UNIVERSITAS INDONESIA

PERANCANGAN SOFTWARE PENDETEKSI KORONA DENGAN METODE *HIDDEN MARKOV MODEL*

SKRIPSI

JOKO HARTONO
0706199496

FAKULTAS TEKNIK` UNIVERSITAS INDONESIA
PROGRAM STUDI TEKNIK ELEKTRONIKA
DEPOK
DESEMBER 2009

UNIVERSITAS INDONESIA

PERANCANGAN SOFTWARE PENDETEKSI KORONA DENGAN METODE *HIDDEN MARKOV MODEL*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

Joko Hartono
0706199496

FAKULTAS TEKNIK` UNIVERSITAS INDONESIA
PROGRAM STUDI TEKNIK ELEKTRONIKA
DEPOK
DESEMBER 2009

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Joko Hartono**
NPM	**: 0706199496**
Tanda Tangan	**:**
Tanggal	**: Desember 2009**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Joko Hartono
NPM : 0706199496
Program Studi : Teknik Elektro
Judul Skripsi : Perancangan Software Pendeteksi Korona Dengan Metode *Hidden Markov Model*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elekt-o, FakultasTeknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Dr. Ir. Arman Djohan Diponegoro ( )

Penguji : Prof. Dr. Ir. Iwa Garniwa M. K. M.T ( )

Penguji : Prof. Dr. Ir. Harry Sudibyo M.Sc ( )

Ditetapkan di : Ruangan Multimedia A Lt.2 DTE Depok
Tanggal : Senin, 28 Desember 2009

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

(1) Ir. Arman Djohan Diponegoro, selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberi pengarahan, diskusi dan bimbingan serta persetujuan sehingga skripsi ini dapat selesai dengan baik;
(2) Ir. Budi Sudiarto yang telah memberikan data korona; dan
(3) Kedua orang tua beserta adik saya yang telah memberikan dukungan material dan moral;

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 2009

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Joko Hartono
NPM : 0706199496
Program Studi : Teknik Elektro
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

PERANCANGAN SOFTWARE PENDETEKSI KORONA DENGAN METODE *HIDDEN MARKOV MODEL*

beserta perangkat yang ada. Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 2009

Yang menyatakan

(Joko Hartono)

ABSTRAK

Nama : Joko Hartono
Program Studi : Teknik Elektro
Judul : Perancangan Software Pendeteksi Korona Dengan Metode *Hidden Markov Model*

Skripsi ini betujuan untuk merancang sebuah *software* pendeteksi korona yang terjadi pada peralatan listrik yang mengunakan tegangan tinggi. Metode *identifikasi* menggunakan *Hidden Markov Model* (HMM) yang memiliki kelebihan dalam memodelkan persamaan matematika.
Software ini meliputi 2 proses utama, yaitu *training* sebagai proses pengisian *database* dan identifikasi. Input berupa data *audio* (*.wav) yang kemudian diolah melalui beberapa tahapan diantaranya *labelisasi*, pembentukan *codebook* dan pembentukan parameter HMM. Hal yang harus diperhatikan dalam pengolahan ini adalah waktu pencuplikan, jumlah iterasi dan ukuran *codebook* yang digunakan, dimana ketiga variabel ini akan dianalis sehingga dapat diketahui nilai masing – masing parameter yang menghasilkan identifikasi dengan akurasi paling tinggi. Akurasi tertinggi yang dapat dicapai *software* ini hanya sebesar 50% dikarenakan data latih korona yang terbatas.
Kata Kunci : HMM, *codebook*, korona

ABSTRACT

Name : Joko Hartono
Study Program : Teknik Elektro
Title : Corona Detection Design Software Using *Hidden Markov Model*

This final project was made to design a corona detection that occured in the electric equipment using very hight voltage, such as electric guardhouse. Identification methode that used was Hidden Markov Model (HMM). It had an advantage in modeling mathematic equations.
This software contains 2 main proces, *training* as filling in the database and *identification*. The input is audio data which format is (*.wav) then processed pass through many steps, such as : labelisation, forming the *codebook* and HMM parameters. Factor that influenced to the accuration as the result of the software is duration time, amount of iteration and codebook size. With testing the *software*, we will know which setting will result the highest accuration. The maximal accuration of the identification is only 50% because of limited training data.
Key words : HMM, *codebook*, korona

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Energi listrik menjadi kebutuhan yang sangat penting bagi kehidupan manusia. Distribusi listrik dimulai dari pusat pembangkit tenaga listrik dimana level tegangannya mencapai ratusan kilovolt. Untuk bisa dikonsumsi oleh kita level tegangan ini melalui beberapa penyesuaian di gardu listrik TM (Tegangan Menengah) hingga mencapai 220 volt. Idealnya daya listrik selama pendistribusian itu harus konstan, namun sistem transmisi distribusi listrik yang menggunakan kawat rentan terhadap rugi – rugi karena memiliki tahanan yang cukup besar. Hal ini dapat diminimalisir dengan menaikan tegangan, akan tetapi permasalahan tidak akan berhenti sampai di situ, kenaikan tegangan tersebut salah satunya akan menyebabkan timbulnya gejala korona di gardu listrik, gejala ini bersifat akumulatif artinya akan terus bertambah seiring bertambahnya waktu. Gejala ini ditandai dengan suara dengungan dan percikan bunga api listrik (yang jelas terlihat pada malam hari) akan terlihat di sekitar gardu. Gejala ini akan mengakibatkan kerugian energi dan gangguan RI (*Radio Interference*) yang sifatnya merugikan bahkan pada level tertentu akan meledak sehingga mengganggu distribusi lisrik. Korona tidak bisa dihilangkan tetapi dapat diperlambat lajunya [1].

Oleh karena itu dalam skripsi ini akan dirancang sebuah sistem yang dapat mengenali korona. Untuk dapat mengenali korona digunakan teori kecerdasan tiruan karena data spektrum dari setiap level tidak selalu tepat sama. Dengan menggunakan teknik pembelajaran diharapkan hasil pengenalan dapat maksimum. Metoda pengenalan yang digunakan adalah Metoda *Hidden Markov Model* (HMM).

1.2 Tujuan Penulisan

Tujuan dari skripsi ini adalah untuk merancang sebuah sistem berupa *software* pendeteksi korona dengan MATLAB.

1.3 Batasan Masalah

Pembatasan masalah pada skripsi ini adalah :

1. Input korona berupa data suara digital sebagai basis data telah disediakan sehingga proses pengambilannya tidak akan dibahas lebih detail.
2. Jumlah data korona sangat terbatas, yaitu sebanyak 5 buah sampel.

1.4 Metodologi Penulisan

Dalam penulisan skripsi ini, metode yang dilakukan meliputi tahap – tahap sebagai berikut:

1. Studi literatur mengenai korona dengan melakukan pengumpulan data, pencarian informasi melalui buku-buku dan internet.
2. Studi literatur mengenai pemrosesan data audio digital
3. Merancang algoritma untuk beberapa proses yang dibutuhkan, baik dengan menciptakan algoritma baru ataupun memodifikasi algoritma yang telah ada sebelumnya.
4. Membuat sistem simulasi dan pengujian menggunakan perangkat lunak komputasi numerik.
5. Menganalisis dan menyimpulkan hasil pengujian yang dilakukan.
6. Dokumentasi dan laporan.

1.5 Sistematika Penulisan

Sistematika penulisan skripsi ini adalah sebagai berikut :

BAB I Pendahuluan; membahas latar belakang, tujuan penelitian, batasan masalah, metode perancangan, dan sistematika penulisan.

BAB II Landasan Teori; membahas landasan - landasan teori tentang korona dan metoda *Hidden Markov Models* (HMM)

BAB III Perancangan Sistem; membahas blok diagram sistem, prinsip kerja sistem dan perancangan *software*.

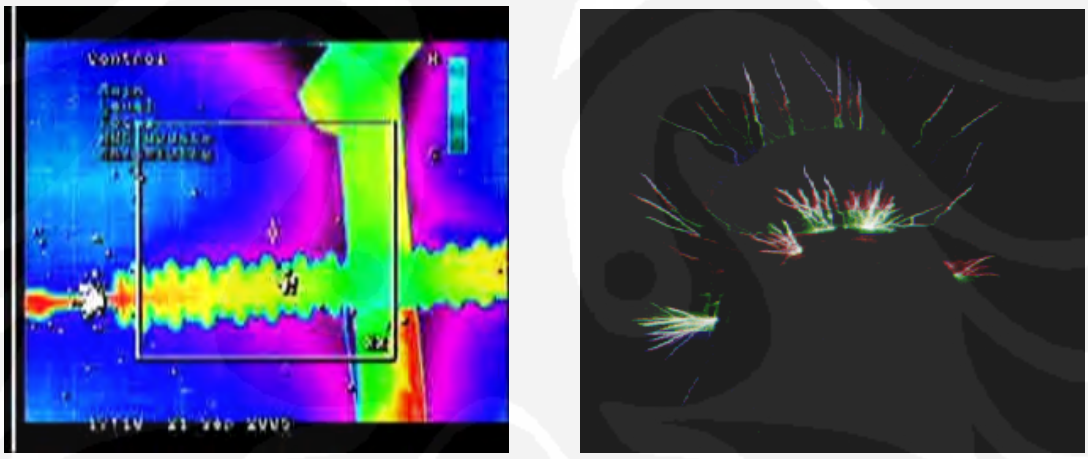
BAB IV Pengujian Sistem; membahas pengujian Program beserta analisisnya.

BAB V Penutup; membahasa kesimpulan dari penulisan skripsi.

BAB II
LANDASAN TEORI

2.1 Korona

Korona adalah ionisasi nitrogen di udara akibat intensitas medan listrik. Korona dapat dilihat dengan jelas dari percikan listrik yang terjadi saat malam hari, besarnya tegangan saat itu kira - kira 5/8 inci pada 3500 volt. Korona ditandai dengan suara berdesis, ozon, senyawa asam (kelembaban di udara) yang terakumulasi dalam bentuk bubuk putih atau bubuk hitam, cahaya pancarannya kuat saat ada sinar ultraviolet muncul dan berdekatan dengan sinar infra merah sehingga dapat dilihat secara langsung saat malam hari [2]. Fenomena yang muncul di sekitar gardu listrik ditunjukan oleh Gambar 2.1 (a) dan Gambar 2.1 (b).



(a.) (b.)

Gambar 2.1 Gambar lecutan korona
(a) Dilihat menggunakan sensor panas
(b) Dilihat dengan mata telanjang saat malam hari.

Akumulasi senyawa asam dan percikan listrik dapat menghasilkan karbon pada bahan penyekat. Korona juga berkontribusi terhadap kerusakan kimiawi yang terjadi pada lapisan pada penyekat. Kerusakan pada penyekat menyebabkan timbulnya medan listrik sehingga menimbulkan kebocoran, karbon dan kerusakan pada penyekat NCI [2]. Dalam simulasi, lingkaran korona dimunculkan di

sembarang tempat pada penyekat NCI saat tegangan mencapai 500 kV. Setelah dua tahun penyekat NCI diganti karena duapertiga bagiannya terbakar.

Besarnya tegangan bervariasi tergantung konfigurasi penyekat dan jenisnya. NCI biasanya berkisar pada tegangan 160 kV, pin dan cap berkisar pada tegangan 220 kV atau 345 kV tergantung toleransi pengaturannya.

Flash over adalah peristiwa dimana tegangan melewati tegangan *breakdown* tetapi tidak memiliki arus yang dapat menyebabkan kemungkinan terjadinya percikan. Percikan dapat disebabkan oleh kerusakan karena arus berlebih pada jaringan listrik sehingga tegangan turun dibawah 50 % atau hingga *protector* rusak / terbuka. *Flash – over* diakibatkan tegangan yang melewati perangkat yang rusak atau pemasangan yang kurang baik.

Korona terjadi karena adanya ionisasi dalam udara akibat dari perbedaan tegangan yang cukup tinggi antara dua elektroda, sehingga menyebabkan hilangnya elektron dari molekul udara [3]. Oleh karena lepasnya elektron dan ion, apabila di sekitarnya terdapat medan listrik, maka elektron-elektron bebas ini mengalami gaya yang mempercepat geraknya, sehingga terjadilah tabrakan dengan molekul lain. Akibatnya ialah timbulnya ion-ion dan elektron-elektron baru. Proses ini berjalan terus menerus dan jumlah elektron dan ion bebas menjadi berlipat ganda jika perbedaan tegangan antara dua elektroda semakin besar. Jika gradien tegangan di sekitar permukaan elektroda melampaui batas maksimum gradien tegangan yang mampu ditahan oleh udara, maka akan timbul korona.

2.1.1 Proses Pembentukan Korona [3]

Bila kedua kawat sejajar yang penampangnya kecil (dibandingkan dengan jarak antara kedua kawat tersebut) diberi tegangan bolak-balik, maka korona dapat terjadi. Dan pada tegangan yang cukup rendah tidak dapat terlihat kejadian tersebut. Bila tegangan dinaikkan, maka korona dapat terjadi secara bertahap. Awalnya kawat kelihatan bercahaya seperti sampul, mengeluarkan suara mendesis (*hissing effect*) dan berbau ozon. Warna cahayanya adalah ungu (violet) muda. Kegagalan pertama berawal dekat permukaan penghantar, yaitu tekanan elektrostatik atau gradien tegangannya maksimum dan ketebalan lapisan udara bertarnbah dengan penambahan tekanan. Dan jika tegangan terus dinaikkan, gejala tersebut akan semakin jelas terlihat, yakni cahaya bertambah terang

terutama pada bagian yang runcing, kasar, dan kotor. Namun, kenaikan tegangan tidak boleh melebihi batasan tertentu karena dapat menyebabkan terjadinya lompatan api. Bila tegangan masih juga dinaikkan, maka dapat terjadi busur api dan korona mengeluarkan panas, yang dapat dibuktikan dengan menggunakan wattmeter. Dalam keadaan udara lembab, korona menghasilkan asam nitrogen (*nitrous acid*) yang menyebabkan kawat menjadi berkarat.

Korona terjadi karena adanya ionisasi dalam udara, yaitu adanya kehilangan elektron dari molekul udara. Oleh karena lepasnya elektron dan ion, maka apabila di sekitarnya terdapat medan listrik, maka elektron-elektron bebas ini mengalami gaya yang mempercepat geraknya, sehingga terjadilah tabrakan dengan molekul lain. Akibatnya ialah timbul ion-ion dan elektro-elektron baru. Proses ini berjalan terus-menerus dan jumlah elektron dan ion bebas menjadi berlipat ganda bila gradien tegangan cukup besar. Pelepasan korona terjadi karena

1.) Reaksi kimia
2.) Asam nitrit
3.) Ozon
4.) Sinar ultraviolet
5.) Suara

Kondisi yang mempengaruhi timbulnya korona :

1.) Pergerakan udara
2.) Suhu udara
3.) Kelembaman udara

Proses ionisasi (pelipatgandaan elektron) akan berhenti jika medan listrik menurun. Tumbukan elektron selain menyebabkan terjadinya ionisasi molekul juga menyebabkan terjadinya eksitasi elektron atom gas, yakni berubahnya kedudukan elektron dari orbitnya semula ke tingkat orbit yang Iebih tinggi. Ketika elektron berpindah kembali ke tingkat orbit yang lebih dalam terjadi pelepasan energi berupa cahaya radiasi dan gelombang elektromagnetik, yakni berupa suara bising (*noise*).

2.1.1.1 Indikasi Korona

Korona ditandai dengan suara berdesis, ozon, senyawa asam (kelembaban di udara) yang terakumulasi dalam bentuk bubuk. Pancarannya kuat saat ada sinar ultraviolet namun lemah terhadap sinar tak nampak dan berdekatan dengan sinar infra merah sehingga dapat dilihat dengan mata secara langsung di malam hari. Berikut ini adalah teknologi yang mampu mendeteksi korona :

1.) Sinar matahari bi-spectral
2.) Penglihatan malam
3.) Infrared
4.) Gelombang suara manusia (100 - 20 Khz)
5.) Gelombang ultrasonik (40, 10-80 kHz)
6.) Gelombang akustik (150 Khz)

2.1.1.2 Menghilangkan Korona [3]

Untuk mengurangi korona pada peralatan listrik (gardu listrik) dapat dilakukan dengan menumpulkan sudut objek bertegangan tinggi sehingga dekat terhadap objek. Tempatkan sudut objek yang tajam di suatu tempat dengan tegangan *breakdown* yang lebih tinggi di udara. Caranya adalah dengan menempatkan bahan pengganti agar bersinggungan dengan konduktor sehingga tegangan *breakdown* akan lebih tinggi daripada keadaan sekitar. Melindungi sudut objek yang tajam dengan film penyekat dapat meningkatkan korona dengan nilai medan listrik (E) yang tinggi. Dikenal dengan *corona dope*, benda ini adalah semacam lapisan cat atau gel, dan *Glyptal* atau paku halus. Semprotan acrylic juga bisa digunakan sebagai alternatif walaupun hasil *coating* lebih tipis. Simpan adonan tersebut dalam penyekat (yang terbuat dari bahan sulfur atau parafin, silikon rtv) atau juga dicampur dengan minyak dan larutan penyekat lainnya. Cara lain yang cukup populer untuk mengurangi kadar korona adalah menutup sekeliling konduktor dengan bahan film semikonduktor atau lapisan dengan diameter yang lebih besar sehingga menurunkan kekuatan medan. Tidak dibutuhkan tembaga berukuran besar untuk mengalirkan arusnya (mikro/mili), tetapi cukup tembaga dengan diameter sedikit melebihi konduktor. Gunakanlah tembaga dari Produsen yang terkenal seperti Belden, Rowe – Talley dan Caton karena memilki kualitas yang baik.

Cincin medan juga sering digunakan pada peralatan bertegangan tinggi untuk mengontrol distribusi medan listrik. Jangan membiarkan medan yang akan muncul pada daerah bebas/terbuka diantara dua konduktor atau rangkaian konduktor sehingga memunculkan tegangan bernilai menengah. Tegangan ini berasal dari pembagi kapasitif atau resistif, pembagi kapasitif sangat simpel yang dapat berperan sebagai kapasitansi dari cincin tersebut.

2.1.2 Jenis - Jenis Korona

2.1.2.1 Menurut Bentuk [4]

1.) Cahaya Tampak

Salah satu bentuk tahapan yang terjadi pada proses korona adalah tampaknya cahaya pada disekitar permukaan penghantar. Cahaya yang berwarna ungu muda ini berasal dari pengaruh tekanan yang berlebihan dari medan listrik. Cahaya ini hanya dapat dilihat pada kondisi yang gelap. Seperti yang telah disebutkan di atas bahwa cahaya ini berasal dari proses rekombinasi antara ion nitrogen dengan elektron bebas.

2.) Interferensi Frekuensi

Pada proses korona terjadi emisi energi yang kemudian meradiasi benda yang ada di sekitarnya. Salah satu radiasinya adalah munculnya sinyal *noise* pada jalur komunikasi, penerima radio dan penerima TV. Sinyal *noise* ini disebut sebagai interferensi radio.

Interferensi radio diawali dengan adanya benturan-benturan yang diakibatkan oleh pergerakan elektron. Adanya pergerakan elektron akan menimbulkan aliran arus yang cukup lemah. Aliran arus tersebut akan menghasilkan medan magnet dan medan elektrostatis di sekitar pergerakannya. Akibat keduanya dibentuk secara tiba-tiba dan dengan waktu yang singkat, medan magnet dan elektrostatis ini memiliki frekuensi yang tinggi. Hal ini menyebabkan medan tersebut dapat menginduksi pulsa tegangan di dekat antena radio dan kemudian menghasilkan interferensi radio. Inilah mekanisme terjadinya interferensi radio akibat benturan elektron.

3.) Suara Bising

Kemudian bentuk dari proses korona lainnya adalah timbulnya bunyi-bunyi di sekitar penghantar. Bunyi-bunyi ini dapat didengar oleh kuping manusia dan juga bergantung dari besar frekuensi yang dibangkitkannya. Bunyi-bunyi yang dibangkitkan oleh kawat konduktor ini biasa disebut *acoustical noise* atau gangguan bising. Gangguan bising merupakan bentuk korona yang mengganggu orang yang berada di sekitar konduktor tersebut dengan gangguan berisik. Gangguan bising yang dihasilkan korona dapat diukur dengan satuan dB. Besar dB yang dihasilkan di sepanjang kawat konduktor dipengaruhi oleh konduktor yang digunakan dan juga kondisi cuaca disekitar konduktor.

2.1.3 Efek yang Ditimbulkan Korona [4]

Akumulasi senyawa asam dan percikan listrik dapat menghasilkan karbon pada bahan penyekat. Korona juga berkontribusi terhadap kerusakan kimiawi yang terjadi pada lapisan pada penyekat. Kerusakan pada penyekat menyebabkan timbulnya medan listrik sehingga menimbulkan kebocoran, karbon dan kerusakan pada penyekat NCI. Dalam simulasi, lingkaran korona dimunculkan di sembarang tempat pada penyekat NCI saat tegangan mencapai 500 kV. Setelah dua tahun penyekat NCI diganti karena duapertiga terbakar.

1.) Gangguan Bising

Bunyi-bunyi yang dibangkitkan korona biasa disebut *acoustical noise* atau gangguan bising. Gangguan bising merupakan bentuk korona yang mengganggu orang yang berada di sekitar lokasi dengan gangguan berisik. Gangguan bising yang dihasilkan korona dapat diukur dengan satuan dB.

Gangguan bising dipengaruhi oleh beberapa faktor, antara lain: besarnya gradien tegangan, kondisi cuaca dan jarak sumber bunyi. Diantara ketiga faktor tersebut, besar gradien tegangan merupakan faktor yang paling berpengaruh. Sedikit perubahan gradien tegangan dapat mengakibatkan adanya penambahan level daya dari gangguan bising. Pengaruh akibat gradien tegangan merupakan faktor yang paling penting dalam menghitung besarnya level daya dari gangguan bising.

2.) Interferensi Frekuensi

Pada proses korona terjadi emisi energi yang kemudian meradiasi benda yang ada di sekitarnya. Salah satu radiasinya adalah munculnya sinyal *noise* pada jalur komunikasi, penerima radio dan penerima TV. Sinyal *noise* ini disebut sebagai interferensi radio.

Interferensi radio diawali dengan adanya benturan-benturan yang diakibatkan oleh pergerakan elektron. Adanya pergerakan elektron akan menimbulkan aliran arus yang cukup lemah. Aliran arus tersebut akan menghasilkan medan magnet dan medan elektrostatis di sekitar pergerakannya. Akibat keduanya dibentuk secara tiba-tiba dan dengan waktu yang singkat, medan magnet dan elektrostatis ini memiliki frekuensi yang tinggi. Hal ini menyebabkan medan tersebut dapat menginduksi pulsa tegangan di dekat antena radio dan kemudian menghasilkan interferensi radio. Inilah mekanisme terjadinya interferensi radio akibat benturan elektron.

3.) Gangguan Pada Performa Peralatan Elektronik

Pada bahan isolasi, korona menyebabkan terjadinya transfer elektron dan lompatan-lompatan listrik di tempat yang seharusnya tidak boleh terjadi hal-hal tersebut. Dengan adanya kejadian ini akan menyebabkan adanya energi yang hilang. Pada saat yang sama, akibat adanya korona ini akan menyebabkan timbulnya panas di sekitar daerah terjadinya korona. Sedangkan pada bagian lain bahan isolasi, korona akan menghasilkan arus transien yang dapat mengalir ke peralatan yang terhubung dengan bahan isolasi tersebut. Menurut statistik IEEE, kegagalan isolasi merupakan penyebab utama terjadinya kerusakan pada sistem dan peralatan kelistrikan.

Secara garis besar korona memiliki dua efek yang sangat penting secara ekonomis karena dapat menyebabkan naiknya biaya operasional dan perawatan dari peralatan listrik. Efek yang pertama adalah korona dapat mengurangi usia pakai dari bahan isolasi yang digunakan. Sedangkan efek yang kedua adalah kemungkinan terjadinya arus transien yang dapat mengganggu aktivitas kerja dari peralatan komunikasi, kontrol dan alat ukur.

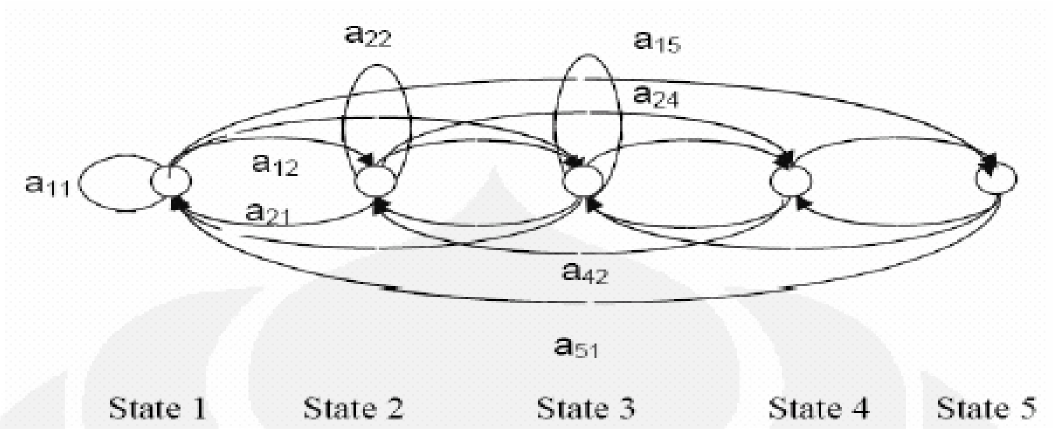
2.2 Hidden Markov Model (HMM)

Hidden Markov Model (HMM) pertama kali dikenali dan dipelajari pertama kali di awal tahun 1970, metode pemodelan statisik ini semakin populer beberapa tahun belakangan ini karena dua hal, pertama adalah pemodelannya yang sangat matematis oleh karena itu aplikasinya sangat luas dan mampu memodelkan dengan tepat. Proses dunia nyata menghasilkan output observasi yang dapat dikarakteristikan sebagai sinyal. Sinyal tersbut dapat berupa discrete di alam (contoh : karakter alphabet, kuantisasi vektor dari codebook, dll) atau kontinyu (contoh : sampel suara, pengukuran temperatur, musik dll). Sumber sinyal bersifat tetap ataupun berubah – ubah (berubah terhadap waktu). Sinyal dapat dikatakan murni (keluar langsung dari sumber sinyal) atau merupakan hasil perubahan dari sumber sinyal aslinya (*noise*) atau distorsi transmisi, gema, gaung dan lainnya. Permasalahan pokok yang menarik adalah memodelkan sinyal dalam dunia nyata ke dalam bentuk sinyal model. Ada beberapa alasan mengapa ada ketertarikan dalam mengaplikasikan sinyal dalam model. Yang pertama, model sinyal dapat menghasilkan deskripsi teori dasar mengenai proses sistem. Sebagai contoh kita ingin meningkatkan frekusensi suara *noise* untuk mendesain sistem yang secara optimal membuang *noise* dan mengulangi proses distorsi. Alasan yang kedua adalah pemodelan sinyal membuat kita dapat mempelajari mengenai sumber sinyal pada dunia nyata tanpa harus memiliki sumber sinyalnya. Hal ini sangat penting ketika harga untuk mendapatkan sinyal bernilai tinggi.[5]

2.2.1 Parameter HMM

2.2.1.1 Parameter A

Parameter A disebut sebagai probabilitas transisi, merupakan probabilitas kedudukan suatu *state* terhadap semua *state* yang ada, termasuk kedudukan terhadap *state* itu sendiri. Contoh dari matriks transisi dapat dilihat pada Gambar 2.2.



Gambar 2.2 Matrik transisi

Parameter A pada HMM dinyatakan dalam sebuah matriks dengan ukuran M x M dengan M adalah jumlah *state* yang ada. Matriks transisi pada Gambar 2.2 terdiri dari 5 *state* sehingga setiap *state* memiliki 5 hubungan transisi, maka parameter A dapat dituliskan dalam bentuk matriks seperti pada Persamaan 2.1

$$A = aij = \begin{pmatrix} a11 & a12 & a13 & a14 & a15 \\ a21 & a22 & a23 & a24 & a25 \\ a31 & a32 & a33 & a34 & a35 \\ a41 & a42 & a43 & a44 & a45 \\ a51 & a52 & a53 & a54 & a55 \end{pmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.1)$$

2.2.1.2 Parameter B

Parameter B disebut sebagai probabilitas *state*, merupakan probabilitas kemunculan suatu *state* dalam deretan seluruh *state* yang ada. Parameter B dalam HMM dituliskan dalam bentuk matriks kolom dengan ukuran M x 1 dimana M merupakan jumlah seluruh *state* yang ada. Sebagai contoh, jika terdapat 5 buah *state* dalam suatu kondisi, maka matriks B yang terbentuk ditunjukkan oleh Persamaan 2.2.

$$B = \begin{pmatrix} b1 \\ b2 \\ b3 \\ b4 \\ b5 \end{pmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.. \quad (2.2)$$

2.2.1.3 Parameter μ

Parameter μ disebut sebagai probabilitas awal, merupakan probabilitas kemunculan suatu *state* di awal. Sama halnya dengan parameter B, parameter μ juga dituliskan dalam bentuk matriks kolom dengan ukuran M x 1 dimana M adalah jumlah *state*nya. Jadi jika terdapat 5 *state*, maka parameter μ yang dihasilkan akan ditunjukkan seperti pada Persamaan 2.3

$$\Pi = \begin{pmatrix} c1 \\ c2 \\ c3 \\ c4 \\ c5 \end{pmatrix} \quad \text{……..…..} \quad (2.3)$$

Dari ketiga parameter utama maka HMM dapat dituliskan dalam bentuk λ = (A, B, μ). Dari kesemua parameter yang ada maka bisa diperoleh suatu probabilitas observasi (O). Fungsi untuk probabilitas O ditunjukkan oleh Persamaan 2.4.

$$P(O) = \sum_{i=1}^{N} P(A_{ij}) * P(B_i) \quad \text{……..(2.4)}$$

Berikut adalah contoh perhitungan untuk mencari probabilitas observasi:

Citra 1 → (w1, w1, w2, w1, w2) → P(O) citra1 = c1*a11* b1* a12 *b2* a21*b2* a12*b1

Citra 2 → (w2, w1, w1, w3, w2) → P(O) citra2 = c2* a21* b2* a11 *b1*a13*b1 *a32 * b3

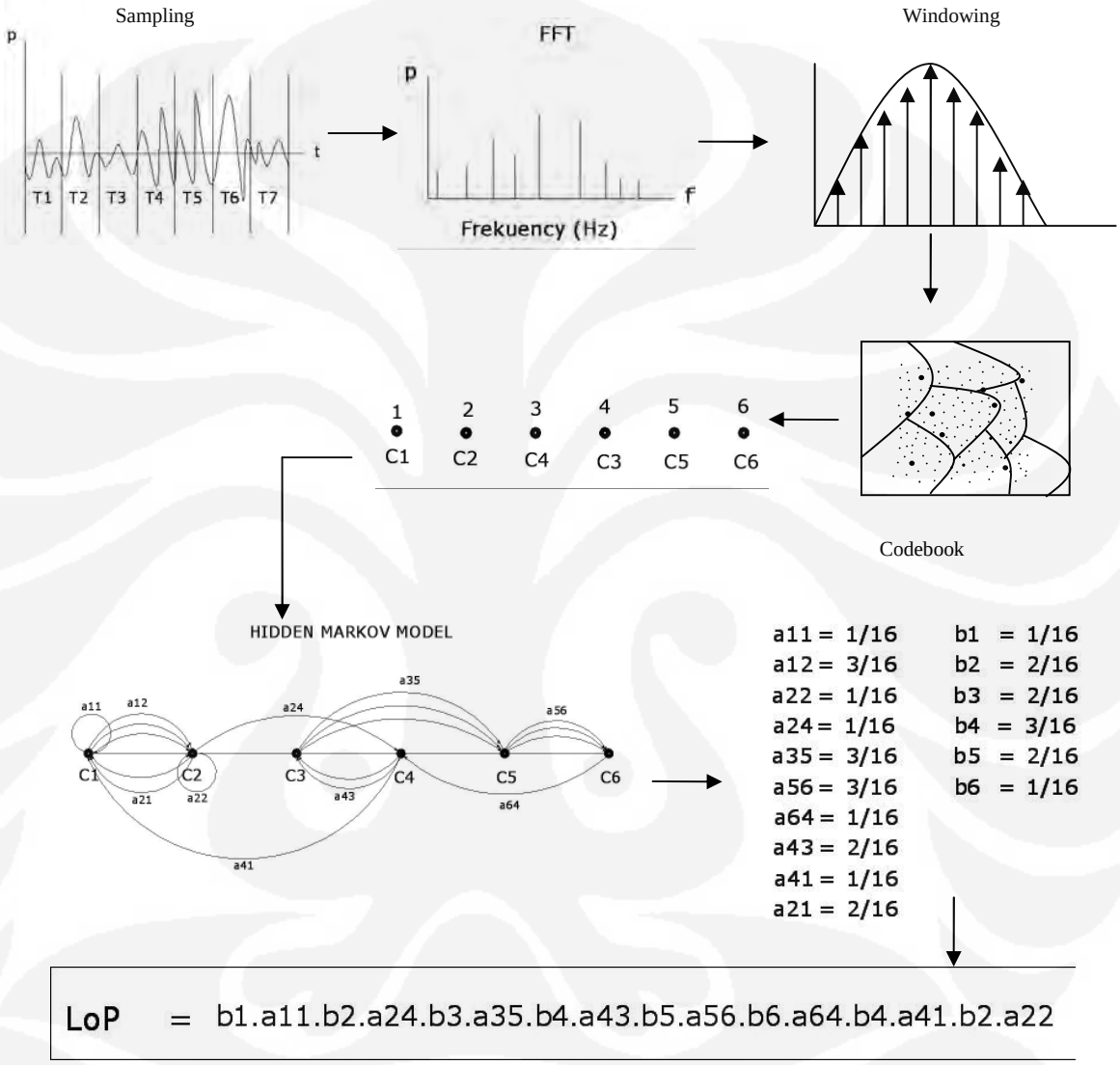
Proses terjadinya nilai probabilitas HMM adalah sebagai berikut:

1.) Data dibagi menjadi data-data kecil melalui proses *frame blocking* kemudian dicocokkan berdasarkan *codebook* yang dimiliki. Pada proses pencocokan dengan *codebook* akan dihitung jarak dari tiap data dengan *centroid centroid*nya. Jarak yang paling dekat akan menentukan urutan kode observasi.

2.) Data yang telah dikenali berdasarkan *codebook* akan dicocokkan dengan nilai pada parameter HMM. Parameter HMM sesuai dengan urutan kode observasi.

2.2.2 Proses Pembentukan Parameter HMM [1]

Proses pembentukan parameter HMM hingga didapat **LOP** (*Log Of Probability*) sebagai pembanding *database* dengan *identifikasi* dapat dilihat pada Gambar 2.3.

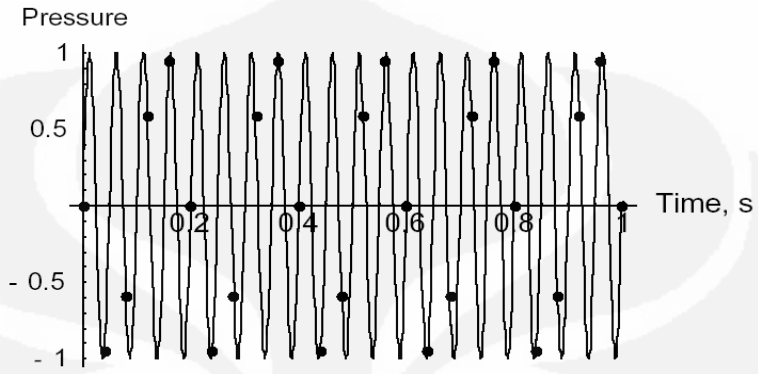


Gambar 2.3 Proses Pembentukan Parameter HMM

1.) Sampling

Proses *sampling* berbeda-beda untuk setiap suara. Bila *sampling* terhadap suatu sinyal suara tidak akurat maka dapat terjadi *misleading* atau hasil yang tidak sesuai dengan aslinya.

Aliasing merupakan hasil dari *sampling* secara diskrit pada suatu sinyal yang terlalu rendah sehingga memberikan resolusi yang rendah pula. Gambar 2.4 menunjukan sampel sinyal 10 Hz yang nampak menjadi sinyal 5 Hz pada Gambar 2.4. Ini menunjukan sebuah contoh terjadinya *aliasing*.



Gambar 2.4 Sampling diskrit suatu sinyal

2.) Fast Fourier Transform (FFT)

FFT mengubah masing-masing *frame* dari domain waktu ke domain frekuensi. FFT adalah *fast algorithm* untuk mengimplementasikan *Discrete Fourier Transform (DFT)* yang mana didefinisikan sebagai *N* sampel $\{x_n\}$, ditunjukan dengan Persamaan 2.5.

$$X_n \sum_{k\,0}^{N\ 1} x_k e^{\ 2\pi jkn/N} \quad \text{.....(2.5)}$$

Hasil *sequence* $\{x_n\}$ diinterpretasikan sebagai berikut :

1.) frekuensi nol untuk harga $n\ 0$

2.) frekuensi positif $0\ \ f\ \ f_s/2$ untuk harga $1\ \square n\ \square N/2\ \ 1$

3.) frekuensi negatif $f_s/2\ \ f\ \ 0$ untuk harga $N/2\ 1\ \square n\ \square N\ \ 1$

3.) Windowing

Langkah selanjutnya adalah *windowing* masing-masing *individual frame* untuk meminimalisasikan diskontinuitas sinyal pada permulaan dan akhir dari masing-masing *frame*. *Windowing* ini untuk meminimalisasikan *spectral distortion* dengan menggunakan *window* untuk men-*taper* sinyal ke nol pada permulaan dan akhir masing-masing *frame*. Jika kita mendefinisikan *window*

sebagai *w(n),* $0 \le n \le N-1$, dimana *N* adalah angka sampel pada masing-masing *frame*. Hasil *windowing* adalah signal yang dinyatakan dengan Persamaan 2.6.

$$y_1(n) = x_1(n)w(n), \qquad 0 \le n \le N-1 \quad \text{.....(2.6)}$$

Pada program ini menggunakan *Hamming windowing* yang dinyatakan dalam Persamaan 2.7.

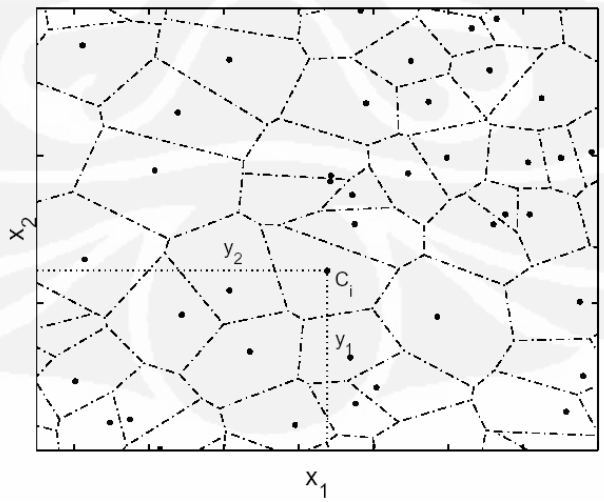
$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1 \quad \text{.....(2.7)}$$

4. Vector Quantization

VQ adalah proses dari pemetaan vektor dari ruang vektor yang besar menjadi sebuah wilayah yang terbatas. Masing-masing wilayah ini disebut *cluster* dan dapat direpresentasikan dengan *centroid* yang disebut *codeword*. Koleksi dari semua *codeword* disebut *codebook* yang berhubungan untuk suara yang telah diketahui.

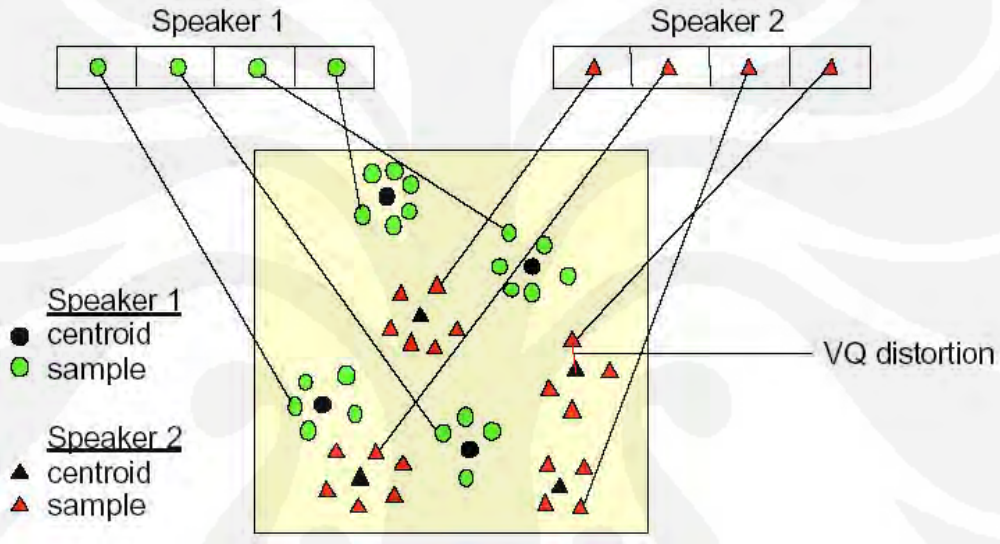
VQ diinterpretasikan dengan skalar kuantisasi. Sinyal input akan dikuantisasi menjadi *codebook* $C = \{y_k \mid k = 1,...,N\}$. Hampir keseluruhan sinyal input merupakan sebuah vektor yang harus dikodekan kedalam ruang multidimensi. Gambar 2.5 merupakan contoh ruang dua dimensi dari *codebook*. Gambar 2.5 menunjukan partisi dari ruang multidimensi sebuah input vektor yang dibagi menjadi *L* wilayah yang dapat dinotasikan sebagai $P = \{C_1, C_2, ..., C_L\}$ dimana

$$C_i = \{x \mid d(x, y_i) \le d(x, y_j), j \ne i\} \quad \text{.....(2.8)}$$



Gambar 2.5 Codebook dari suatu input vektor

Gambar 2.6 menunjukan konseptual diagram untuk mengilustrasikan proses *recognition*. Pada Gambar 2.6 hanya diGambarkan 2 suara dari 2 *speaker* dalam ruang akustik dua dimensi. Lingkaran menunjukan vektor akustik dari suara 1, sedangkan segitiga adalah vektor akustik dari suara 2. Dalam tahap *training, VQ codebook* untuk masing-masing suara yang telah diketahui dibuat dengan mengumpulkan vektor akustik *training*-nya menjadi sebuah *cluster*. Hasil *codeword*-nya ditunjukan pada Gambar 2.6 dengan lingkaran dan segitiga hitam untuk suara 1 dan 2. Jarak dari sebuah vektor ke *codeword* terdekat disebut *distortion*.



Gambar 2.6 Diagram konsep pembentukan codebook dengan vector quantization.

Pada tahap *recognition*, sebuah input dari suara yang tidak dikenal akan dilakukan proses *vector-quantized* dengan menggunakan semua *trained codebook* dan selanjutnya dihitung total *VQ distortion*-nya. Total distortion yang paling kecil antara *codeword* dari salah satu suara dalam *database* dan *VQ codebook* dari suara input diambil sebagai hasil identifikasi.

Dalam pembentukan *codebook* untuk iterasi guna memperbaiki VQ digunakan *General Lloyd Algorithm (GLA)* atau yang sering disebut dengan *LBG Algoritm. LBG VQ algorithm* tersebut dapat diimplementasikan dengan prosedur rekursif sebagai berikut :

1.) Mendesign suatu vektor *codebook* yang merupakan *centroid* dari keseluruhan vektor *training*.

2.) Menjadikan ukuran *codebook* dua kali lipat dengan membagi masing-masing *current codebook* C_n menurut aturan

$C_n^{\square} = C_n(1+\varepsilon)$ …….. (2.9)

$C_n^{\square} = C_n(1-\varepsilon)$ ……..…….(2.10)

dimana *n* bervariasi dari 1 sampai dengan *current size codebook* dan ε adalah parameter *splitting* $(\varepsilon = 0.01)$.

3.) *Nearest Neighbour Search*, yaitu mengelompokan *training vector* yang mengumpul pada blok tertentu. Selanjutnya menentukan *codeword* dalam *current codebook* yang terdekat dan memberikan tanda vektor yaitu *cell* yang diasosiasikan dengan *codeword* yang terdekat.

4.) *Centroid update,* yaitu menentukan *centroid* baru yang merupakan *codeword* yang baru pada masing-masing *cell* dengan menggunakan *training vector* pada *cell* tersebut.

5.) Iterasi 1

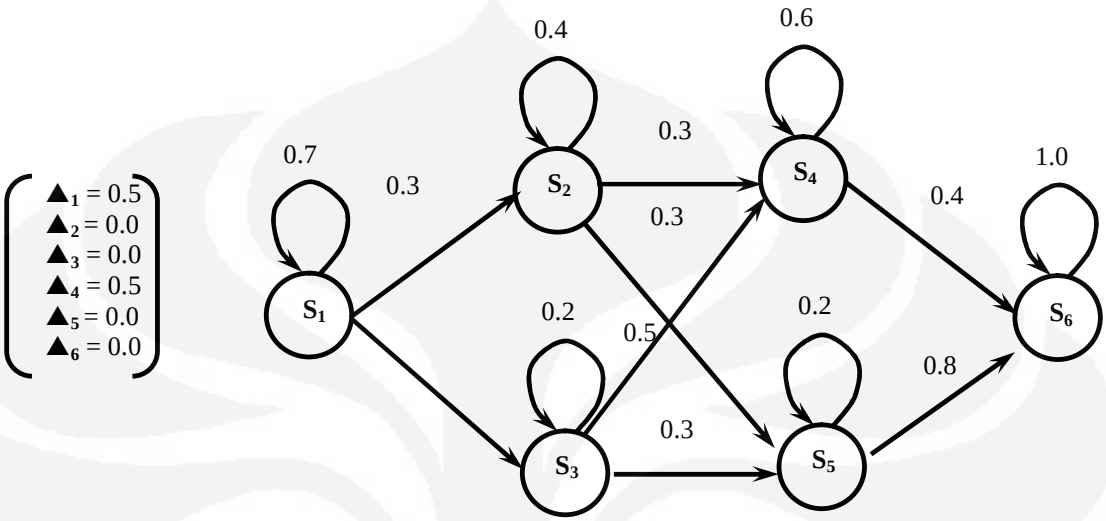
mengulang step 3 dan 4 sampai jarak rata-rata dibawah *present threshold.*

6.) Iterasi 2

mengulang step 2, 3, 4 sampai *codebook* berukuran *M*.

2.2.3 Topologi HMM

HMM memiliki beberapa macam topologi, salah satunya adalah model kiri - kanan 6 kondisi yang ditunjukan oleh Gambar 2.7.



Gambar 2.7 Model kiri – kanan 6 kondisi

S menunjukan *state* atau variabel B dimana nilai probabilitasnya ditunjukan dengan simbol sigma (▲), sedangkan nilai transisi antar *state* adalah variabel A dan **S1** adalah nilai awal *state* atau variabel μ.

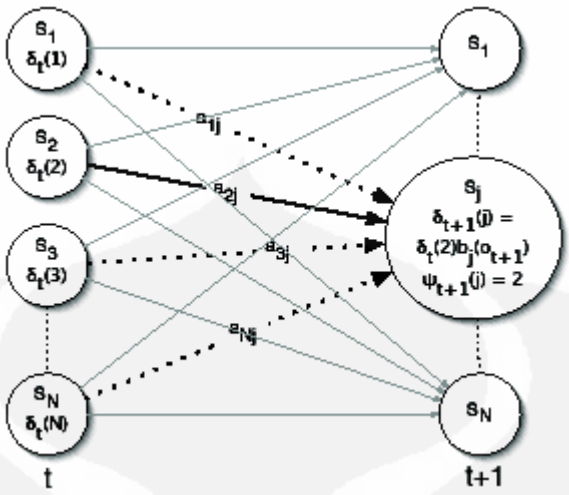
2.2.4 Dekoding

Tujuan dari dekoding adalah untuk menemukan urutan *state Hidden* yang paling menyerupai observasi yang diberikan. Solusinya adalah dengan dengan menggunakan *viterbi algorithm* yang merupakan turunan dari *forward algorithm* tetapi probabiliti perpindahannya dioptimalkan untuk setiap langkahnya. Pertama kita mendefinisikan persamaannya yang ditunjukan oleh Persamaan 2.11:

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P(q_1 q_2 \cdots q_t = s_t, o_1, o_2 \cdots o_t | \lambda) \quad \text{.....(2.11)}$$

Sedangkan langkah pertama *viterbi algorithm* dengan menggunakan Persamaan 2.11 yaitu inisialisasi, sedangkan langkah selanjutnya adalah rekursi dengan menggunakan Persamaan 2.12, proses rekursi dapat dilihat pada Gambar 2.8.

$$q_T^* = \arg \max_{1 \le i \le N} [\delta_T(i)]. \quad \text{.....(2.12)}$$

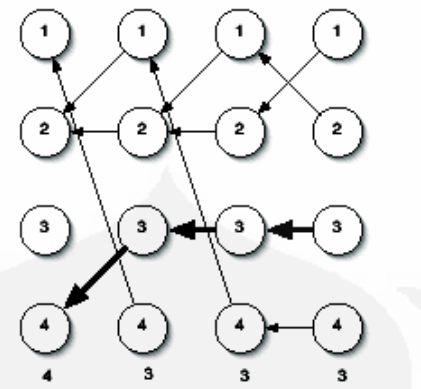


Gambar 2.8 Proses rekursi

Perbedaan dengan *forward algorithm* berada pada proses ini dimana kita memaksimalkan *state* daripada menjumlahkannya kemudian disimpan sebagai *pointer*. Langkah berikutnya adalah *termination* dengan menggunakan Persamaan 2.13. Dan langkah terakhir adalah *backtracking*, proses ini menghasilkan *state* yang paling menyerupai observasi yang terlewatkan pada proses rekursi, tetapi tidak mudah untuk menemukannya. Proses *backtracking* dapat dilakukan dengan menggunakan Persamaan 2.13.

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \cdots, 1. \quad \text{.....(2.13)}$$

Gambar 2.9 adalah proses *backtracking*, dimana jalur *state* yang terbaik adalah yang diberi tanda panah tebal, dimana jalur yang digunakan lebih pendek dan *state transition* sangat minimal sehingga mempercepat proses pengenalan dan menghasilkan obervasi yang lebih baik.



Gambar 2.9 Proses *backtracking*

2.2.5 *Training* [1]

Tujuan utama dari metoda *Hidden Markov Model* adalah memperkirakan parameter modelnya, $\lambda = (A, B, \pi)$. Ada dua langkah untuk mendapatkannya tergantung dari bentuk sampel, yang akan dijadikan referensi sebagai *training* yang terawasi dan tidak terawasi. Jika sampel *training* berisi input dan output dari sebuah proses, kita dapat membuat sebuah *training* yang terawasi dengan menyamakan input sebagai observasi dan ouput sebagai *state*. Tetapi jika hanya input yang ada pada data *training* maka kita hanya mendapatkan *training* yang tidak terawasi. Jnis *training* yang akan dijelaskan adalah *supervised training*.

Solusi termudah untuk membuat model λ, adalah dengan memperbanyak sampel data *training*, dimana setiap sampel diberikan klasifikasi yang tepat. Contoh pengklasifikasian yang sering digunakan adalah *PoS tagging*, langkah – langkahnya adalah sebagai berikut :

1.) $t_1.....t_N$ adalah *tags* atau label, di dalam HMM diinisialisasikan sebagai *state* $s_1....s_N$

2.) $w_1.....w_N$ adalah *words*, di dalam HMM diinisialisasikan sebagai observasi $v_1...v_N$

Dengan dua model di atas kita dapat membuat Gambaran mengenai output atau *state* yang paling menyerupai dengan *words* atau observasi. Selanjutnya untuk menentukan model λ, kita dapat menggunakan *likelihood estimates* (MLE) dari data sampel yang berisi *tags* beserta *PoS*-nya. Untuk menentukan matriks transisi dapat diperoleh dengan Persamaan 2.14

$$a_{ij} = P(t_i|t_j) = \frac{Count(t_i, t_j)}{Count(t_i)}$$ .....(2.14)

Dimana *count* (t_i,t_j) adalah adalah waktu dari matrik *J* dibagi waktu dari matriks *I* pada data *training*. Sedangkan untuk memperoleh matriks observasi dengan Persamaan 2.15

$$b_j(k) = P(w_k|t_j) = \frac{Count(w_k, t_j)}{Count(t_j)}$$ .....(2.15)

Dimana *count*(w*k*, t*j*) adalah waktu *tags* matriks *J* terhadap *words* matriks *K* pada data training. Dan parameter terakhir adalah distribusi probabilitas dapat diperoleh dengan Persamaan 2.16.

$$\pi_i = P(q_1 = t_i) = \frac{Count(q_1 = t_i)}{Count(q_1)}$$ .....(2.16)

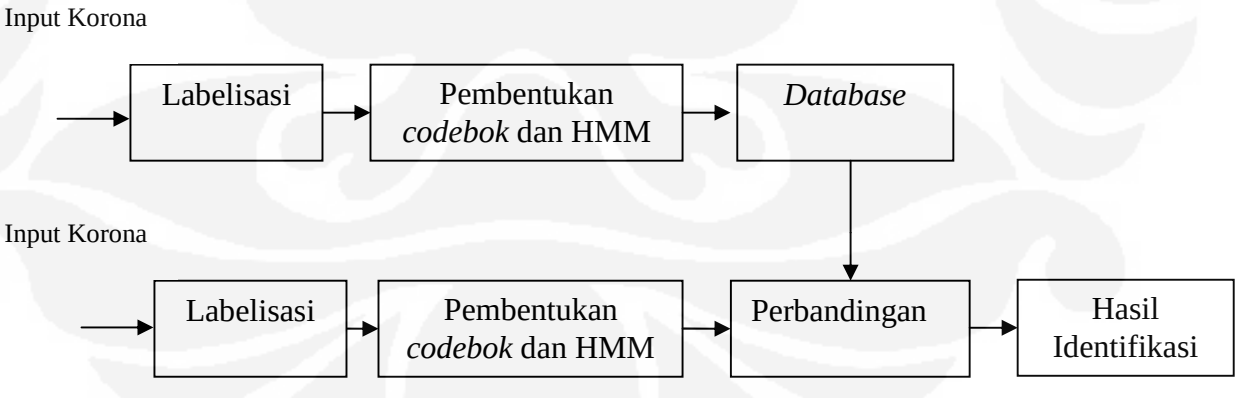
Dalam prakteknya, untuk menentukan parameter HMM dari *counts* untuk mendapatkan hasil yang baik dengan menghindari perhitungan yang menghasilkan 0 sehingga model tidak akan muncul pada proses *training*.

BAB III
PERANCANGAN

Perancangan sistem pendeteksi korona ini menggunakan metoda Hidden Markov Model (HMM). Metoda ini dipilih karena kelebihannya yang mampu memodelkan berbagai aplikasi ke dalam persamaan matematis. Input sistem berupa file audio (*.wav). Pengolahan dilakukan di dalam *notebook* dengan spesifikasi : Processor Intel Pentium M 1.8 Ghz dan Memori sebesar 768 MHz menggunakan MATLAB Versi R2008a.

3.1 Prinsip kerja

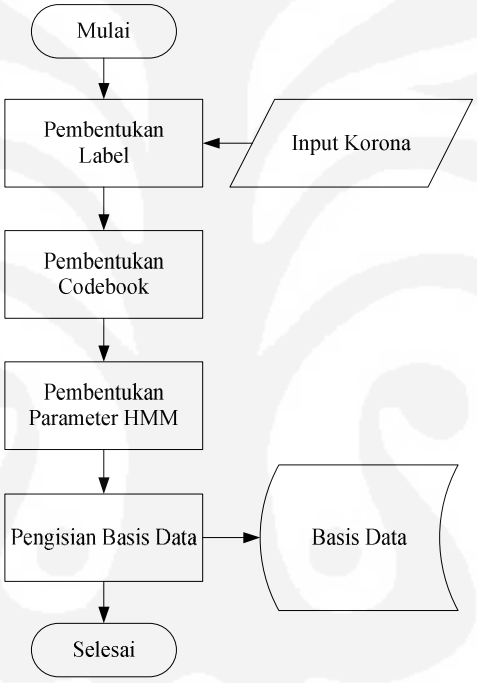
Prinsip kerja dari sistem ini adalah mendeteksi korona berupa data digital audio (*.wav) kemudian menentukan levelnya apakah termasuk ke dalam kriteria rendah, sedang ataupun tinggi sebagai hasil identifikasinya. Untuk mengetahui level korona yang akan diidentifikasi, sebelumnya telah dilakukan proses *training* yaitu pembentukan *database* yang berisi data berupa parameter HMM (Hidden Markov Model) untuk ketiga kriteria tersebut sebagai proses identifikasi. Diagram blok perancangan *software* ditunjukan oleh Gambar 3.1



Gambar 3.1 Diagram Blok Perancangan *Software*

3.2 Pembentukan *Database*

Pembentukan *database* meliputi 3 proses, yaitu : Labelisasi, pembentukan *codebook* dan pembentukan parameter HMM. Dalam MATLAB, file *database* berbentuk (*.mat). Labe11.mat merupakan *database* untuk level rendah, label2.mat untuk level sedang, dan label3.mat untuk level tinggi. Sedangkan *code*book_korona.mat merupakan *database* yang berisi *codebook* serta hmm_korona.mat yang berisi parameter HMM-nya. Diagram alir dari pembentukan *database* ditunjukan oleh Gambar 3.2.



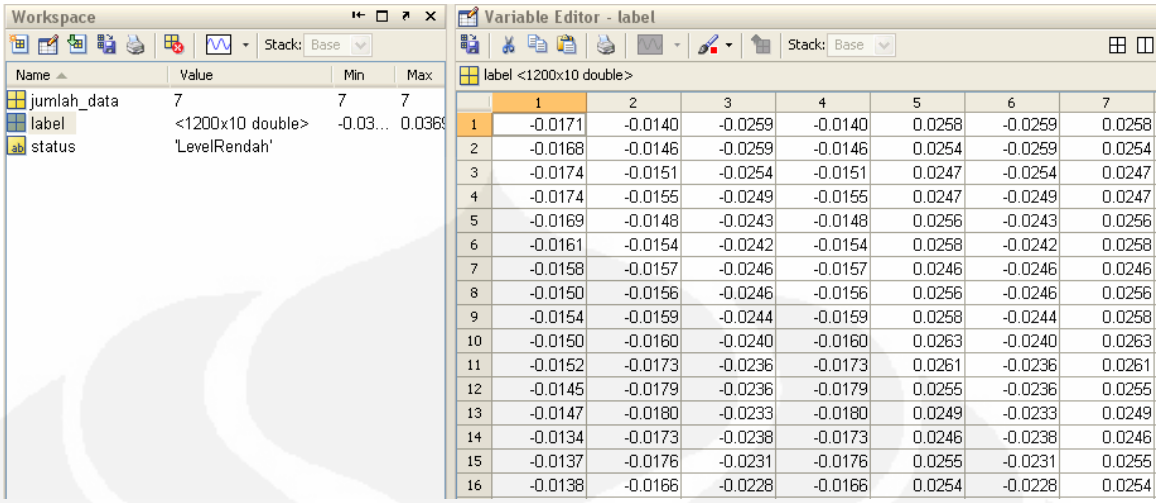
Gambar 3.2 Diagram Alir Pembentukan *Database*

3.2.1 Labelisasi

Proses labelisasi adalah proses pemberian tanda/label pada data latih korona yang akan dimasukan ke dalam *database*, contoh penamaan : Input korona yang diketahui memiliki level rendah kadar koronanya diberi label rendah_1, rendah_2, hingga rendah_n. begitupun dengan input korona berlevel sedang dan tinggi. *Database* labelisasi ini memiliki 3 buah parameter yaitu : Jumlah data yang berisi informasi mengenai jumlah data latih, hasil perhitungan input, dan status yang berisi level input yang bersangkutan. Berikut ini *listing* program labelisasi :

```
function Cari_Input_Callback(hObject, eventdata,
handles)
[nama_file, nama_path]=uigetfile({'*.wav','WAV
File(*.wav)'},'Open Sound File');
if  ~isequal(nama_file, 0)
    wav_file_name = nama_file;
 guidata(hObject, handles);
 set(handles.edit_input,'string',nama_file);
else
 return;
end
fs=12000;
eval (['load ' labelx ' label jumlah_data']);
jumlah_akhir=jumlah_data+1;
if jumlah_akhir==1
    [label]=zeros(fs*sampling,ite);
    [speech2]=wavread(wav_file_name,fs*sampling);
    [a,b]=size(speech2);
    if b>1
        [speech] = speech2(:,1);
    else
        speech(:,1)=speech2;
    end
        label(:,1)=speech;
else
    [speech2]=wavread(wav_file_name,fs*sampling);
    [a,b]=size(speech2);
    if b>1
        [speech] = speech2(:,1);
    else
        speech(:,1)=speech2;
    end
    label(:,jumlah_akhir)=speech;
end
 jumlah_data=jumlah_akhir;
 eval(['save ' labelx ' status label jumlah_data']);
       msgbox ('label has been created successfully
');
```

Pembentukan *database* labelisasi untuk level rendah, sedang dan tinggi pada MATLAB ditunjukan oleh Gambar 3.3, Gambar 3.4 dan Gambar 3.5.



Workspace

Stack: Base

Name	Value	Min	Max
jumlah_data	7	7	7
label	<1200x10 double>	-0.03...	0.036
status	'LevelRendah'		

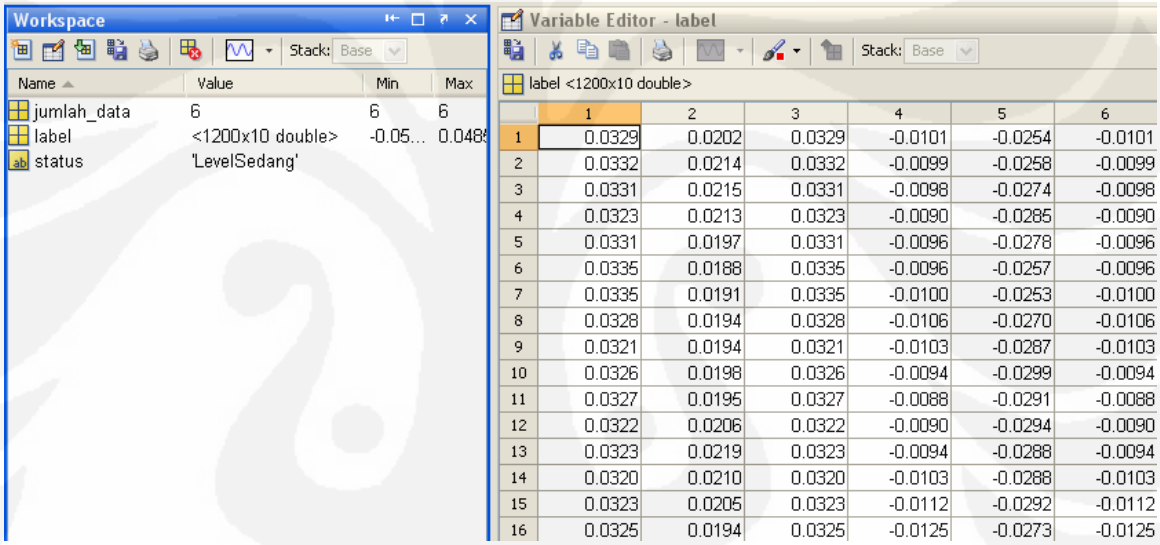
Variable Editor - label

Stack: Base

label <1200x10 double>

	1	2	3	4	5	6	7
1	-0.0171	-0.0140	-0.0259	-0.0140	0.0258	-0.0259	0.0258
2	-0.0168	-0.0146	-0.0259	-0.0146	0.0254	-0.0259	0.0254
3	-0.0174	-0.0151	-0.0254	-0.0151	0.0247	-0.0254	0.0247
4	-0.0174	-0.0155	-0.0249	-0.0155	0.0247	-0.0249	0.0247
5	-0.0169	-0.0148	-0.0243	-0.0148	0.0256	-0.0243	0.0256
6	-0.0161	-0.0154	-0.0242	-0.0154	0.0258	-0.0242	0.0258
7	-0.0158	-0.0157	-0.0246	-0.0157	0.0246	-0.0246	0.0246
8	-0.0150	-0.0156	-0.0246	-0.0156	0.0256	-0.0246	0.0256
9	-0.0154	-0.0159	-0.0244	-0.0159	0.0258	-0.0244	0.0258
10	-0.0150	-0.0160	-0.0240	-0.0160	0.0263	-0.0240	0.0263
11	-0.0152	-0.0173	-0.0236	-0.0173	0.0261	-0.0236	0.0261
12	-0.0145	-0.0179	-0.0236	-0.0179	0.0255	-0.0236	0.0255
13	-0.0147	-0.0180	-0.0233	-0.0180	0.0249	-0.0233	0.0249
14	-0.0134	-0.0173	-0.0238	-0.0173	0.0246	-0.0238	0.0246
15	-0.0137	-0.0176	-0.0231	-0.0176	0.0255	-0.0231	0.0255
16	-0.0138	-0.0166	-0.0228	-0.0166	0.0254	-0.0228	0.0254

Gambar 3.3 *Database* level rendah



Workspace

Stack: Base

Name	Value	Min	Max
jumlah_data	6	6	6
label	<1200x10 double>	-0.05...	0.048
status	'LevelSedang'		

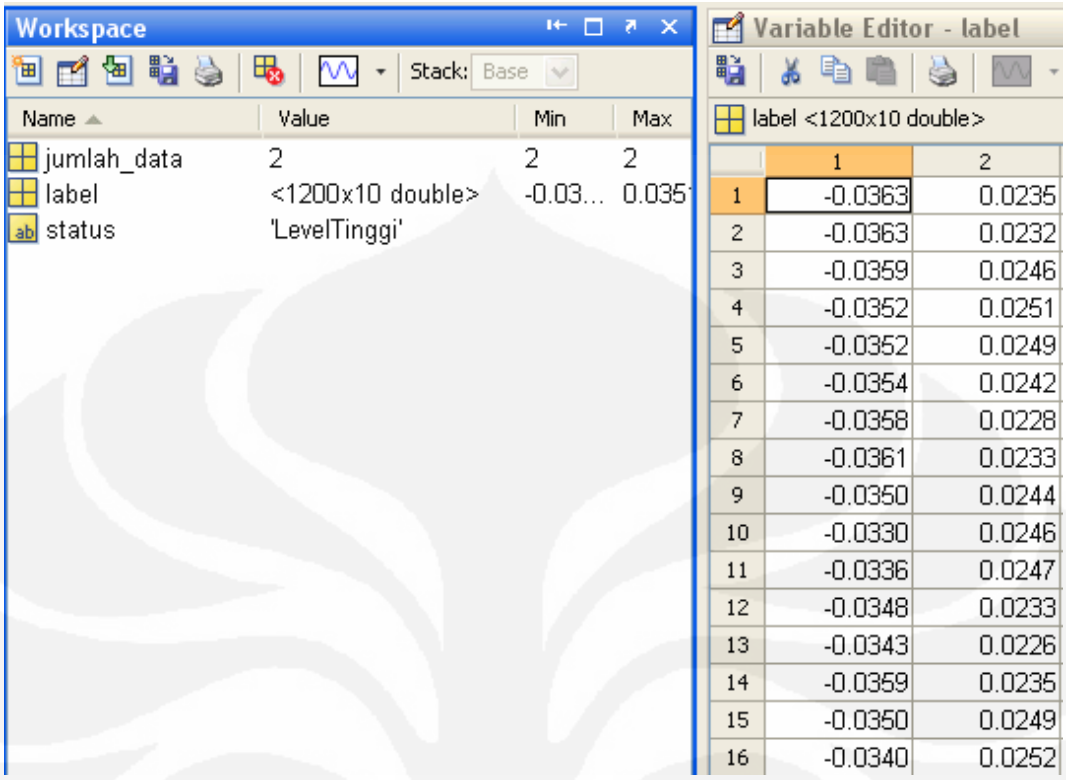
Variable Editor - label

Stack: Base

label <1200x10 double>

	1	2	3	4	5	6
1	0.0329	0.0202	0.0329	-0.0101	-0.0254	-0.0101
2	0.0332	0.0214	0.0332	-0.0099	-0.0258	-0.0099
3	0.0331	0.0215	0.0331	-0.0098	-0.0274	-0.0098
4	0.0323	0.0213	0.0323	-0.0090	-0.0285	-0.0090
5	0.0331	0.0197	0.0331	-0.0096	-0.0278	-0.0096
6	0.0335	0.0188	0.0335	-0.0096	-0.0257	-0.0096
7	0.0335	0.0191	0.0335	-0.0100	-0.0253	-0.0100
8	0.0328	0.0194	0.0328	-0.0106	-0.0270	-0.0106
9	0.0321	0.0194	0.0321	-0.0103	-0.0287	-0.0103
10	0.0326	0.0198	0.0326	-0.0094	-0.0299	-0.0094
11	0.0327	0.0195	0.0327	-0.0088	-0.0291	-0.0088
12	0.0322	0.0206	0.0322	-0.0090	-0.0294	-0.0090
13	0.0323	0.0219	0.0323	-0.0094	-0.0288	-0.0094
14	0.0320	0.0210	0.0320	-0.0103	-0.0288	-0.0103
15	0.0323	0.0205	0.0323	-0.0112	-0.0292	-0.0112
16	0.0325	0.0194	0.0325	-0.0125	-0.0273	-0.0125

Gambar 3.4 *Database* level sedang



	1	2
1	-0.0363	0.0235
2	-0.0363	0.0232
3	-0.0359	0.0246
4	-0.0352	0.0251
5	-0.0352	0.0249
6	-0.0354	0.0242
7	-0.0358	0.0228
8	-0.0361	0.0233
9	-0.0350	0.0244
10	-0.0330	0.0246
11	-0.0336	0.0247
12	-0.0348	0.0233
13	-0.0343	0.0226
14	-0.0359	0.0235
15	-0.0350	0.0249
16	-0.0340	0.0252

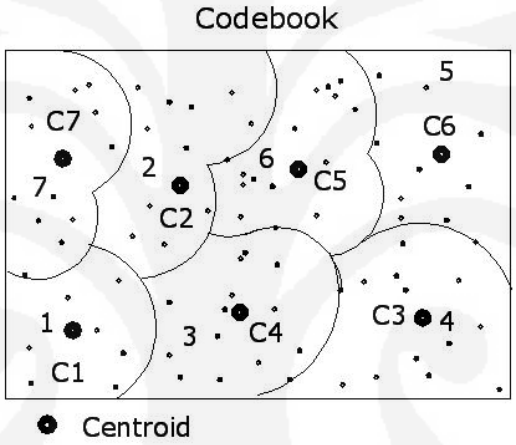
Gambar 3.5 *Database* level tinggi

3.2.2 Pembentukan *Codebook*

Proses selanjutnya adalah proses penggabungan ketiga label (rendah, sedang, dan tinggi) ke dalam sebuah *database* yang diberi nama *code*book_korona.mat**,** *database* ini berisi 2 buah parameter yaitu : *Code* dan *names*. Variabel *code* menunjukan ukuran *codebook*, jumlah iterasi dan label yaitu hasil hasil perhitungan**.** Ukuran *codebook* yang tersedia dalam program ini adalah 32, 64, dan 128, 256. Dimana keempat ukuran *codebook* ini akan dijadikan bahan perbandingan untuk dilihat berapa nilai *codebook* yang paling sesuai pada proses identifikasi korona. Jumlah iterasi merupakan banyaknya proses pengulangan yang dilakukan dalam menentukan *centroid* guna mendapatkan *centroid* yang cukup presisi. Menurut teori semakin besar jumlah iterasinya, maka akan semakin resisi letak *centroid* yang didapat, namun dengan jumlah iterasi yang tinggi maka proses pembuatan *codebook* akan berjalan sangat lambat, oleh karena itu iterasi yang dilakukan juga tidak perlu teralu besar. Dalam ini ditentukan *default* ntuk besarnya iterasi adalah 10 dengan harapan letak *centroid* yang diperoleh cukup presisi dan waktu proses relatif cepat. *Codebook* dan *centroid* ditunjukan oleh

Gambar 3.6. Sedangkan label adalah hasil perhitungan untuk mendapatkan parameter HMM. Variabel *names* berisi 3 buah label atau kondisi yang diharapkan dari identifikasi sebagai hasil proses labelisasi, dalam program ini ketiga label tersebut adalah LevelRendah, LevelSedang dan LevelTinggi. Berikut ini merupakan listing program pembentukan *codebook* :

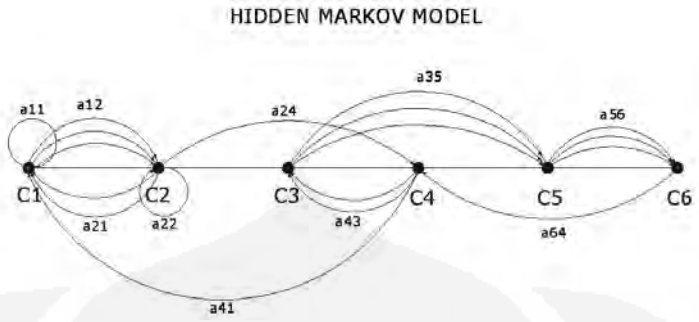
```
Code=VQ_training(speech,ukuran_codebook,jumlah_iter
asi);
F=extraction(speech,100,78,12000);
    save feat_default F
    [a,b]=size(F);

Cfinal=split2(F,floor(log(M)/log(2)),iteration);
    Code=Cfinal;
    save codebook Code
```



Gambar 3.6 *Code*book

3.2.3 Pembentukan HMM

Setelah terbentuk nilai *codebook* kemudian dilakukan proses pembentukan HMM. Pada program, nilai HMM ini disimpan dalam sebuah *database* bernama hmm_korona.mat. *Database* ini berisi beberapa variabel, yaitu : A1, A2, A3, B1, B2, B3, available, p01, p02 dan p03. Kesemua variabel tersebut adalah parameter HMM, hasil pengolahan *training* yang akan dibandingkan dengan identifikasi input yang sesungguhnya. Nilai identifikasi yaitu probability (p01, p02, dan p03) yang mendekati dari salah satu parameter itu akan diidentifikasi sesuai dengan label yang bersangkutan. Vaariasi variabel tersebut ada tiga sesuai dengan variasi label hasil identifikasi. Proses pembentukan HMM ditunjukan oleh Gambar 3.7.



a11 = 1/16 b1 = 1/16
a12 = 3/16 b2 = 2/16
a22 = 1/16 b3 = 2/16
a24 = 1/16 b4 = 3/16
a35 = 3/16 b5 = 2/16
a56 = 3/16 b6 = 1/16
a64 = 1/16
a43 = 2/16
a41 = 1/16
a21 = 2/16

LoP = b1.a11.b2.a24.b3.a35.b4.a43.b5.a56.b6.a64.b4.a41.b2.a22

Gambar 3.7 Gambar Perhitungan HMM

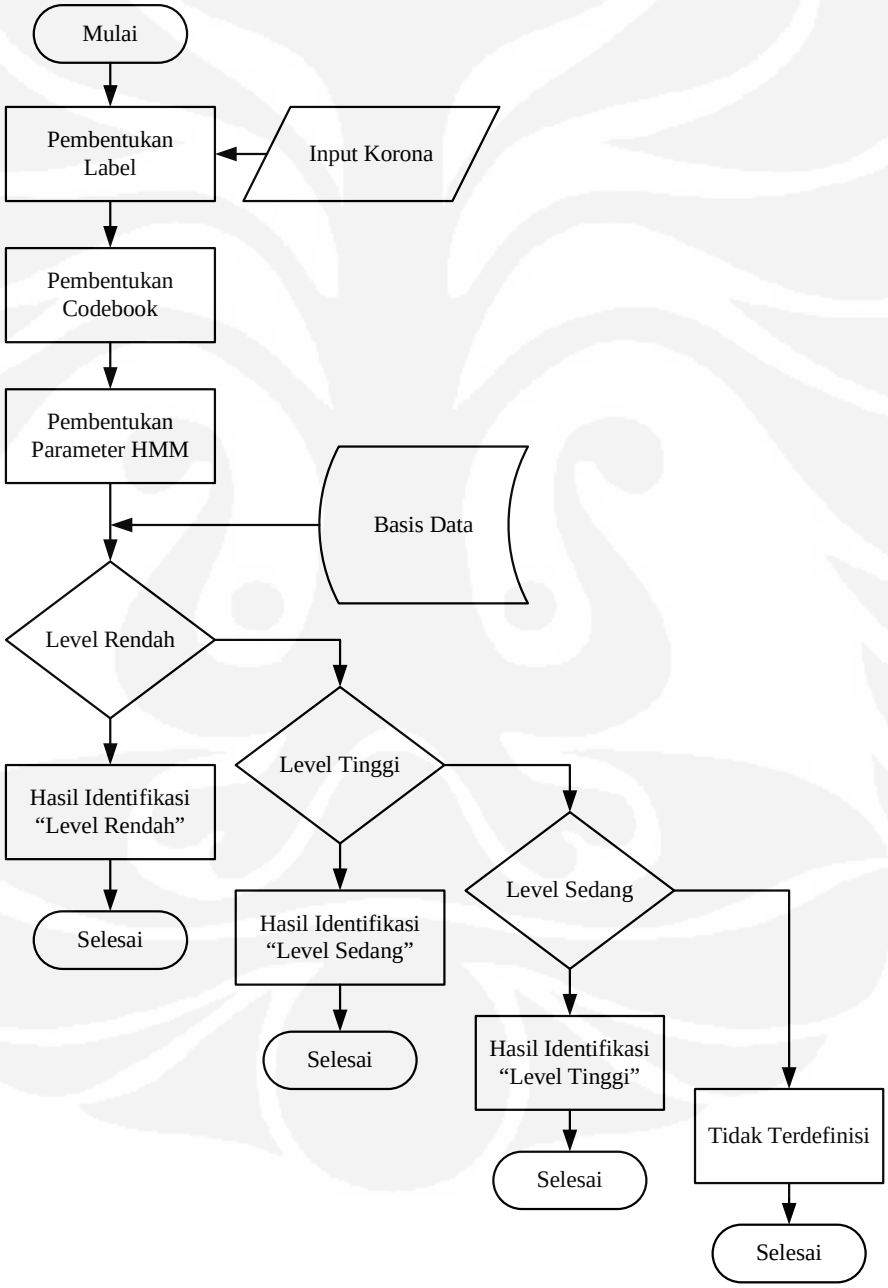
Berikut ini merupakam *listing* program pembentukan parameter HMM :

```
function Cfinal=VQ_training(speech,M,iteration);
    F=extraction(speech,100,78,12000);
   % [baris,kolom]=size(F);
    save feat_default F
    [a,b]=size(F);
    Cfinal=split2(F,floor(log(M)/log(2)),iteration);
    Code=Cfinal;
    save codebook Code
```

3.3 Proses Pengenalan

Proses pengenalan yaitu proses pengenalan input korona dengan membandingkan parameter HMM yang telah tersedia di dalam *database*. Proses pengenalan meliputi pembentukan domain frekuensi sinyal input korona yang masih berbentuk analog dengan metoda FFT (*Fast Fourier Transform*). Spektrum frekuensi-nya akan membentuk nilai vektor *real* dan imajiner yang akan dipetakan dalam *codebook* dalam bentuk sample points. Selanjutnya sampel poin terdekat

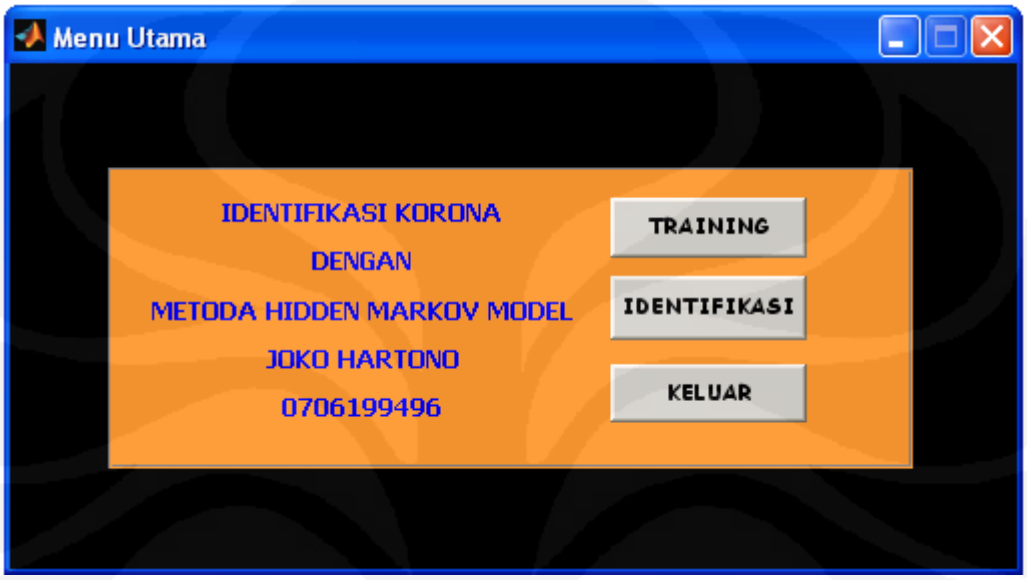
akan dikuantisasikan ke satu titik vektor yang dinamakan *centroid*. Letak dari *centroid* ini kemudian dicocokan dengan letak *centroid* yang ada pada *codeword* dalam *database*. Hasil perbandingan adalah urutan kode observasi. Kemudian matriks dari nilai observasi yang didapat akan dicocokan dengan matriks – matriks dari parameter – parameter HMM dalam *database* kemudian hitung besar *log of probability* untuk nilai korona yang akan dipengenalan kemudian tampilkan hasil yang memiliki *log of probability* tertinggi sebagai proses pengenalan. Diagram alir proses *identifikasi* ditunjukan oleh Gambar 3.8.



Gambar 3.8 Diagram Alir Proses Pengenalan

3.4 Software

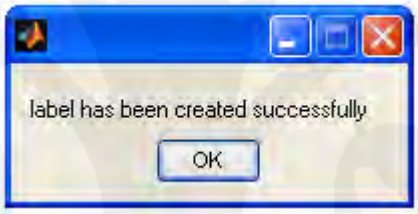
Software menggunakan MATLAB versi R2008a, berikut ini merupakan penjelasan cara pengoperasian berikut tampilannya. Gambar 3.9 merupakan tampilan menu utama ketika program pertama kali dijalankan. Ada 3 pilihan yaitu : *Training*, *Identifikasi* dan keluar.



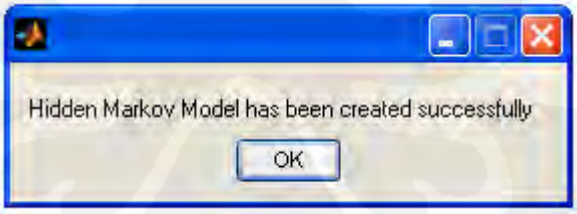
Gambar 3.9 Menu Utama

Gambar 3.10 merupakan menu *training* yang meliputi 3 proses, yaitu : Labelisasi, pembentukan *codebook* dan pembentukan parameter HMM. Langkah pertama yang dilakukan adalah menentukan nilai durasi pencuplikan, jumlah *iterasi*, besar *codebook*, level korona kemudian menentukan input korona sebagai data latih yang tersimpan di dalam pc. Proses *labelisasi* ditandai dengan munculnya tampilan yang ditunjukan oleh Gambar 3.11. Langkah selanjutnya adalah pembentukan *codebook* dan HMM dengan cara menekan tombol proses, Gambar 3.12 menunjukan indikasi bahwa *codebook* dan parameter HMM telah terbentuk.

Gambar 3.10 Menu Pengisian *Database*



Gambar 3.11 Indikasi Terbentuknya Label



Gambar 3.12 Indikasi Terbentuknya Parameter HMM

Sedangkan Gambar 3.13 adalah menu pengenalan korona, langkah pertama pengoperasiannya adalah dengan mengatur nilai *iterasi*, durasi pencuplikan, ukuran *codebook* dan input korona yang akan diidentifikasi. Kemudian menekan tombol proses**,** hasil identifikasi ditunjukan oleh text "HASIL IDENTIFIKASI".

Gambar 3.13 Menu Pengenalan

BAB IV
HASIL PENGUJIAN DAN ANALISIS

4.1 Hasil Uji Coba

Jumlah data sampel setiap level adalah 9 buah, ujicoba dilakukan dalam 2 bagian, pertama dengan 3 buah data latih kemudian dengan 5 buah data latih. Dimana pengujian dilakukan dengan *codebook* yang berbeda-beda, yaitu : 32, 128 dan 512. Untuk bagian pertama data korona 1 hingga data korona 3 menjadi data latih, sisanya menjadi data uji sedangkan pengujian yang kedua data korona 1 hingga data korona 5 menjadi data latih, sisanya menjadi data uji. Jumlah iterasi yang diuji adalah 5 kali dan 10 kali serta waktu pencuplikannya 0.01 detik, 0.1 detik dan 1 detik. Hasil pengujian ditunjukan oleh Tabel 4.1, Tabel 4.2, Tabel 4.3, Tabel 4.5, Tabel 4.5, dan Tabel 4.5. Sedangkan Tabel 4.4, Tabel 4.8 dan Tabel 4.9 merupakan tabel rekap. Hasil pengujian ditujukan untuk mengetahui variasi ukuran *codebook*, jumlah *iterasi*, jumlah data latih dan waktu pencuplikan yang menghasilkan akurasi tertinggi.

4.1.1 Pengujian Pertama

Pengujian dilakukan menggunakan data korona 1, data korona 2 dan data korona 3 sebagai data latih, sisanya menjadi data uji.

Tabel 4.1 Hasil pengujian pertama dengan codebook = 32

No	Nama	Hasil					
		Iterasi					
		5			10		
		Pencuplikan (s)					
		0.01	0.1	1	0.01	0.1	1
1	Rendah_4	Ok	-	-	Ok	-	-
2	Rendah_5	-	-	-	Ok	Ok	-
3	Rendah_6	-	-	-	-	-	Ok
4	Rendah_7	-	-	-	-	-	-
5	Rendah_8	-	-	-	-	-	-
6	Rendah_9	-	-	-	-	-	-
7	Sedang_4	-	-	-	-	-	-
8	Sedang_5	-	-	-	-	-	-

9	Sedang_6	-	-	-	-	-	-
10	Sedang_7	Ok	Ok	Ok	Ok	Ok	Ok
11	Sedang _8	-	Ok	Ok	-	Ok	Ok
12	Rendah_9	-	Ok	Ok	Ok	-	-
13	Tinggi_4	-	-	-	-	-	-
14	Tinggi_5	Ok	-	Ok	-	Ok	Ok
15	Tinggi_6	-	Ok	Ok	-	Ok	Ok
16	Tinggi_7	Ok	-	Ok	Ok	-	Ok
17	Tinggi_8	Ok	Ok	-	Ok	-	-
18	Tinggi_9	-	-	-	-	-	-
Akurasi Per Pencuplikan		27.8%	27.8%	30%	30%	27.8%	30%
Akurasi Per Iterasi		28.5%			29.2%		
Akurasi Total		28.87%					

Tabel 4.2 Hasil pengujian pertama dengan codebook = 128

No	Nama	Hasil					
		Iterasi					
		5			10		
		Pencuplikan (s)					
		0.01	0.1	1	0.01	0.1	1
1	Rendah_4	Ok	-	-	-	-	-
2	Rendah_5	-	-	-	Ok	Ok	-
3	Rendah_6	-	-	-	-	-	Ok
4	Rendah_7	-	-	-	-	-	-
5	Rendah_8	-	-	-	-	-	-
6	Rendah_9	-	-	-	-	-	-
7	Sedang_4	-	-	-	-	-	-
8	Sedang_5	-	-	-	-	-	-
9	Sedang_6	-	-	-	-	-	-
10	Sedang_7	Ok	Ok	Ok	Ok	Ok	Ok
11	Sedang _8	-	Ok	Ok	-	Ok	Ok
12	Rendah_9	Ok	Ok	Ok	Ok	-	-
13	Tinggi_4	-	-	-	-	-	-
14	Tinggi_5	Ok	-	Ok	-	Ok	Ok
15	Tinggi_6	-	Ok	Ok	-	Ok	Ok
16	Tinggi_7	Ok	-	Ok	Ok	-	Ok
17	Tinggi_8	Ok	Ok	-	Ok	Ok	-
18	Tinggi_9	-	Ok	-	-	-	-
Akurasi Per Pencuplikan		30%	30%	30%	27.8%	30%	30%
Akurasi Per Iterasi		29.2%			29.2%		
Akurasi Total		29.2%					

Tabel 4.3 Hasil pengujian pertama dengan codebook = 512

No	Nama	Hasil					
		Iterasi					
		5			10		
		Pencuplikan (s)					
		0.01	0.1	1	0.01	0.1	1
1	Rendah_4	Ok	-	-	Ok	-	-
2	Rendah_5	-	Ok	-	-	Ok	-
3	Rendah_6	-	-	-	-	-	Ok
4	Rendah_7	-	-	-	-	-	-
5	Rendah_8	-	-	-	-	-	-
6	Rendah_9	-	-	-	-	-	-
7	Sedang_4	-	-	-	-	-	-
8	Sedang_5	-	-	-	-	-	-
9	Sedang_6	-	-	-	-	-	-
10	Sedang_7	Ok	Ok	Ok	Ok	Ok	Ok
11	Sedang _8	Ok	Ok	Ok	Ok	Ok	Ok
12	Rendah_9	Ok	Ok	Ok	Ok	Ok	Ok
13	Tinggi_4	Ok	-	Ok	Ok	-	-
14	Tinggi_5	Ok	Ok	Ok	Ok	Ok	Ok
15	Tinggi_6	-	Ok	Ok	-	Ok	Ok
16	Tinggi_7	Ok	-	Ok	Ok	-	Ok
17	Tinggi_8	Ok	Ok	-	Ok	Ok	-
18	Tinggi_9	-	Ok	-	-	Ok	-
Akurasi Per Pencuplikan		44.4 %	44.4 %	39%	44.4 %	44.4 %	39%
Akurasi Per Iterasi		42.6%			42.6%		
Akurasi Total		42.6%					

Tabel 4.4 Hasil rekap pengujian Pertama

Codebook	Iterasi	Pencuplikan	Persentase
32	5	0.01	30%
		0.1	30%
		1	27.8%
	10	0.01	30%
		0.1	30%
		1	30%
128	5	0.01	30%
		01	30%
		1	30%
	10	0.01	30%

		0.1	30%
		1	30%
512	5	0.01	44.4 %
		0.1	44.4 %
		1	39%
	10	0.01	44.4 %
		0.1	44.4 %
		1	39%

4.1.2 Pengujian Kedua

Pengujian dilakukan menggunakan data korona 1, data korona 2, data korona 3, data korona 4 dan data korona 5 latih, sisanya menjadi data uji.

Tabel 4.5 Hasil pengujian kedua dengan codebook = 32

No	Nama	Hasil					
		Iterasi					
		5			10		
		Pencuplikan (s)					
		0.01	0.1	1	0.01	0.1	1
1	Rendah_6	Ok	Ok	-	Ok	-	-
2	Rendah_7	-	-	-	-	-	-
3	Rendah_8	-	-	-	-	-	-
4	Rendah_9	-	-	-	-	-	-
5	Sedang_6	-	-	-	-	-	-
6	Sedang_7	-	Ok	-	-	-	-
7	Sedang_8	-	-	-	-	-	-
8	Sedang_9	-	-	-	-	-	-
9	Tinggi_6	-	Ok	-	-	Ok	Ok
10	Tinggi_7	Ok	-		Ok	Ok	Ok
11	Tinggi_8	Ok	-	Ok	Ok	Ok	Ok
12	Tinggi_9	Ok	Ok	Ok	Ok	Ok	Ok
Akurasi Per Pencuplikan		33.3 %	33.3%	16.7%	33.3%	33.3%	33.3%
Akurasi Per Iterasi		27.8 %			33.3%		
Akurasi Total		30.5 %					

Tabel 4.6 Hasil pengujian kedua dengan codebook = 128

No	Nama	Hasil					
		Iterasi					
		5			10		
		Pencuplikan (s)					
		0.01	0.1	1	0.01	0.1	1
1	Rendah_6	Ok	-	-	-	-	-
2	Rendah_7	-	-	-	-	-	-
3	Rendah_8	-	-	-	-	-	-
4	Rendah_9	-	-	-	-	-	-
5	Sedang_6	Ok	Ok	-	Ok	Ok	Ok
6	Sedang_7	Ok	Ok	Ok	Ok	Ok	Ok
7	Sedang_8	-	Ok	Ok	Ok	-	Ok
8	Sedang_9	Ok	Ok	Ok	Ok	Ok	Ok
9	Tinggi_6	-	-	-	-	-	-
10	Tinggi_7	-	-	-	-	-	-
11	Tinggi_8	-	-	-	-	Ok	-
12	Tinggi_9	-	-	Ok	-	-	-
Akurasi Per Pencuplikan		33.3 %	50%	50%	33.3	50%	50%
Akurasi Per Iterasi		27.8 %			33.3%		
Akurasi Total		30.5 %					

Tabel 4.7 Hasil pengujian kedua dengan codebook = 512

No	Nama	Hasil					
		Iterasi					
		5			10		
		Pencuplikan (s)					
		0.01	0.1	1	0.01	0.1	1
1	Rendah_6	Ok	Ok	-	Ok	Ok	-
2	Rendah_7	Ok	Ok	-	Ok	Ok	-
3	Rendah_8	-	-	-	-		
4	Rendah_9	-	-	-	-		
5	Sedang_6	-	-	-	-	-	-
6	Sedang_7	-	-	-	-	-	-
7	Sedang_8	-	-	-	-	-	-
8	Sedang_9	-	-	-	-	-	-
9	Tinggi_6	Ok	Ok	-	Ok	Ok	Ok
10	Tinggi_7	Ok	Ok	Ok	Ok	Ok	Ok
11	Tinggi_8	Ok	Ok	Ok	Ok	Ok	Ok

12	Tinggi_9	Ok	Ok	Ok	Ok	Ok	Ok
Akurasi Per Pencuplikan		50%	50%	33.3	50%	50%	33.3
Akurasi Per Iterasi		47.2%			44.43%		
Akurasi Total		45.8%					

Tabel 4.8 Hasil rekap pengujian kedua

Codebook	Iterasi	Pencuplikan	Persentase
32	5	0.01	33.3%
		0.1	33.3%
		1	16.7%
	10	0.01	33.3%
		0.1	33.3%
		1	33.3%
128	5	0.01	33.3%
		01	33.3%
		1	33.3%
	10	0.01	33.3%
		0.1	33.3%
		1	33.3%
512	5	0.01	50%
		0.1	50%
		1	33.3%
	10	0.01	50%
		0.1	50%
		1	50%

Tabel 4.9 Perbandingan akurai maksimum pengujian pertama dan kedua

No	Pengujian	Akurasi
1	Pertama	44.4 %
2	Kedua	50 %

4.2 Analisis

Faktor yang akan dianalisis dalam pengujian ini adalah, ukuran *codebook*, jumlah *iterasi*, waktu pencuplikan dan jumlah data latih sehingga dapat diketahui setingan yang dapat menghasilkan nilai akurasi tertinggi.

4.2.1 Analisis Terhadap Ukuran Codebook

Analisis terhadap *codebook* dilakukan terhadap pengujian kedua yaitu dengan 5 buah sampel uji. Pengaruh ukuran *codebook* terhadap akurasi pengujian

software dapat dilihat pada Tabel 4.5, Tabel 4.6 dan Tabel 4.7. Dari tabel tersebut dapat disimpulkan bahwa semakin besar ukuran *codebook* yang digunakan maka akurasi pembacaan akan semakin tinggi. Terdapat peningkatan nilai akurasi ketika ukuran *codebook* yang digunakan lebih besar dimana ukuran *codebook* 512 menghasilkan akurasi yang lebih baik (50%) dibandingkan dengan ukuran *codebook* 32 dan 128 dengan kenaikan persentase sebesar (50% - 30.5% = 19.5%). Hal ini disebabkan karena ukuran *codebook* yang besar membuat jumlah *codeword* (*centorid*) semakin banyak. Banyaknya *centroid* ini membuat proses kuantisasi pemilihan nilai vektor data semakin teliti, sehingga pemetaan terhadap vektor data dapat dilakukan dengan jarak yang lebih kecil. Dengan kata lain, distorsi VQ (jarak antara sebuah vektor data dengan *codeword* terdekat) pada akhir *iterasi* akan semakin kecil.

Walaupun ukuran *codebook* yang lebih besar menghasilkan nilai akurasi yang lebih tinggi, namun level korona yang dapat diidentifikasi pada *codebook* berukuran rendah tidak serta merta dapat diidentifikasi pada *codebook* yang berukuran tinggi, hal ini dapat dilihat pada Tabel 4.1 dan Tabel 4.2 dimana data korona dengan level tinggi (Tinggi_6 dan Tinggi_7) dapat diidentifikasi dengan baik menggunakan ukuran *codebook* 32 namun ketika menggunakan ukuran *codebook* 128, level tersebut tidak sepenuhnya dapat diidentifikasi dengan baik oleh *software*. Hal ini dikarenakan karakteristik sinyal audio korona untuk level rendah, sedang dan tinggi identik dimana penentuan levelnya dilakukan hanya dengan bantuan software *wavesurfer* yang menganalisis bentuk gelombang dan mengeluarkan suaranya.

4.2.2 Analisis Terhadap Jumlah *Iterasi*

Analisis terhadap Jumlah ***Iterasi*** dilakukan terhadap pengujian kedua yaitu dengan 5 buah sampel uji. Pengaruh jumlah *iterasi* terhadap akurasi pengujian *software* dapat dilihat pada Tabel 4.5. Dimana dengan ukuran *codebook* yang sama namun dengan jumlah iterasi yang lebih tinggi, dihasilkan akurasi sebesar 33.3 % yaitu sebanyak 10 kali dibandingkan dengan jumlah *iterasi* sebanyak 5 kali yang hanya menghasilkan akurasi sebesar 16.7 % dengan kenaikan sebesar (33.3% - 16.7 % = 16.6 %).

Dengan jumlah *iterasi* yang lebih banyak karakteristik gelombang korona akan semakin baik perhitungan pembentukan *codebook*nya karena dikhawatirkan prosesnya terlewat atau tidak sempurna.

4.2.3 Analisis Terhadap Waktu Pencuplikan

Analisis terhadap waktu pencuplikan dilakukan terhadap pengujian kedua yaitu dengan 5 buah sampel uji. Pengaruh waktu pencuplikan terhadap akurasi pengujian *software* dapat dilihat pada Tabel 4.7. Dimana dengan ukuran *codebook* dan jumlah *iterasi* yang sama namun dengan waktu pencuplikan yang lebih tinggi yaitu 0.01 detik dihasilkan akurasi sebesar 50 % dibandingkan dengan waktu pencuplikan sebesar 1 detik yang hanya menghasilkan akurasi sebesar 33.3 % dengan kenaikan sebesar 16.7%.

Hal ini dikarenakan semakin tinggi waktu pencuplikan maka titik sampling pada proses pembentukan *codebook* akan semakin banyak sehingga akurasi akan semakin tinggi.

4.2.4 Analisis Terhadap Jumlah Data Latih

Pengaruh jumlah data latih terhadap nilai akurasi pengujian dapat dilihat pada Tabel 4.9, dimana pengujian pertama yang menggunakan data latih sebanyak 3 buah sampel menghasilkan akurasi maksimal sebesar 44.4%, sedangkan pengujian kedua yang menggunakan data latih sebanyak 5 buah sampel menghasilkan nilai akurasi maksimal sebesar 50%.

Dari data tabel di atas dapat diketahui bahwa dengan memperbanyak data latih maka nilai akurasi akan lebih baik, hal ini dikarenakan kemiripan antar level korona akan berkurang sehingga tiap level akan memberikan karakteristik yang unik sehingga mudah dibedakan.

BAB V
KESIMPULAN

Hasil analisis dapat disimpulkan sebagai berikut :

1.) *Software* yang dirancang mampu mengidentifikasi korona dengan akurasi maksimal sebesar 50% dengan ukuran *codebook* 512, jumlah *iterasi* 10 kali, waktu pencuplikan 0.01 detik dan jumlah data latih sebanyak 10.

2.) Faktor yang mempengaruhi rendahnya nilai akurasi adalah terbatasnya data latih yang hanya sebanyak 5 sampel.

DAFTAR ACUAN

[1] Al-Faraj, M.A., Farag, A.S., Shewhdi, M.H., *Enviromental Effect On High Voltage AC Transmission Lines Audible Noise, IEEE Transaction On Power Delivery*

[2] Kuffel, E., Zaengl, W.S., *High Voltage Engineering Fundamentals*, Pergamon Press, Oxford, 1984.

[3] Korona Detection Technologi (August 2008). Diakses 4 Agustus 2008 dari : http://www.corona-technology-course.com.

[4] Corona & Testing - Who, What, When, Where & Why (August 2008). Diakses 4 Agustus 2008 dari: http://www.plantmaintenance.com/articles/corona.shtml

[5] Rabiner, H.L., *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, IEEE Electrical Insulation Magazine Vol. 77, February 1989.

[6] Arman D.Diponegoro, et al.,IJJS September 6 2006, *"The Comparison of Vector Quantization Algorithms in Fish Species Acoustic Voice Recognition Using Hidden Markov Based on the phase detection of schooling reflection acoustic wave", Electrical Engineering Department, University of Indonesia*, Indonesia.

[7] Ahmad Mujadid Amin. "Pengenalan Suara Manusia Menggunakan *Hidden Markov Model*". Skripsi. Program Sarjana Fakultas Teknik Universitas Indonesia. 2007.

DAFTAR PUSTAKA

"Using Ultrasound for High Voltage Insulation Testing", http://www.mt-online.com/articles/3-97ultra.cfm, 6 Maret 2006.

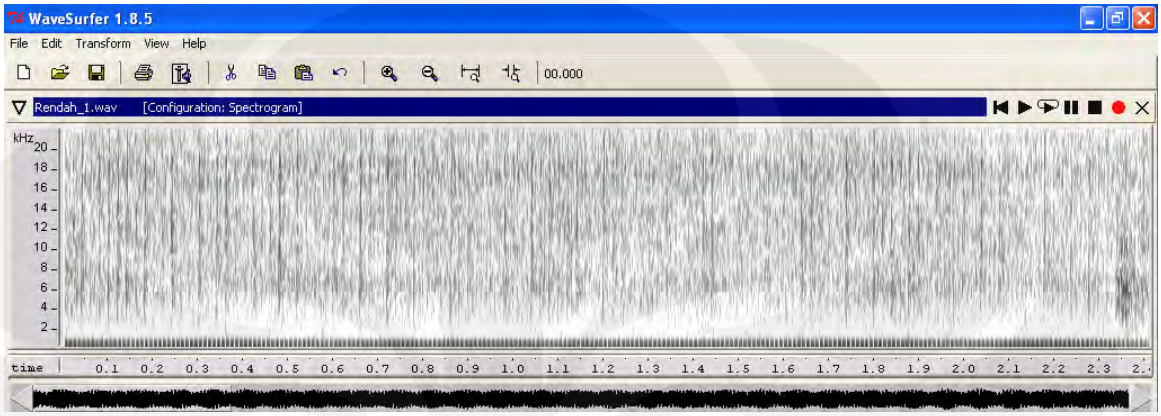
Rabiner, H.L., *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, IEEE Electrical Insulation Magazine Vol. 77, February 1989.

Emmanuel C, Ifeachor dan Barri W, Jervis. *Digital Signal Processing*. *A Practical Approach, Second Edition*. Prentice Hall.

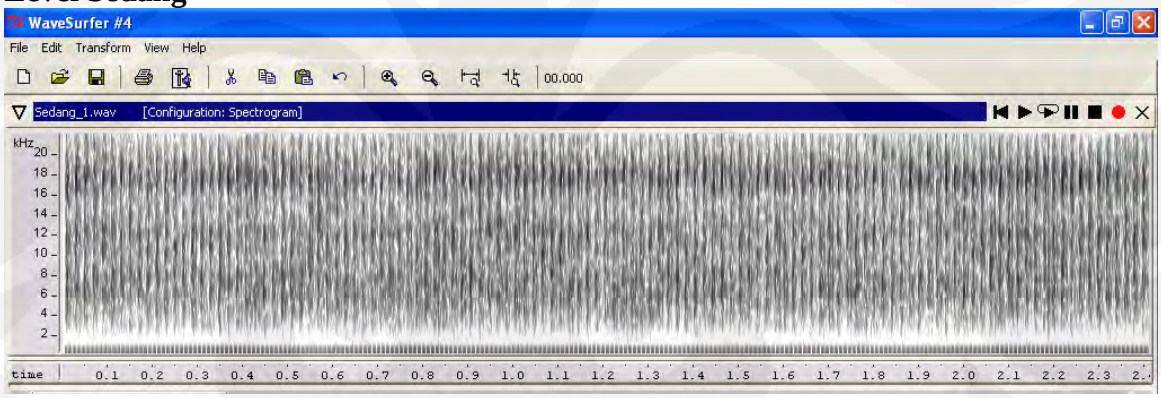
LAMPIRAN

Spektrum frekuensi korona untuk setiap level menggunakan *software wafesurfer*.

Level Rendah



Level Sedang



Level Tinggi