BAB 3
METODE PENELITIAN

3.1 ALAT YANG DIGUNAKAN

Pada penelitian yang dilakukan ini digunakan beberapa alat dan *software* yang digunakan dalam membantu penelitian seperti:

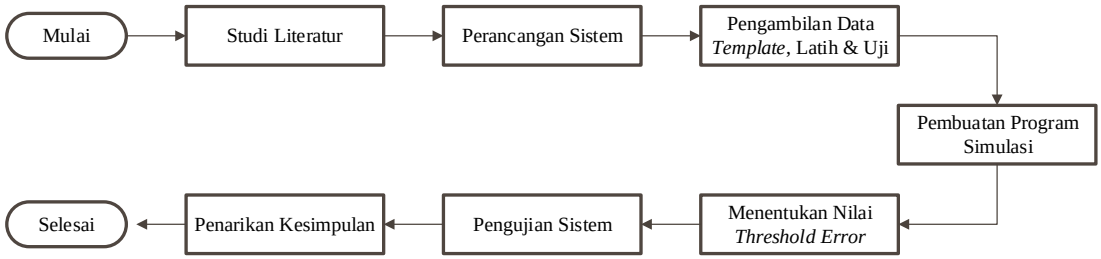
1. MATLAB R2019a Matlab adalah *software* yang dikembangkan oleh MathWork.Inc., yang sangat bermanfaat untuk menyelesaikan berbagai masalah komputasi numerik. MATLAB mengintegrasikan komputasi matematik, visualisasi, dan bahasa pemrograman untuk memberikan lingkungan fleksibel bagi komputasi teknis. Arsitektur terbukanya membuat pengguna mudah dalam mengeksplorasi data, menciptakan algoritma dan menciptakan beberapa perangkat grafik (GUI). Dikenal karena perhitungan vektor dan matriks dengan kecepatan yang tinggi, MATLAB menawarkan solusi terhadap permasalahan secara matematik dan secara visual. MATLAB akan digunakan untuk pengolahan data, mulai dari prapengolahan hingga tahap pengenalan ayat, pengenalan kata dan mendapatkah hasil akhir.

2. *Microphone* digunakan untuk mengambil data penutur dari pembacaan surah Al-Kautsar, dapat menggunakan *microphone eksternal* atau *microphone* yang terdapat pada *headphone*.

3. Spesifikasi komputer yang digunakan dalam penelitian ini menggunakan spesifikasi: *processor* Core i5 7500, Ram 8 GB, HDD 1 TB, Nvidia Geforce GTX 1060 dan menggunakan *Operating System* Windows 10 64bit.

3.2 DIAGRAM ALUR PENELITIAN

Penelitian ini dilakukan dengan mengacu pada diagram yang ditunjukkan oleh Gambar 3.1 tahap pertama dari penelitian adalah memahami objek penelitian dengan cara memahami beberapa tinjauan pustaka yang terkait dengan penelitian yang akan dilakukan. Dengan adanya tinjauan pustaka maka dapat memahami konsep dasar dari dilakukannya penelitian dan sebagai acuan terhadap penelitian yang dilakukan mengenai parameter yang menandakan hasil penelitian baik.



Gambar 3.1 Diagram Alur Penelitian

Tahap selanjutnya adalah perancangan sistem, dalam perancangan sistem ini dilakukan penentuan metode yang akan digunakan. Selanjutnya pengambilan data *template,* data latih dan data uji, data *template* adalah data yang diambil dari penutur yang telah fasih dalam membaca Al-Qur'an yang nantinya akan dijadikan sebagai *template*. Sedangkan data latih dan uji adalah data yang diambil secara *real time*. Tahap berikutnya adalah pembuatan program simulasi dengan menggunakan MATLAB. Selanjutnya menentukan nilai ambang batas, dimana nilai ambang batas yang dicari adalah nilai ambang batas galat (*threshold error*). Kemudian pengujian sistem, dimana sistem akan diuji keakurasianya dalam mendeteksi nilai ambang batas galat dari data latih maupun data uji. Nilai galat dari data latih dan uji akan dibandingkan dengan nilai ambang batas yang didapatkan dari data *template*. Tahap terakhir adalah penarikan kesimpulan, bahwa galat antara data *template* dengan data latih maupun data uji harus semakin kecil, apabila lebih besar dari nilai ambang batas maka bacaan dianggap tidak tepat, akan tetapi jika lebih kecil dari nilai ambang batas maka bacaan dianggap sudah tepat.

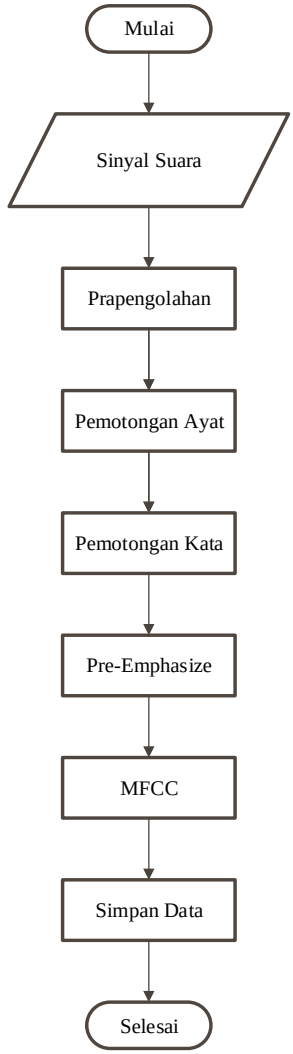
3.3 PERANCANGAN SISTEM

Pada penelitian ini dirancang sebuah sistem yang dapat mendeteksi ketepatan dalam pembacaan surah Al-Kautsar dengan mendapatkan ciri dari setiap kata pada surah Al-Kautsar. Pengujian data tutur menggunakan 10 identitas penutur yang berbeda dengan masing-masing membaca surah Al-Kautsar baik laki-laki maupun perempuan, yang telah fasih membaca Al-Qur'an atau belum. Setiap penutur uji direkam sebanyak 6 kali perekaman, 3 perekaman pembacaan secara benar dan 3 perekaman pembacaan yang disalahkan pada salah satu kata dalam surah Al-Kautsar sehingga total dari data terdapat 60 rekaman penutur (30 bacaan benar dan 30 bacaan salah). Untuk data *template* perekaman dilakukan dengan penutur yang telah fasih

dalam pembacaan Al-Qur'an sebagai acuan dalam pembacaan Al-Qur'an secara benar. Perekaman penutur dilakukan dengan *microphone*, frekuensi cuplik yang digunakan adalah sebesar 44.100 Hz yang hasil rekamanya akan berformat *.wav. Data penutur yang sudah diperoleh berformat *.wav akan diolah menggunakan *software* MATLAB R2019a dengan metode ekstraksi ciri MFCC dan pendeteksi (*clasiffier*) nya menggunakan *Cosine Similarity* untuk mendapatkan nilai ambang batas galat. Pada perancangan sistem ini terbagi menjadi dua tahapan yaitu tahap pelatihan dan tahap pengujian.

1) Bagan *template*

Bagan *template* dari rancangan sistem ditunjukkan oleh Gambar 3.2.

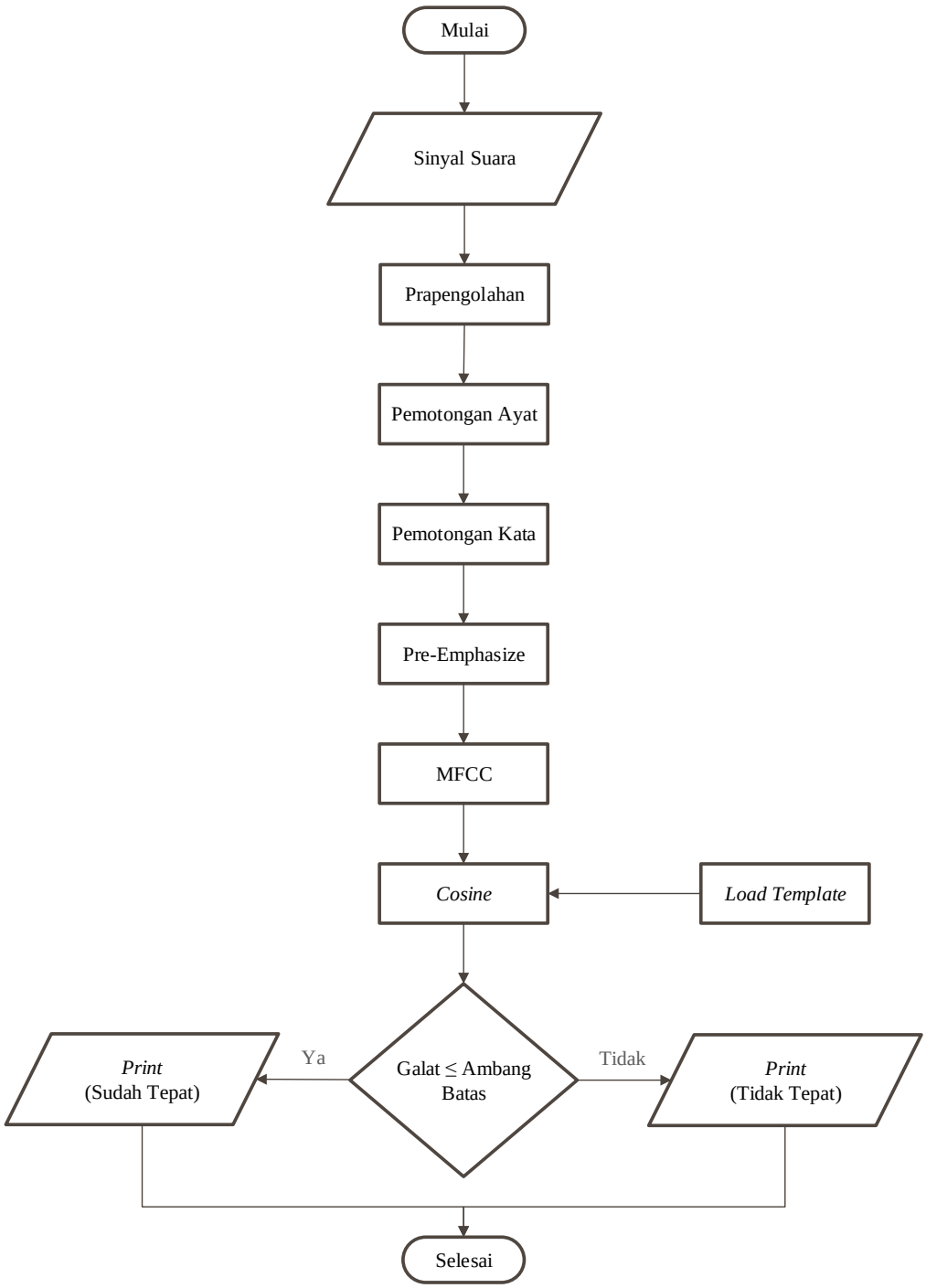


Gambar 3.2 Bagan *Template*

Pada tahap *template* data rekaman hanya menggunakan 1 buah data rekaman. Dimana data rekaman tersebut menggunakan penutur yang telah fasih membaca Al-Qur'an sebagai acuan dari pembacaan surah Al-Kautsar yang benar.

2) Bagan pelatihan dan pengujian

Bagan pelatihan dan pengujian pada rancangan sistem ditunjukkan oleh Gambar 3.3.

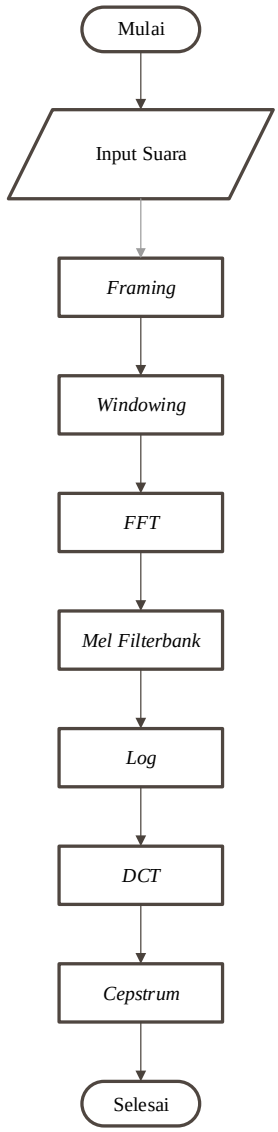


Gambar 3.3 Bagan Pelatihan dan Pengujian

Pada tahap pelatihan digunakan untuk mendapatkan nilai parameter yang menghasilkan hasil yang baik. Data rekaman yang digunakan berjumlah 30 rekaman penutur yang terdiri dari (15 bacaan benar dan 15 bacaan salah) baik laki-laki maupun perempuan. Data rekaman diambil secara acak dari 60 data rekaman penutur. Pada tahap pengujian digunakan untuk melihat kinerja dari sistem yang telah dirancang. Data rekaman yang digunakan pada tahap pengujian berjumlah sama dengan data pada tahap pelatihan, namun data rekaman penutur yang digunakan adalah data yang tidak digunakan sebagai data pada tahap pelatihan.

3) Bagan Ekstraksi Ciri MFCC

Ekstrasi ciri (*feature extraction*) pada penelitian ini menggunakan metode MFCC dengan bagan alir yang ditunjukkan oleh Gambar 3.4.



Gambar 3.4 Bagan Ekstraksi Ciri MFCC

3.3.1 Data Masukan

Data penutur menggunakan rekaman data dengan format file *.wav*. Sinyal wicara yang direkam menggunakan frekuensi cuplik 44.100 Hz dengan lama perekaman 25 – 35 detik. Untuk dapat mengolah data dari sinyal wicara terlebih dahulu dirubah kedalam bentuk diskrit dengan menggunakan perintah '*audioread'* sehingga mendapatkan nilai-nilai pada setiap titik sinyal wicara. Adapun kode program untuk memilih data penutur dan mengubah sinyal wicara kedalam bentuk diskrit adalah sebagai berikut:

```
[fname, pname] = uigetfile('*.wav', 'Pilih satu file audio');
%% Baca file suara
fprintf('Memproses file: %s\n', fname);
au_name = fullfile(pname, fname);
[suara, fs] = audioread(au_name);
suara = suara';
```

Variabel 'suara' adalah nilai-nilai dari sinyal wicara yang telah dirubah kedalam bentuk diskrit, sedangkan 'fs' adalah variabel yang menunjukan frekuensi cuplik dari sinyal wicara. 'au_name' merupakan lokasi (*path*) data penutur berada.

3.3.2 Pra-Pengolahan

Prapengolahan merupakan tahap pertama dalam proses pengolahan sinyal wicara. Pada tahap ini dilakukan pengolahan file sinyal wicara dari hasil rekaman. Di prapengolahan proses yang dilakukan pertama adalah proses *centering*. *Centering* digunakan untuk membuang nilai bias (*offset*) agar *baseline* sinyal berada pada sumbu-x *origin*. Dengan kata lain untuk membentuk sinyal *zero-mean.* Data masukan diolah sehingga didapatkan rata-rata (*mean*) vektor sinyal. Vektor sinyal tersebut yang kemudian diolah ke tahapan selanjutnya. Proses dari *centering* dikodekan sebagai berikut:

```
%% centering
suara1 = suara - mean(suara);
fprintf(' >> Centering ... \n');
```

Variabel ‘suara2’ merupakan hasil keluaran dari proses *centering*. Proses kedua adalah normalisasi, normalisasi digunakan untuk menskalakan nilai amplitudo tiap data sinyal wicara supaya sesuai dengan skala yang diinginkan. Secara umum penskalaan dilakukan dengan rentang nilai dari -1 hingga 1. Untuk melakukan normalisasi sinyal wicara yang telah melalui proses *centering* akan dibagi dengan nilai maksimal *absolute* dari sinyal wicara tersebut. Proses normalisasi dikodekan sebagai berikut:

```
%% Normalisasi amplitudo sinyal suara
suara2 = suara1 / max(abs(suara1));
fprintf(' >> Normalisasi ... \n');
```

Proses ketiga adalah EPD, pada proses ini digunakan untuk menemukan secara otomatis titik awal dan akhir dari sinyal wicara saat membaca surah Al-Kautsar. Variabel ‘suara2’ atau sinyal wicara yang telah di proses *centering* dan normalisasi akan masuk ke proses EPD. Sinyal wicara dicari nilai magnitudo nya dengan metode STFT dengan menggunakan fungsi *spectrogram* yang dimana dilakukan *framing*. *Framing* dilakukan secara *overlapping* sebesar 50% atau 0,5 dengan lebar *frame* yang didapatkan dari ‘ts’ bernilai 0,01 dikali ‘fs’. Variabel ‘ts’ merupakan nilai waktu cuplik (*time sampling*). Pengkodean dari program di atas dapat dikodekan sebagai berikut:

```
% fungsi untuk end point detection
function suara_EPD = EPD(suara, fs, ts)
% hitung panjang data tercuplik/frame
pjg = ts * fs;  % lebar framing
% terapkan stft (spectrogram)
[s, ~, ~, ~] = spectrogram(suara,hamming(pjg),floor(0.5*pjg),pjg);
```

Nilai magnitudo yang telah didapatkan kemudian diambil nilai *absolute* nya. Setelah itu mencari jumlah kolom dan baris dari data nilai magnitudo. Pengkodean program dapat dilakukan sebagai berikut:

```
% gunakan nilai absolute nya
abs_s = abs(s);
% cari jumlah kolom dan baris dari data nilai magnitude
```

```
num_r = size(abs_s, 1);
num_c = size(abs_s, 2);
```

Variabel ‘abs_s’ adalah nilai magnitudo yang telah diabsolutkan, sedangkan ‘num_r’ dan ‘num_c’ adalah jumlah baris dan kolom dari nilai magnitudo.Nilai magnitudo kemudian dikenakan nilai ambang batas dengan nilai 0,05. Nilai magnitudo yang memiliki nilai lebih besar dari nilai ambang batas akan dilabelkan menjadi ‘1’ sedangkan nilai magnitudo yang memiliki nilai di bawah nilai ambang batas dilabelkan menjadi ‘0’. Nilai magnitudo yang dilabelkan menjadi ‘1’ dan ‘0’ pada tiap *frame* akan dicek terkait nilai *energy* dan zero *crossing* nya untuk mendapatkan nilai ‘1’ diawal sinyal wicara bacaan dan ‘1’ diakhir yang menandakan akhir dari sinyal wicara bacaan. Kode program untuk mencari nilai di atas dapat menggunakan kode program berikut:

```
a = 1;
b = 1;
temp = zeros(size(s));
% cek tiap frame terkait energy dan zero crossingnya
for idx = 1 : num_c
   [nr, ~] = find(abs_s(:, idx) >thEPD);
   if numel(nr) > thEPD * num_r
      temp(:, idx) = 1;
   end
   for ida = 1 : num_r
      if (temp(ida, idx) == 1)
         lokasi(a) = b;
         a = a+1;
      end
      b = b + 1;
   end
end
% dapatkan sinyal yang terpotong
suara_EPD = suara(lokasi(1) : lokasi(end));
```

Variabel ‘thEPD’ adalah nilai ambang batas, ‘lokasi’ adalah indeks dari nilai ‘1’ yang ditemukan dari nilai magnitudo yang sudah dikenakan nilai ambang batas. Seteleh nilai ‘1’ diawal dan diakhir ditemukan, selanjutnya potong sinyal wicara dengan nilai indeks dimana nilai ‘1’ diawal dan diakhir ditemukan. Variabel ‘suara_EPD’ adalah variabel baru untuk hasil dari proses EPD, dimana ‘suara_EPD’ memiliki sinyal wicara saat mulai dan berakhirnya pembacaan surah Al-Kautsar.

3.3.3 Pemotongan Ayat

Pemotongan ayat dilakukan dengan menggunakan metode *envelope* terhadap sinyal wicara pembacaan surah Al-Kautsar yang sudah diatur temponya. Tempo antar ayat dibuat sekitar 3 sampai 4 ketukan, diasumsikan memenuhi kondisi tersebut. Untuk mencari nilai *envelope* dari sinyal suara_EPD dapat menggunakan kode program sebagai berikut:

```
% Hitung envelope sinyal
env = imdilate(abs(suara_EPD), true(1, 1501));
```

Didalam proses *envelope* dilakukan perhitungan nilai maksimum *absolute* dengan fungsi *imdilate* menggunakan vektor yang berukuran $1 \times N$ dengan bernilai *true* dan bernilai 1 sebanyak ukuran vektor. Dimana nilai suara_EPD akan dikalikan dengan nilai vektor tersebut sehingga menghasilkan *matrix* baru yang berisi nilai *envelope* dengan nama variabel ‘env’. Nilai *envelope* yang sudah didapatkan akan dikenakan nilai ambang batas 0,05, nilai *envelope* yang berada di atas nilai ambang batas akan dilabelkan sebagai ‘1’ dan nilai yang ada di bawah akan dilabelkan sebagai ‘0’. Pengkodean nilai ambang batas sebagai berikut:

```
% Bagian thresholding
ayat = env<thayat;
```

Variabel ‘thayat’ adalah nilai dari ambang batas untuk ayat. Variabel ‘Ayat’ menampung nilai-nilai *envelope* yang memiliki nilai di atas ambang batas dari ‘suara_EPD’. Selanjutnya mencari indeks dari nilai ‘0’ yang saling berdekatan. Untuk mencari nilai ‘0’ yang saling berdekatan kode program yang dapat digunakan adalah sebagai berikut:

```
% Dapatkan index dari 0 data Ayat
[~, ca] = find(ayat==0);
% Cari jumlah nilai 0 dari nilai x...xn
range = unique(ca(1,:));
t_n = [];
n_hilang = [];
inc = 1;
for i = 1:numel(range)-1
   temp1 = ca(i);
   temp2 = ca(i+1);
   if temp1 + 1 ~= temp2
      s = temp1+1 : temp2-1;
      n_hilang{inc} = s(:);
      t_n(inc) = numel(s);
      inc = inc + 1;
   end
end
```

Variabel ‘n_hilang’ mengandung rentang nilai titik sinyal wicara yang dianggap sebagai ‘0’ dan ‘t_n’ adalah indeks jumlah nilai ‘0’ terbanyak yang saling berdekatan. Setelah ditemukan indeks nilai ‘0’ yang saling berdekatan, kemudian dicari posisi 2 indeks yang memiliki nilai ‘0’ terbanyak. Untuk mencari 2 indeks nilai ‘0’ terbanyak dapat dilakukan dengan kode program berikut:

```
%% Ambil 2 index dari data 0 terbanyak
[~, rc] = maxk(t_n,2);
```

Variabel ‘rc’ merupakan variabel yang menunjukan 2 indeks nilai ‘0’ yang memiliki jumlah terbanyak. 2 nilai ‘0’ yang terbanyak kemudian diurutkan sesuai indeks urutan pencarian jumlah ‘0’ terbanyak. Untuk mengurutkan 2 nilai ‘0’ terbanyak yang sudah ditemukan agar sesuai posisi indeks dapat dilakukan dengan kode program sebagai berikut:

```
rc = sort(rc,'ascend');
```

Setelah 2 indeks nilai ‘0’ terbanyak diurutkan sesuai dengan posisi indeks urutan pencarian jumlah ‘0’ terbanyak, kemudian dapatkan rentang nilai titik sinyal wicara pada variabel ‘n_hilang’ yang berada pada 2 indeks nilai ‘0’ terbanyak yang sudah ditemukan. Pengkodean program dapat dilakukan sebagai berikut:

```
m1 = cell2mat(n_hilang(rc(1)))';
m2 = cell2mat(n_hilang(rc(2)))';
```

Variabel ‘m1’ menunjukan rentang nilai titik sinyal wicara yang dianggap sebagai ‘0’, ‘m1’ merupakan jeda antara ayat pertama dan ayat kedua. Variabel ‘m2’ menunjukan rentang nilai titik sinyal wicara yang dianggap sebagai jeda antara ayat kedua dan ayat ketiga. Nilai rentang titik sinyal wicara jeda antar setiap ayat yang sudah didapatkan kemudian dikenakan ke sinyal suara_EPD untuk didapatkan sinyal wicara pada setiap ayat bacaan surah Al-Kautsar. Kode program yang digunakan untuk mendapatkan sinyal wicara setiap ayat adalah sebagai berikut:

```
%% Ambil data suara setiap Ayat
% Ayat 1
Ayat1 = suara_EPD(1 : m1(1));
% Ayat 2
Ayat2 = suara_EPD(m1(end) : m2(1));
% Ayat 3
Ayat3 = suara_EPD(m2(end) : end);
```

3.3.4 Pemotongan Kata

Pemotongan kata dilakukan dengan menggunakan metode *envelope* terhadap sinyal wicara bacaan surah Al-Kautsar yang sudah dipotong menjadi 3 ayat. Pada ayat pertama dan kedua memiliki kata sebanyak 3 kata sedangkan pada ayat ketiga memiliki kata sebanyak 4 kata. Tempo antar kata yang dibuat sekitar 1 sampai 2 ketukan diasumsikan memenuhi kondisi tersebut. Mencari nilai *envelope*, melakukan ambang batas galat, mencari indeks ‘0’ yang saling berdekatan menggunakan kode program yang sama dengan proses pemotongan ayat. Selanjutnya dicari posisi indeks yang memiliki nilai ‘0’ terbanyak, pada ayat pertama dan kedua dicari 2 posisi indeks sedangkan pada ayat ketiga dicari 3 posisi indeks yang memiliki nilai ‘0’ terbanyak.

Posisi indeks yang bernilai ‘0’ terbanyak pada setiap ayat dianggap jeda antar kata yang digunakan untuk melakukan pemisahan setiap kata pada setiap ayat. Pengkodean dari proses di atas dapat dilakukan dengan kode program berikut:

```
%% Ambil index dari data 0 terbanyak pada ayat 1,2 dan 3
  if m == 1 || m == 2
    % Cari range nilai titik 0 dari index terbanyak Ayat ke 1 & 2
    [~, rc] = maxk(t_n,2);
    rc = sort(rc,'ascend');
    % Kata 1
    m1 = cell2mat(n_hilang(rc(1)))';
    % Kata 2
    m2 = cell2mat(n_hilang(rc(2)))';
    % Ambil data suara setiap Ayat ke 1 & 2
    % Kata 1
    Kata1 = ayat(1 : m1(1));
    % Kata 2
    Kata2 = ayat(m1(end) : m2(1));
    % Kata 3
    Kata3 = ayat(m2(end) : end);
  elseif m == 3
    [~, rc] = maxk(t_n,3);
    rc = sort(rc,'ascend');
    % Kata 1
    m1 = cell2mat(n_hilang(rc(1)))';
    % Kata 2
    m2 = cell2mat(n_hilang(rc(2)))';
    % Kata 3
     m3 = cell2mat(n_hilang(rc(3)))';
    % Ambil data suara setiap Ayat ke 3
    % Kata 1
    Kata1 = ayat(1 : m1(1));
```

```
    % Kata 2
    Kata2 = ayat(m1(end) : m2(1));
    % Kata 3
    Kata3 = ayat(m2(end) : m3(1));
    % Kata 4
    Kata4 = [];
    Kata4 = ayat(m3(end) : end);
end
```

Variabel ‘rc’ merupakan variabel yang menunjukan indeks nilai ‘0’ yang memiliki jumlah terbanyak pada setiap ayat. Nilai ‘0’ yang terbanyak kemudian diurutkan sesuai indeks urutan pencarian jumlah ‘0’ terbanyak. Variabel ‘m1’, ‘m2’ dan ‘m3’ menunjukan rentang nilai titik sinyal wicara yang dianggap sebagai ‘0’, ‘m1’ dan ‘m2’ merupakan jeda antara kata pertama, kedua dan ketiga pada ayat pertama dan kedua sedangkan ‘m3’ merupakan jeda antara kata pertama, kedua, ketiga dan keempat pada ayat ketiga. Variabel ‘Kata1’, ‘Kata2’, ‘Kata3’ dan ‘Kata4’ adalah sinyal wicara kata pada surah Al-Kautsar.

3.3.5 *Pre-Emphasize*

Setiap kata pada surah Al-Kautsar selanjutnya dilakukan proses pemfilteran sinyal dengan menggunakan filter *pre-emphasize*. Tujuan dikenakanya filter *pre-emphasize* adalah untuk mendapatkan bentuk *spectral* frekuensi sinyal wicara yang lebih halus. Nilai α (koefisien penekanan) yang digunakan adalah $0{,}9375$. Kode program untuk melakukan pemfilteran *pre-emphasize* adalah sebagai berikut:

```
Kata_pre{n,m} = filter([1 -.9375], 1, input{n,m});
```

Variabel ‘Kata_pre’ merupakan sinyal wicara kata yang telah dikenakan filter *pre-emphasize*. Sedangkan ‘input’ adalah variabel untuk data masukan yaitu Kata.

3.3.6 Ekstraksi Ciri

Setiap kata yang sudah melalui proses *pre-emphasize* kemudian akan dimasukan ke proses selanjutnya yaitu MFCC. Keluaran MFCC adalah sebuah vektor, pada setiap kata akan digunakan vektor 1×13. MFCC akan digunakan untuk

mendapatkan fitur koefisien MFCC dari setiap kata pada surah Al-Kautsar. Proses awal dari ekstraksi ciri MFCC adalah melakukan *framing*, dimana lebar *frame* yang digunakan adalah 256. Pada setiap *frame* dilakukan *filtering* dengan menggunakan fungsi *windowing* yang lebarnya sama dengan *frame*. Jenis *window* yang digunakan adalah *hamming window*, digunakanya *hamming window* adalah untuk menghasilkan *sidelobe* level yang tidak terlalu tinggi (kurang lebih -43 dB), selain itu derau yang dihasilkanpun tidak terlalu besar.

Proses selanjutnya adalah mencari nilai FFT pada setiap *window* yang telah dibuat, FFT digunakan untuk menganalisis frekuensi sehingga dapat mempermudah pemrosesan sinyal wicara. Nilai FFT yang didapatkan kemudian dilakukan *Mel-frequency Wrapping* dengan *filterbank* diterapkan dalam kawasan frekuensi. *Filterbank* menggunakan representasi konvolusi dalam melakukan filter terhadap sinyal. Konvolusi dilakukan dengan melakukan multiplikasi antara *spectrum* sinyal dengan koefisien *filterbank*. Dalam *Mel-frequency wrapping* didapatkan vektor *Mel-filterbank* 1×13, dimana vektor *Mel-filterbank* yang digunakan hanya dari nilai vektor 2 sampai 13 sehingga menjadi vektor 1×12. Hal ini dikarenakan nilai vektor pertama memiliki nilai magnitudo yang besar sehingga dianggap sebagai komponen DC. Proses selanjutnya adalah mengkonversikan nilai-nilai FFT menjadi satu nilai. Mengurangi nilai *Mel-filterbank* dengan mengganti setiap nilai *log* dasarnya untuk mengambil nilai logaritma *Mel-filtered* (*log Mel*) pada *segment* tutur. Proses terakhir spektrum *log Mel* dikonversi menjadi spektrum menggunakan DCT yang merupakan nilai dari hasil *Mel-frequency* yang diubah menjadi kawasan waktu. Hasil dari ekstraksi ciri MFCC digunakan untuk mendapatkan informasi dari suatu sinyal wicara yang diucapkan oleh manusia yang disebut sebagai koefisien MFCC (*Cepstrum*).

3.3.7 *Cosine Similarity*

Pada proses *cosine similarity* akan dilakukan perbandingan setiap kata dari data latih maupun data uji dengan data *template*. Cara pendeteksianya adalah dengan membandingkan antara koefisien MFCC di kata pertama pada data *template* dengan koefisien MFCC kata pertama pada data latih maupun data uji dan seterusnya. Kode program untuk melakukan pengukuran *cosine similarity* adalah sebagai berikut:

```
jarak = 'cosine'; % Metode Perhitungan Jarak
```

```
F = [ceps'; fmccs(k, :)];
ds = pdist(F,  jarak);
DIST(k) = ds;
```

Variabel ‘jarak’ adalah metode pengukuran jarak yang digunakan. Variabel ‘F’ berisi nilai *cepstrum* dari tiap kata data *template* dengan data latih maupun data uji. Variabel ‘DIST’ berisi nilai jarak antara *cepstrum* dari kata pertama data *template* dengan *cepstrum* dari kata pertama data latih maupun data uji dan seterusnya yang didapatkan dari proses perhitungan pada variabel ‘ds’. Nilai dari ‘DIST’ kemudian dilakukan perhitungan galat. Galat tersebut akan dibandingkan dengan nilai ambang batas. Apabila galat lebih kecil dari nilai ambang batas maka bacaan akan dianggap sudah tepat, akan tetapi jika galat lebih besar dari nilai ambang batas maka bacaan akan dianggap tidak tepat. Program untuk melakukan perhitungan galat dikodekan sebagai berikut:

```
therror = THRES(itr); % Nilai Ambang Batas
if DIST(h) <= therror
    fprintf(' --> Ayat %s Kata %s Pembacaannya SUDAH TEPAT\n',
    seq_ayat{n}, seq_kata);
    result(h, 1) = 1;
else
    fprintf(' --> Ayat %s Kata %s Pembacaannya TIDAK TEPAT\n',
    seq_ayat{n}, seq_kata);
end
res1 = [res1, result'];
```

Variabel ‘therror’ merupakan nilai ambang batas yang bervariasi dari nilai 0,01 sampai 1 dengan kelipatan 0,01. Variabel ‘result’ dan ‘res1’ merupakan nilai logikal ‘1’ dan ‘0’ yang menandakan benar atau salahnya kata pada setiap bacaan, dimana nilai ‘1’ adalah kata benar dan nilai ‘0’ adalah kata salah.

3.3.8 Pengukuran Kualitas Unjuk Kerja

Unjuk kerja pada penelitian ini dilakukan dengan menggunakan *recall* dan *precision*. *Recall* digunakan untuk mengukur ketepatan seluruh bacaan maka hanya

menggunakan variabel ‘TPB’ dan ‘FNB’. Kode program untuk mendapatkan nilai dari variabel ‘TPB’ dan ‘FNB’ adalah sebagai berikut:

```
if sum(allResult(m - 2, :)) / size(allResult,2) == 1
    TPB = TPB + 1;
    fprintf('<-> Bacaan SUDAH BENAR\n');
else
    FNB = FNB + 1;
    fprintf('<-> Bacaan BELUM BENAR\n');
end
```

Variabel ‘TPB’ merupakan nilai TP untuk setiap bacaan, ‘FNB’ merupakan nilai FN untuk setiap bacaan sedangkan ‘allResult’ adalah nilai logika ‘1’ dan ‘0’. Nilai TP bertambah saat jumlah kata yang benar dibagi dengan total banyaknya kata pada bacaan menghasilkan nilai 1, apabila tidak sama dengan 1 maka nilai FN bertambah. Untuk mencari persentase *recall* dari bacaan menggunakan perhitungan (2.16), perhitungan untuk mencari persentase *recall* bacaan dapat dikodekan sebagai berikut:

```
recallB = TPB / (TPB + FNB);
acctpb = recallB*100;
```

Dimana variabel ‘acctpb’ adalah nilai persentase *recall* bacaan yang didapatkan. Untuk pengujian terhadap ketepatan setiap katanya menggunakan *recall* dan *precision* yang melibatkan variabel ‘TPK’, ‘TNK’, ‘FPK’ dan ‘FNK’. Pengkodean untuk mencari nilai dari variabel ‘TPK’, ‘TNK’, ‘FPK’ dan ‘FNK’ adalah sebagai berikut:

```
for iy = 1 : length(res1)
    if res1(iy) == MAPS(m - 2, iy)
        if res1(iy) == 1
            TPK = TPK + 1;
        else
            TNK = TNK + 1;
        end
    else
```

```
            if res1(iy) == 1
                FPK = FPK + 1;
            else
                FNK = FNK + 1;
            end
        end
    end
```

Dimana ‘TPK’ adalah TP pada kata, ‘TNK’ adalah TN pada kata, ‘FPK’ adalah FP pada kata dan ‘FNK’ adalah TN pada kata. Nilai *recall* kata melibatkan nilai dari TP pada kata dan FN pada kata. Dimana TP pada kata adalah kata yang benar terdeteksi sudah tepat sedangkan FN pada kata adalah kata yang benar terdeteksi tidak tepat. Nilai *precision* kata melibatkan nilai dari TP pada kata dan FP pada kata, dimana FP pada kata adalah kata salah terdeteksi sudah tepat. Untuk mendapatkan nilai TP, FN dan FP pada kata dilakukan secara *hard decision*, dimana setiap nilai logikal kata ‘1’ dan ‘0’ yang didapatkan dari proses *thresholding* akan dibandingkan dengan *mapper*. *Mapper* berisi nilai logikal ‘1’ dan ‘0’, dimana nilai dalam *mapper* menandakan posisi kata benar dan kata salah pada setiap bacaan.

Untuk mencari persentase *recall* dari kata menggunakan perhitungan (2.16), sedangkan untuk mencari nilai *precision* dari kata menggunakan perhitungan (2.17). Perhitungan untuk mencari persentase *recall* dan *precision* kata dapat dikodekan sebagai berikut:

```
% Recall Kata
recallK = TPK / (TPK + FNK);
acctpk = recallK * 100;
fprintf(' > Recall Kata : %.2f %%\n', acctpk);
% Prercision Kata
precissionK = TPK / (TPK + FPK);
precis = precissionK*100;
fprintf(' > Precision Kata : %.2f %%\n', precis);
```

Variabel ‘acctpk’ adalah nilai persentase *recall* kata yang didapatkan dan ‘precis’ adalah nilai persentase *precision* kata yang didapatkan. Hasil persentase *recall*

bacaan, kata dan *precision* kata yang didapatkan bergantung pada proses perhitungan galat, dimana galat tersebut dibandingkan dengan nilai ambang batas. Nilai ambang batas dilakukan dengan cara pengujian dari 0,01 hingga 1 dengan kelipatan 0,01. Dimana hasil dari persentase *recall* bacaan yang didapatkan nantinya dicari nilai ambang batas mana yang menghasilkan persentase *recall* bacaan yang terbaik dengan melihat persentase *recall* kata dan persentase *precision* kata.