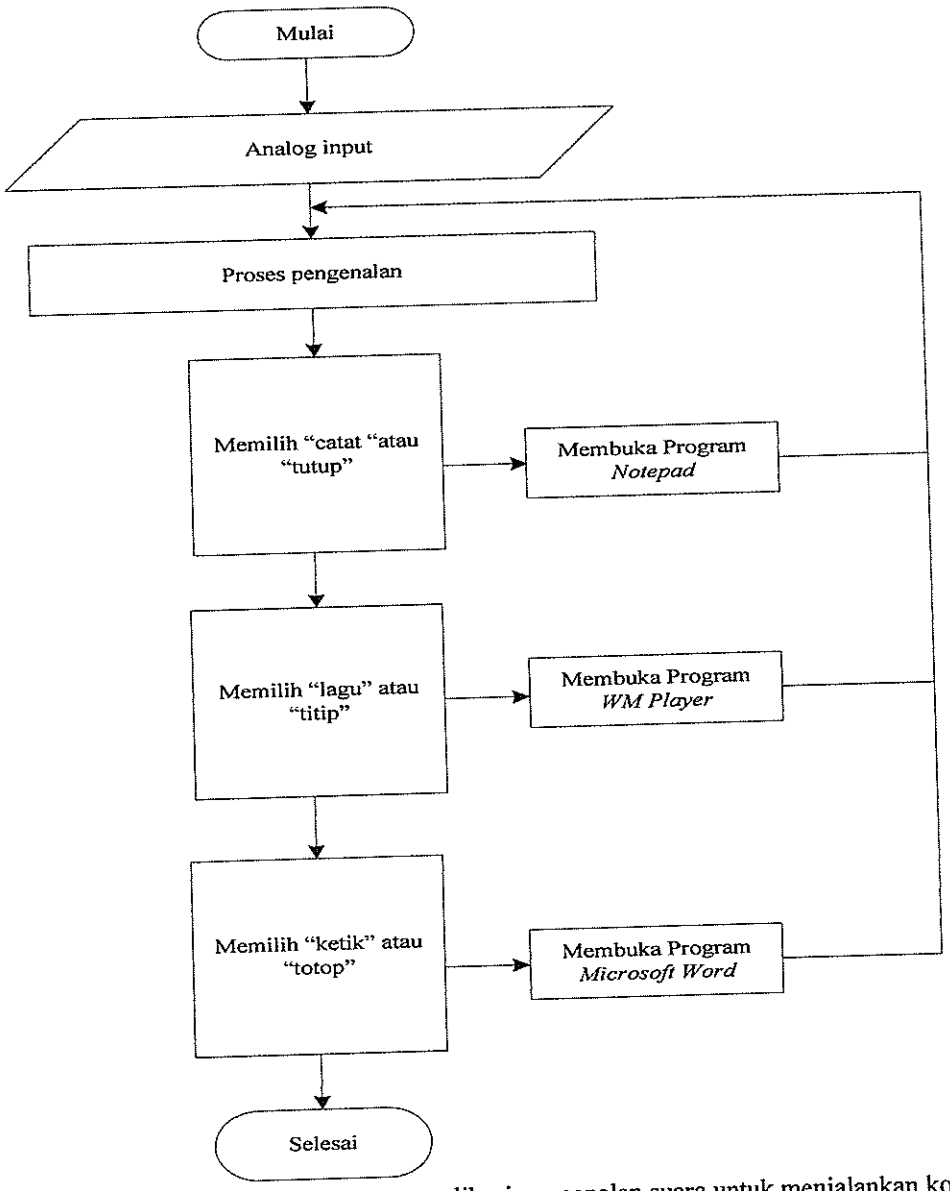
```
function kenali(inputs:array of single):byte;
var i,j,t:integer;
    choose,n,x,y:integer;
    small,d:double;
    jml:double;
    bbt:double;
begin
    fillchar(hasil,sizeof(hasil),0);
    small := 10000;
    choose := 0;
    for t := 0 to vsize-1 do begin
        bbt := 0;
        jml:=0;
        for n := 0 to hsize-1 do jml:=jml+bobot[n,t];
            if jml=0 then continue;
               for n := 0 to hsize-1 do begin
                   d:=inputs[n];
                   bbt:=bbt+sqr(d-bobot[n,t]);
               end;
                   bbt:=sqrt(bbt);
                   hasil[t]:=bbt;
               if small>bbt then begin
                   small:=bbt;
                   choose:=t;
                end;
            end;
          result:=choose;
     end;
    procedure belajar(ch:byte;data:array of single);
    var epoh,n,x,y:integer;
       alpha,d:double;
       t,tanda:integer;
       awal:boolean;
       s:string;
        c:byte;
    begin
    alpha:=0.1;
    awal:=true;
       for t:=0 to hsize-1 do begin
         if bobot[t,ch]<>0 then begin
            awal:=false;
            break;
         end;
       end;
       tanda:=1;
       if not awal then begin
           c:=kenali(data);
           if c<>ch then begin
            s:='Tanda  rumus  (-)  karena  huruf  lebih  dikenali  sebagai
'+kenals[c];
            application.messagebox(pchar(s),'Keterangan');
            tanda:=-1
           end else tanda:=1;
        end;
      if awal then application.messagebox('Ini baru proses pemasukan nilai
         awal !','Keterangan');
             for epoh:=1 to 20 do begin
                 for n:=0 to hsize-1 do begin
                     d:=data[n];
```

```
            if not awal then
              bobot[n,ch]:=bobot[n,ch]+tanda*alpha*(d-bobot[n,ch])
              bobot[n,ch]:=d;
            end;
            alpha:=alpha-0.1*alpha;
        end;
end;
```

Dalam proses pembelajaran akan dibagi menjadi dua yaitu pembelajaran data latih dan pembelajaran data baru/uji dimana kedua data tersebut secara berjalan secara *real time*.

5.3.5 Proses Menjalankan Aplikasi

Dalam proses ini menggunakan suatu *tool* berupa analog input sebagai proses awal dalam proses pengenalan selanjutnya, sebagaimana digambarkan dalam Gambar 5.11.



Gambar 5.11 Alur program utama aplikasi pengenalan suara untuk menjalankan komputer

Proses menjalankan aplikasi program komputer dilakukan dengan memasukkan sinyal ucapan berupa kata tertentu yang telah diproses pada bagian utama awal, yaitu:

- Memasukkan kata "caatat" untuk membuka program *Notepad*
- Memasukkan kata "tutup" untuk menutup program *Notepad*
- Memasukkan kata "laguu" untuk membuka program *Windows Media Player*
- Memasukkan kata "tiitip" untuk menutup program *Windows Media Player*
- Memasukkan kata "keetiik" untuk membuka program *Microsoft Word*
- Memasukkan kata "totop" untuk menutup program *Microsoft Word*

Proses membuka adalah dengan statemen kendali dimana bila ada suatu karakter ":" pada file "buka.txt" maka ia akan melakukannya, yang instruksi adalah sebagai berikut:

```
if pos(':',uppercase(buka[i]))>0 then begin
          if app[i]>31 then begin
             maututup:=buka[i+3];
             ada:=false;
             EnumWindows(@CekApps, 0);
             if not ada then app[i]:=0 else
             SendMessage(adawnd,WM_CLOSE,0,0);
             app[i]:=0;
          end;
          if app[i]=0 then app[i]:=winexec(pchar(buka[i]),sw_show)
       end else begin
```

Sedangkan instruksi menutup adalah sebagai berikut:

```
   maututup:=buka[i];
          EnumWindows(@CloseApps, 0);
```

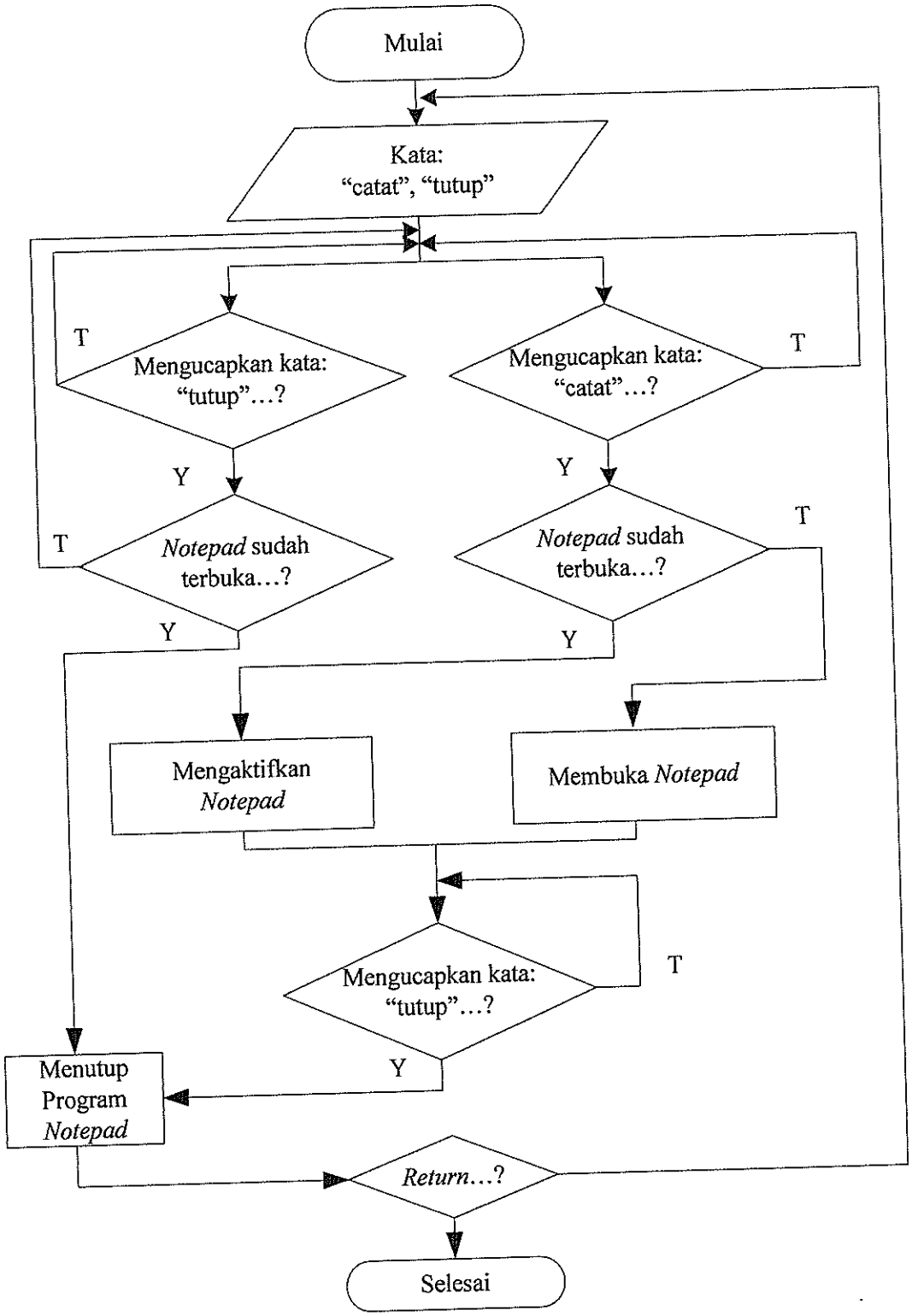
Adapun pada file "buka.txt" berisi berikut:

```
c:\windows\notepad.exe
c:\Program Files\Windows Media Player\mplayer2.exe
c:\Program Files\Microsoft Office\Office10\WINWORD.EXE
Notepad
Windows Media Player
Microsoft Word
```

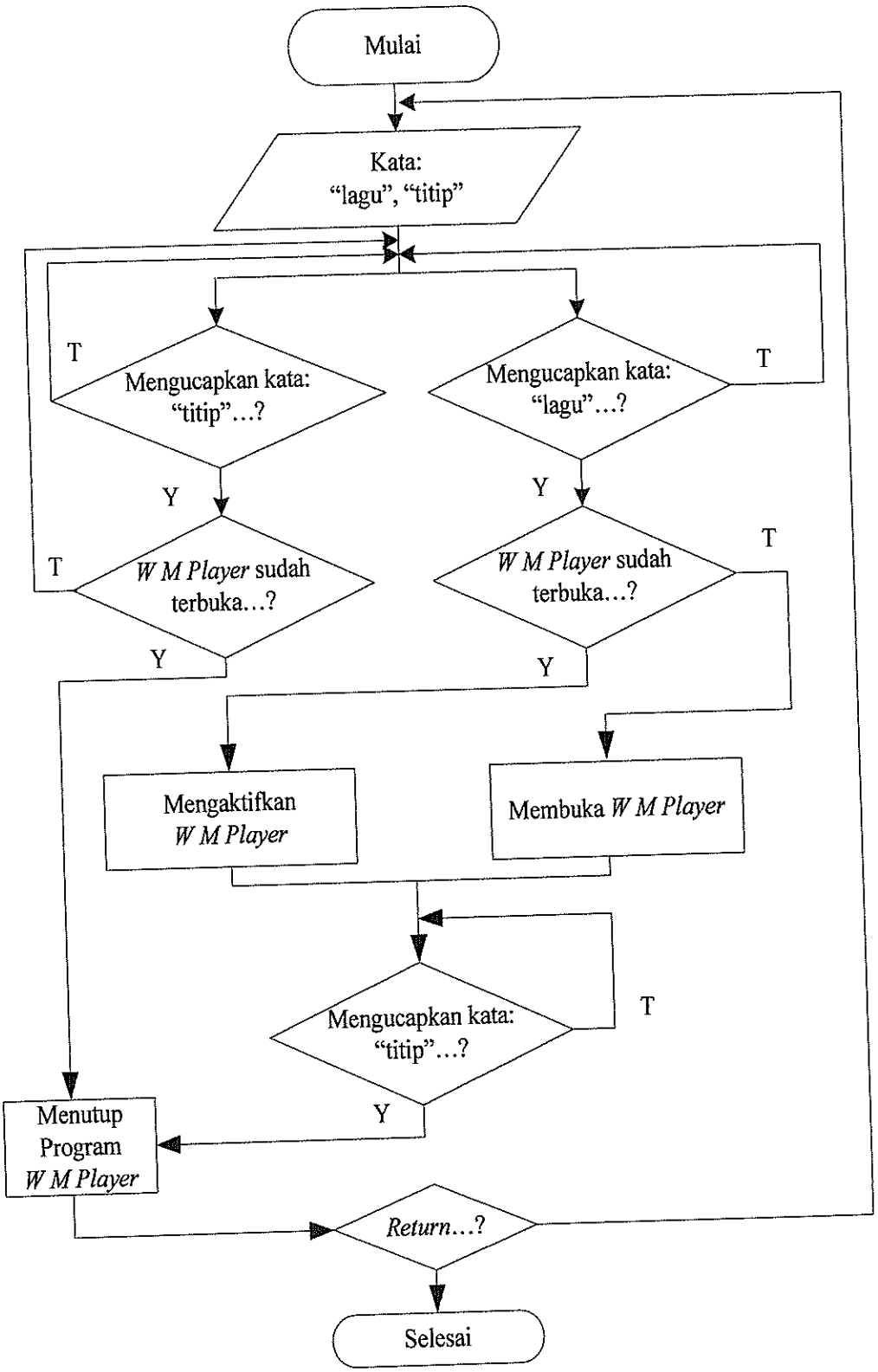
Data berupa kata-kata yang dideteksi untuk ditampilkan pada daftar *scroll* pada file "kenali.txt" sebagai berikut:

```
caatat
laguu
keetiik
tutup
tiitip
totop
```

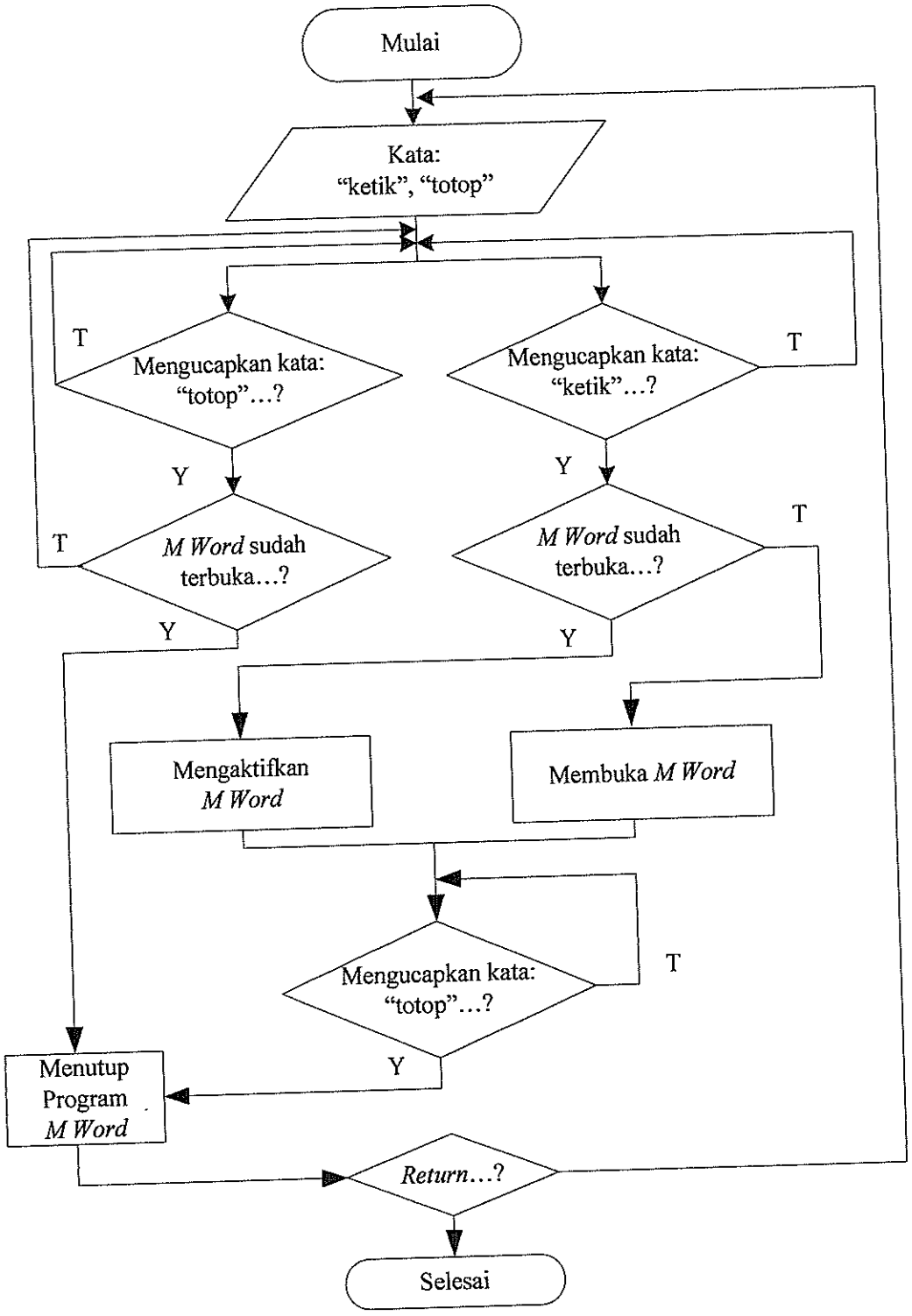
Microsoft Windows merupakan salah satu sistem operasi yang *multitasking* yaitu dapat menjalankan beberapa program namun hanya satu *window* yang aktif. Jadi, dalam program aplikasi ini dapat mengaktifkan program komputer (*window*) yang mana program komputer (*window*) ini sebelumnya sudah dijalankan, namun tidak aktif sebagaimana diperlihatkan pada Gambar 5.12, 5.13 dan 5.14. Misalkan dalam kasus ini, program *Notepad*, program *Windows Media Player*, program *Microsoft Word* sudah dijalankan, sedangkan program *Notepad* yang sedang aktif, maka pada program aplikasi ini dapat mengaktifkan program *Windows Media Player* yang sebelumnya tidak aktif, dengan mengucapkan kata"laguu".



Gambar 5.12 Alur sub program aplikasi pengenalan suara untuk menjalankan Program *Notepad*



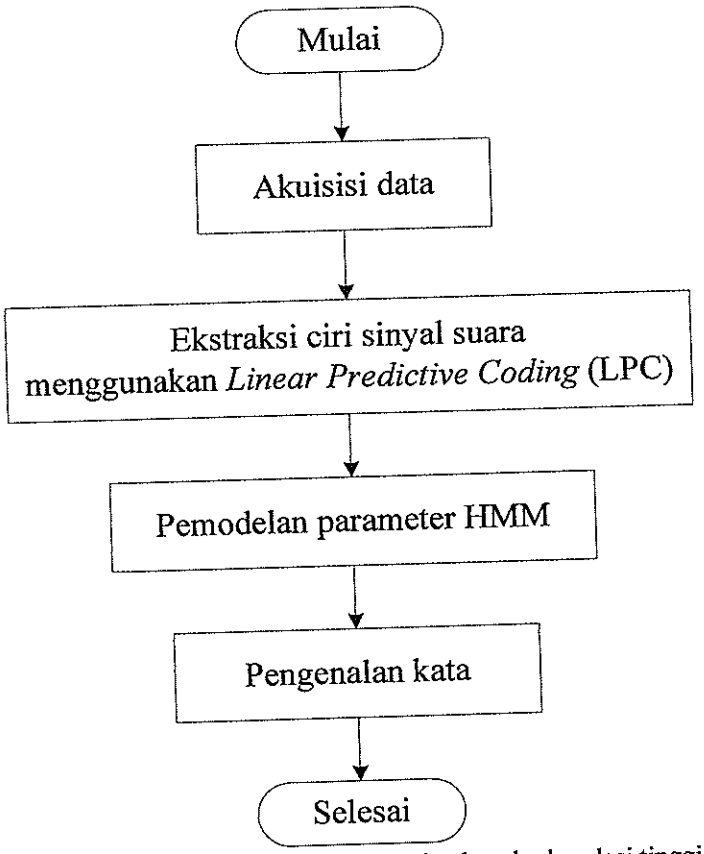
Gambar 5.13 Alur sub program aplikasi pengenalan suara untuk menjalankan Program *Windows Media Player*



Gambar 5.14 Alur sub program aplikasi pengenalan suara untuk menjalankan Program *Microsoft Word*

5.4 Pengenal Ucapan dengan Metode Hidden Markov Model

Sistem pengenalan kata dapat diwujudkan ke dalam suatu perangkat lunak (program) menggunakan bahasa pemrograman Matlab 6.5. Dalam pembuatan program simulasi pengenalan kata ini, perancangan sistem merupakan tahap yang penting. Perancangan bertujuan agar dalam pembuatannya dapat berjalan secara sistematis, terstruktur dan rapi sehingga hasil program dapat berjalan sesuai dengan apa yang dikehendaki. Secara umum pembuatan program simulasi ini mengikuti alur sesuai yang ditunjukkan pada Gambar 5.15.



Gambar 5.15 Alur program pengenalan kata berkorelasi tinggi.

Dari Gambar 5.15 dapat dilihat bahwa pembuatan program simulasi pengenalan kata di bagi menjadi 4 tahap, tahap pertama adalah akuisisi data, tahap kedua adalah proses ekstraksi ciri sinyal suara menggunakan LPC, tahap ketiga adalah proses pemodelan parameter HMM dan tahap yang keempat adalah pengenalan angka.

5.4.1 Akuisisi Data

Data berupa sinyal suara diperoleh dengan cara merekam suara melalui mikropon yang dihubungkan dengan komputer pribadi. Perekaman suara dilakukan dengan bantuan program aksesories *windows* yaitu *sound recorder* dengan frekuensi *sampling* 8000 Hz, 8 bit, mono. Kata yang akan diucapkan terdiri dari 8 buah kata bahasa Indonesia, yaitu : muka, muak, kamu, kaum, masuk, kamus, kusam dan sukma.

Suara diucapkan oleh 20 orang yang terdiri dari sepuluh orang pria dan sepuluh orang wanita dimana untuk setiap kata diulang sebanyak 10 kali. Dengan *sound recorder* suara direkam dan berkas suara dibuka kembali dengan Cool Edit Pro 2.0 untuk dipotong kemudian disimpan dalam berkas dengan format penamaan ABC.wav. Huruf A menunjukkan nama responden, huruf B menunjukkan kata yang diucapkan dan huruf C menunjukkan suara yang ke-(1-10). Format penamaan ini dimaksudkan untuk mempermudah dalam pengolahan berkas. Untuk pembacaan berkas .wav dalam Matlab dapat dilakukan dengan fungsi wavread.

```
data = wavread('ABC.wav');
```

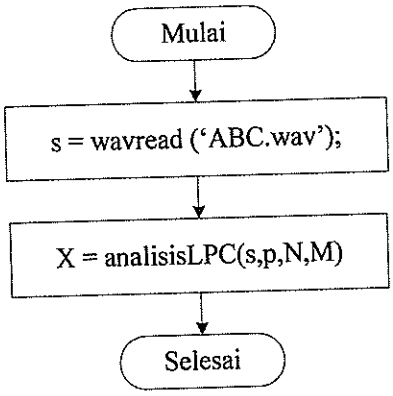
Contoh diatas adalah untuk pembacaan satu berkas suara saja. Jika menggunakan perintah seperti diatas untuk keseluruhan data yang terdiri dari 8 buah kata dan setiap kata terdiri dari 200 suara, maka diperlukan 1600 baris perintah. Untuk tujuan penyederhanaan pembacaan keseluruhan berkas .wav dapat dilakukan dengan perintah sebagai berikut:

```
load nama;

for nm=1:20
        A=nama{nm};
    .....
    for b=1:8
        .....
        k={'Kamu','Kamus','Kaum','Kusam','Masuk','Muak',
           'Muka','Sukma'};
        B=k{1,b};
        nf=strcat(A,B,int2str(C),'.wav');
        dat=wavread(nf);
        .....
     end
    .....
end
```

Variabel nama merupakan sebuah berkas dalam bentuk .mat yang merupakan *cell* yang terdiri dari nama responden. Dengan perintah di atas program akan menjadi lebih sederhana.

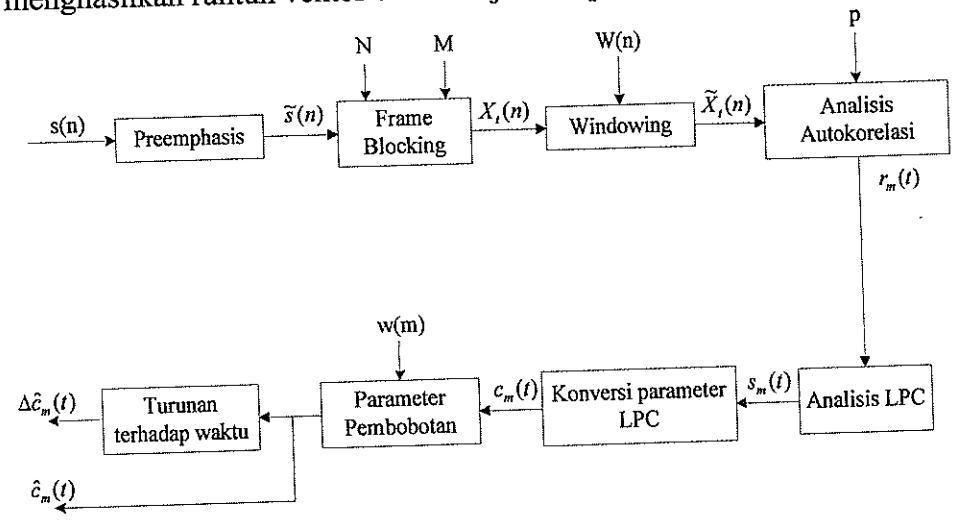
5.4.2 Ekstraksi Ciri

Setelah melalui proses perekaman, sinyal suara akan dibaca atau dipanggil kembali dengan fungsi wavread. Hasil pembacaan data untuk keseluruhan sinyal suara digunakan untuk proses selanjutnya yaitu ekstraksi ciri. Proses ekstraksi ciri digunakan untuk mencari nilai koefisien-koefisien LPC dari sinyal suara. Untuk proses ekstraksi ciri sesuai dengan alur program Gambar 5.16.



Gambar 5.16 Alur program ekstraksi ciri.

Perintah `wavread` dengan satu variabel keluaran pada Matlab akan menghasilkan keluaran berupa matrik n baris dengan satu kolom, sedangkan angka pada tiap-tiap baris menunjukkan amplitudo sinyal. Proses analisis LPC hingga menghasilkan runtun vektor ciri ditunjukkan pada Gambar 5.17.



Gambar 5.17 Blok diagram ekstraksi ciri dengan analisis LPC.

Dari Gambar 5.17 dapat dilihat data dari hasil pembacaan berkas s(n) melewati proses *preemphasis* dengan perintah `sf = filter([1 -0.95],1,s)`. Setelah melalui proses *preemphasis*, sinyal suara dibagi dalam beberapa *frame* berisi N *sample* suara dengan jarak antara *frame* yang berdekatan dipisahkan oleh M *sample*. Setiap *frame* kemudian dijendelakan menggunakan *hamming window*, dianalisis autokorelasi, analisis LPC, konversi parameter LPC menjadi koefisien cepstral, pembobotan parameter dan yang terakhir koefisisen cepstral diturunkan terhadap waktu. Semua proses tersebut dapat dilakukan dengan satu fungsi analisisLPC.

Untuk memperjelas proses ekstraksi ciri dengan perintah analisisLPC, diberikan contoh sebagai berikut :

```
s = wavread('ABC.wav');
X = analisisLPC(s,p,N,M);
```

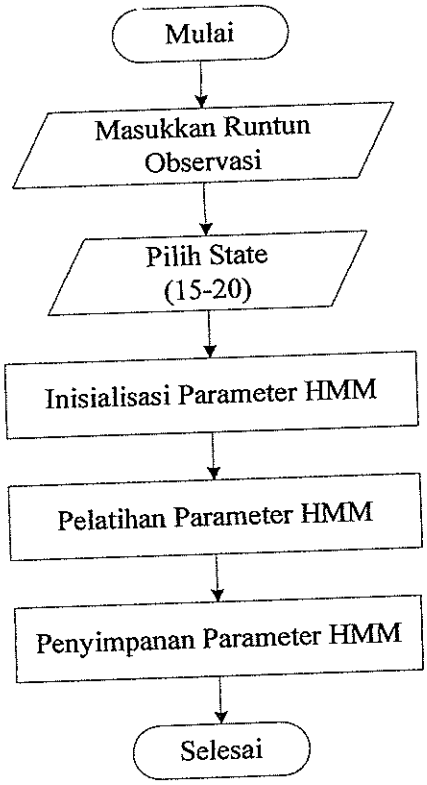
Vektor s hasil keluaran perintah `wavread` digunakan sebagai masukan dalam fungsi analisisLPC. Penjelasan lebih detail tentang variabel masukan yang digunakan dalam fungsi analisisLPC adalah sebagai berikut :

s = matrik berkas suara
p = orde LPC, dengan nilai 10
N = jumlah *sample* tiap *frame*, dengan nilai 400
M = jarak antara *frame* yang berurutan, dengan nilai 100

Dengan kombinasi variabel masukan, maka akan dihasilkan keluaran berupa matrik X dengan MxN, M menunjukkan banyaknya *frame*, sedangkan N menunjukkan bahwa setiap baris terdiri dari koefisien cepstral dan turunan koefisien cepstral terhadap waktu. Hasil dari vektor ciri ini sangat berguna untuk proses pemodelan, pelatihan dan pengenalan.

5.4.3 Pemodelan Parameter HMM

Untuk mendapatkan parameter HMM melalui lima tahap, yaitu : memasukkan runtun *observasi* hasil dari proses ekstraksi ciri, memilih *state*, inisialisasi parameter HMM, pelatihan HMM, pelatihan parameter HMM dengan tujuan untuk mendapatkan parameter yang lebih baik dan penyimpanan. *State* untuk pemodelan parameter HMM dapat dipilih dari *state* 15 sampai 20. Proses untuk mendapatkan parameter HMM ditunjukkan pada Gambar 5.18.



Gambar 5.18 Bagan pemodelan parameter HMM.

5.4.4 Runtun Observasi

Data pelatihan yang digunakan pada program simulasi ini adalah vektor ciri suara dari semua responden masing-masing setiap kata terdiri dari lima suara untuk tiap responden. Jadi data pelatihan untuk masing-

masing kata terdiri dari 100 (20x5) vektor ciri suara yang kemudian dijadikan dalam satu vektor. Untuk mendapatkan data pelatihan tersebut digunakan fungsi Latih_data. Untuk penjelasan fungsi Latih_data adalah sebagai berikut:

```
[data,st]=Latih_data (k,akhir);
```

Variabel input:

k = angka 1-8 yang menggantikan kata kamu, kamus, kaum, kusam, masuk, muak, muka dan sukma.

akhir = akhir dari data pelatihan.

Untuk lebih jelasnya ditunjukkan oleh contoh program sebagai berikut:

```
[data,st]=Latih_data(1,5);
```

Dari contoh di atas akan diperoleh data pelatihan untuk kata kamu yang terdiri dari vektor ciri suara 1 sampai 5. Sebagai catatan bahwa fungsi Latih_data hanya akan bekerja untuk penamaan berkas seperti yang dijelaskan sebelumnya.

5.4.5 Inisialisasi Parameter HMM

Untuk tahap ini akan dilakukan inisialisasi parameter mu, sigma dan A yang merupakan probabilitas transisi. Untuk inisialisasi ketiga parameter ini digunakan alur program sebagai berikut:

```
DIAG_COV = 1;
QUIET = 1;

A = sparse(0.85*diag(ones(1,N))+0.15*diag(ones(1,N-1),1));
A(N,N) = 1;

[mu,Sigma] = hmm_mint(X, st, N, DIAG_COV,QUIET);
Sigma = ones(N,1)*mean(Sigma);
```

Variabel N pada program di atas merupakan *state* yang telah dipilih sebelumnya. Dari alur program di atas akan didapatkan parameter A, mu dan sigma yang akan diestimasi ulang dalam pelatihan parameter HMM.

5.4.6 Pelatihan Parameter HMM

Dalam proses pelatihan ini parameter yang sudah didapatkan dari hasil inisialisasi parameter akan diestimasi sampai NIT. Alur program untuk proses estimasi parameter HMM adalah seperti dibawah ini.

```
logl = zeros(1, NIT);

for n = 1:NIT
    [tmp, logl(n), gamma] = hmm_mest(X, st, A, mu, Sigma, QUIET);
    [mu, Sigma] = mix_par(X, gamma, DIAG_COV, QUIET);
    Sigma = ones(N,1)*(sum((sum(gamma)'*ones(1,p)).*Sigma)/T);
End

    sigma = Sigma(1,:);
    muu=mu;
    sigm=sigma;
    A=tmp;
```

Dari alur program di atas akan didapatkan parameter A, muu yang merupakan mu, dan sigm yang merupakan sigma. Selanjutnya ketiga parameter ini akan disimpan dalam direktori Matlab.

5.4.7 Penyimpanan Parameter

Penyimpanan parameter diperlukan karena parameter ini akan berfungsi untuk proses pengenalan kata. Adapun untuk melakukan penyimpanan parameter digunakan alur program sebagai berikut :

```
yy = strcat('par',int2str(N),int2str(rr));
eval(['save''' yy''' A''' muu''' sigm']);
```

Baris pertama berfungsi untuk membuat sebuah *string* dengan format

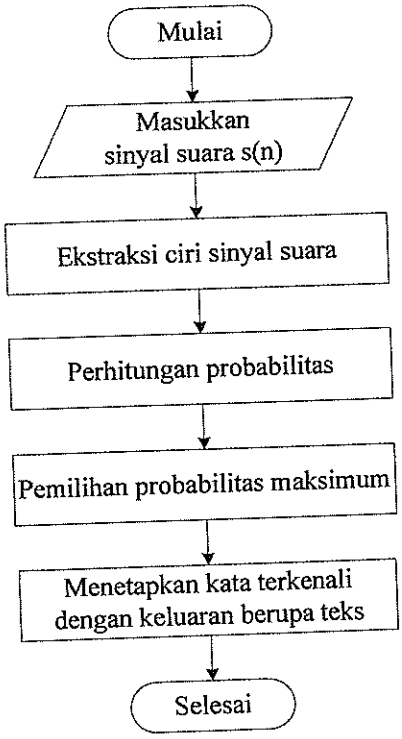
```
parNrr
```

N dalam format *string* di atas menunjukkan *state* sedangkan rr menunjukkan kata kamu, kamus, kaum, kusam, masuk, muak, muka dan sukma (1-8). Dari keseluruhan program untuk pemodelan parameter HMM dapat dilihat pada fungsi cari_par sebagai berikut :

```
for N=15:20;
for rr=1:8;

  [X,st]=Latih_data(rr,5);
  [T p] = size(X);
  DIAG_COV = 1;
  QUIET = 1;
  NIT = 10;

    A = sparse(0.85*diag(ones(1,N))+0.15*diag(ones(1,N-1),1));
    A(N,N) = 1;
    [mu,Sigma] = hmm_mint(X, st, N, DIAG_COV,QUIET);
    Sigma = ones(N,1)*mean(Sigma);
    logl = zeros(1, NIT);

  for n = 1:NIT
    [tmp, logl(n), gamma] = hmm_mest(X, st, A, mu, Sigma, QUIET);
    [mu, Sigma] = mix_par(X, gamma, DIAG_COV, QUIET);
    Sigma = ones(N,1)*(sum((sum(gamma)'*ones(1,p)).*Sigma)/T);
  end

  sigma = Sigma(1,:);
  muu=mu;
  sigm=sigma;
  A=tmp;

yy=strcat('par',int2str(N),int2str(rr));
eval(['save' ' ' yy ' ' 'A' ' ' 'muu' ' ' 'sigm']);
end
```

5.4.8 Pengenalan Kata

Proses pengenalan kata dibagi menjadi 5 tahap, tahap pertama adalah memasukkan sinyal suara s(n), tahap kedua adalah proses ekstraksi ciri sinyal suara, tahap ketiga menghitung probabilitas dari runtun observasi hasil proses ekstraksi ciri, tahap keempat memilih hasil probabilitas maksimum dan tahap kelima menetapkan kata terkenali dengan keluara berupa teks. Diagram proses pengenalan kata ditunjukkan pada Gambar 5.19.



Gambar 5.19 Bagan proses pengenalan kata.

Sinyal suara s(n) dimasukkan ke dalam ekstraksi ciri sehingga didapatkan vektor ciri, dari vektor ciri ini kemudian ditentukan probabilitas terhadap semua kata dan dipilih nilai probabilitas maksimum yang akhirnya nilai maksimum tersebut diubah menjadi keluaran berupa teks.

Untuk proses perhitungan probabilitas dan proses perhitungan maksimum terdapat dalam satu fungsi yaitu cari_maksim. Untuk penggunaan fungsi cari_maksim adalah seperti dibawah ini:

```
[maksim,score]=cari_maksim(s,N);
```

Variabel input s merupakan sinyal suara s(n), sedangkan N adalah jumlah *state*. Untuk variabel keluaran maksim merupakan bilangan yang menunjukkan hasil keputusan probabilitas maksimum. Adapun fungsi cari_maksim adalah sebagai berikut :

```
function [maksim,score]=cari_maksim(s,N)

X=analisisLPC(s,10,400,100);
QUIET = 1;
score=zeros(1,8);
for k=1:8
   eval(['load par' int2str(N) int2str(k)]);
   Sigma = ones(N,1)*sigm;
   score(1,(k+1)) = hmm_vit(X, A, [1 zeros(1,N-1)], muu, Sigma,
                            QUIET);

end
kata = {'kamu','kamus','kaum','kusam','masuk','muak',
        'muka','sukma'};
[m,maksim]=max(score);
maks=(maksim);
maksim=kata{1,maks};
```

DAFTAR PUSTAKA

Brookes, Mike, *Voicebox: Speech Processing Toolbox for Matlab*, Imperial College, London.

2. Davis, S.B. and Mermelstein, P., *Comparison of Parametric Representations for Monosyllabic Word Recognition In Continuously Spoken Sentences, IEEE Transactions on Acoustics, Speech, Signal Processing*, Vol. ASSP-28, No. 4 August 1980
3. Do, Minh N., *Digital Signal Processing Mini Project -An Automatic Speaker Recognition System*, Audio Visual Communications Laboratory Swiss Federal Institute of Technology, Lausanne, Switzerland, 1998
4. Fulop, Sean A., *Approaches to Feature Extraction*, The University of Chicago, Chicago, April 09, 2003
5. Gold, Ben, and Morgan, Nelson, *Speech Audio Signal Processing, Processing and Perception of Speech and Music*, John Wiley & Sons, Inc, New York, 2000.
6. Rabiner, L., *A Tutuorial on Hidden Markov Model and Selected Application in Speech Recognition*, vol. 77, no. 2, pp. 257-286, IEEE, 1989
7. Rabiner, Lawrence, and juang, Biing Hwang, *Fundamentals of Speech Recognition*, Prentice-Hall International, Inc, New Jersey, 1993.
8. Hirsch H.G., Hellwig K., and Dobler S., *Speech Recognition at Multiple Sampling Rate*, Ericsson Eurolab Deutschland, Aalborg, Denmark, 2001
9. Jayadi, Sri, *Pengenalan Angka Terisolasi dengan Menggunakan HMM Melalui Ekstraksi Ciri Mel-Cepstrum Filter Bank*, Jurusan Teknik Elektro Universitas Diponegoro, Tugas Akhir, 2003
10. Howard, D., Mark, B., *Neural Network ToolBox For Use With MATLAB*
11. Ikuta, Hisatoshi, Yoshihisa Ishida, Takashi Honda, and Yasuhiko Arai, *Spoken Word Recognition Using LVQ Based on DP Matching*, Meiji University, Yokohama.

12. Ivana, *Pengenalan Ucapan Vokal Bahasa Indonesia Dengan Jaringan Saraf Tiruan Menggunakan Linear Predictive Coding*, Jurusan Teknik Elektro Universitas Diponegoro, Tugas Akhir, 2003.
13. Kusuma, Aji, *Pengenalan Suara Alat Musik Dengan Metode Jaringan Saraf Tiruan (JST) Learning Vector Quantization (LVQ) Melalui Ekstraksi Linear Predictive Coding (LPC)*, Jurusan Teknik Elektro Universitas Diponegoro, Tugas Akhir, 2004
14. Kusumadewi, Sri, *Artifial Intelligence-Teknik dan Aplikasinya*, Graha Ilmu, Yogyakarta, 2003
15. Kusumadewi, Sri, *Membangun Jaringan Syaraf Tiruan Menggunakan Matlab dan Excel Link*, Graha Ilmu, Yogyakarta, 2003
16. Nurcholish, Muhammad, *Pengenalan Ucapan Fonem Menggunakan Jaringan Saraf Kuantisasi Vektor Linde, Buzo, Dan Gray*, Jurusan Fisika Fakultas MIPA, Universitas Diponegoro, Tugas Akhir, 2003.
17. Nurdian, Hendra, *Pengenalan Pengucap Bergantung Teks dengan Ekstraksi Komponen Cepstral Menggunakan Linear Predictive Coding (LPC)*, Jurusan Teknik Elektro Universitas Diponegoro, Tugas Akhir, 2003
18. Picone, Joseph, *Signal Modelling Techniques In Speech Recognition*, Texas Instruments Systems and Information Sciences Laboratory Tsukuba Research and Development Center, Tsukuba, Japan, 1992
19. Proakis, John G., and Manolakis Dimitris G., *Pemrosesan Sinyal Digital: Prinsip, Algoritma dan Aplikasi*, PT Prenhallindo, Jakarta, 1997
20. Siang, Drs. Jek Jong, *Jaringan Saraf Tiruan dan Pemrograsannya dengan menggunakan MATLAB*, Penerbit Andi Yogyakarta, Yogyakarta, 2005
21. Situmeang, Doni, *Pengenalan Vokal Bahasa Indonesia dengan Jaringan Saraf Tiruan melalui Transformasi Fourier*, Jurusan Teknik Elektro Universitas Diponegoro, Tugas Akhir, 2002.
22. Bahan dari website mengenai Fourier Transforms :
23. Yajid, Muhammad, *Analisis Pengenalan Ucapan dengan Ekstraksi Mel-Frequency Cepstrum Coefficients (MFCC) melalui Jaringan Saraf Tiruan (JST) Learning Vector Quantization (LVQ)*, Jurusan Teknik Elektro Universitas Diponegoro, Tugas Akhir, 2005

24. Zhang, W. Yin, Q., and Liu, Z., *A Speaker Identification and Verification System*, EEL6586 Final Project, 2002

25. Abdulla, W. H. and Kasabov, N. K., *The Concept of Hidden Markov Model in Speech Recognition*, Knowledge Engineering Lab. Information Science Department University of Otago, New Zealand, 1999.

26. Cappe, O., *H2M : A Set of Matlab/Octave Functions for The EM Estimation of Mixtures and Hidden Markov Model*, ENST dpt. TSI/LCTI (CNRS-URA 820), Paris, 2001.

27. Ellis, D., *Speech Recognition*,
Maret 2005.

28. Furui, S., *Digital Speech Processing, Synthesis, and Recognition*, Marcel Dekker, Inc., New York, 1989.

29. Kanungo, T., *Hidden Markov Models*,
Maret 2005