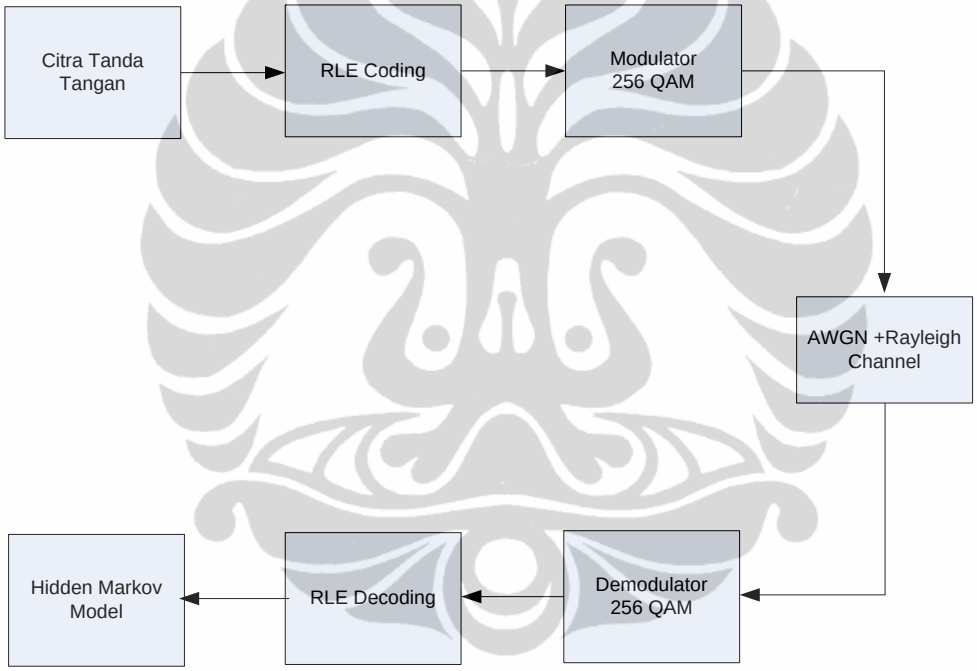
BAB 3
PERANCANGAN SISTEM

Sebelum citra tanda tangan dikenali dengan menggunakan Hidden Markov Model (HMM) citra tanda tangan tersebut ditransmisikan dengan dikompresi menggunakan Run Length Encoding (RLE) dan melewati kanal *fading Rayleigh*. Adapun sistem yang digunakan dapat dilihat pada Gambar 3.1 :



Gambar 3.1 Sistem pentransmisian dan pengenenalan citra tandatangan

Input citra tanda tangan yang digunakan yaitu sebanyak 5 buah yang masing-masing akan ditransmisikan sebanyak 10 dan 20 kali. Dari hasil transmisi ini akan didapatkan citra tanda tangan sebanyak 50 dan 100 buah, diharapkan citra tanda tangan ini sudah mewakili citra yang terkena derau (*noise*) akibat proses pentransmisian. Citra yang diterima ini nantinya akan dijadikan sebagai data *training* citra pada basis data (*database*).

Adapun penjelasan tiap blok pentransmisiannya adalah sebagai berikut :

1. *Input* citra tanda tangan

Misalkan dimasukkan file citra tanda tangan ’setyahadi. bmp’ maka akan dibaca sebagai file RGB dapat dilihat pada Gambar 3.2 sebagai berikut:

RGB <178x287 uint8>

	1	2	3	4	5	6	7
1	255	255	255	255	255	255	255
2	255	255	255	255	255	255	255
3	255	255	255	255	255	255	255
4	255	255	255	255	255	255	255
5	255	255	255	255	255	255	255
6	255	255	255	255	255	255	255
7	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255
9	255	255	255	255	255	255	255
10	255	255	255	255	255	255	255
11	255	255	255	255	255	255	255
12	255	255	255	255	255	255	255
13	255	255	255	255	255	255	255
14	255	255	255	255	255	255	255
15	255	255	255	255	255	255	255

Gambar 3.2 Nilai-nilai *pixel* dari citra tanda tangan ‘Setyahadi.bmp’

Akan didapatkan nilai *pixel* dari citra dari 0 sampai 255.

2. RLE *Coding*

Setelah didapatkan nilai dari file citra tersebut kemudian dilakukan kompresi terhadap nilai-nilai *pixel* tersebut dan didapatkan hasil kompresinya dapat dilihat pada Gambar 3.3 sebagai berikut :

data <1x4513 uint8>

	1112	1113	1114	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124
1	0	73	255	246	255	7	0	73	255	246	255	7	0

Gambar 3.3 Nilai-nilai *pixel* ‘Setyahadi.bmp’ yang telah dikompresi dengan RLE

3. Modulasi 256 QAM

Setelah dikompresi proses selanjutnya dimodulasi agar nilai-nilai *pixel* tersebut berubah menjadi frekuensi domain berupa nilai real dan imaginer sehingga mudah ditransmisikan pada saat memasuki kanal. Adapaun nilai-nilainya dapat dilihat pada Gambar 3.4 sebagai berikut :

data_mod1 <1x4513 double>

	1	2	3	4	5	6
1	15.0000 - 13.0000i	15.0000 + 3.0000i	15.0000 - 13.0000i	15.0000 + 3.0000i	15.0000 - 13.0000i	15.0000 + 3.0000i

Gambar 3.4 Nilai-nilai *pixel* ‘Setyahadi.bmp’ yang telah dimodulasi dengan modulasi 256 QAM

4. Kanal AWGN dan *Fading Rayleigh*

Pada kanal ini dapat dilihat pengaruh *noise* pada nilai citra yang telah dimasukkan yaitu nilai *pixel* nya berubah-ubah karena pengaruh nilai noise yang berubah-ubah. Dapat dibandingkan jika hanya ditransmisikan pada kanal AWGN nilai *pixel* nya tidak terlalu jauh berbeda sedangkan pada kanal *fading Rayleigh* nilai *pixel* nya berubah jauh hal ini disebabkan adanya faktor pengali 'h' (*noise*) pada persamaan 3.1. Adapun nilai-nilainya dapat dilihat pada Gambar 3.5 sebagai berikut :

data_Rayleigh <1x4513 double>

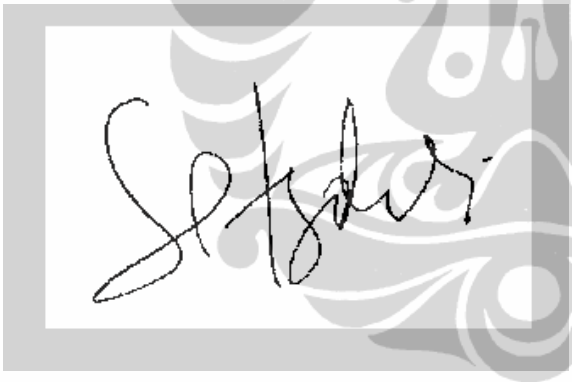
	1	2	3	4	5	6
1	19.9706 + 3.5153i	8.4685 - 12.3052i	-4.1076 + 26.5241i	0.4538 - 6.0766i	-10.0064 - 4.8134i	-9.2417 - 0.9311i

data_AWGN <1x4513 double>

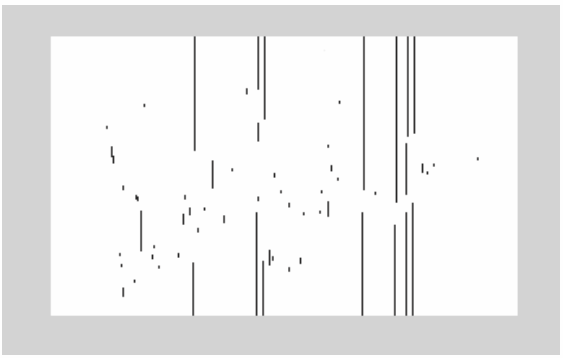
	1	2	3	4	5	6
1	15.0589 - 12.9347i	15.0075 + 2.9757i	14.9782 - 12.9859i	15.0146 + 2.9868i	14.9105 - 13.0830i	15.0557 + 3.0350i

Gambar 3.5 Nilai-nilai *pixel* 'Setyahadi.bmp' yang telah ditransmisikan pada kanal AWGN dan fading Rayleigh

Perbedaan nilai-nilai pixel pada kanal AWGN dan *fading Rayleigh* berpengaruh pada tampilan citra tanda tangan seperti dapat dilihat pada Gambar 3.6.



(A)



(B)

Gambar 3.6 Perbedaan tampilan citra tanda tangan yang ditransmisikan pada kanal AWGN(A) dan *fading Rayleigh* (B)

5. Demodulasi 256 QAM

Setelah melewati kanal selanjutnya didemodulasi lagi didapatkan nilai *pixel* dalam *time domain*. Adapun nilai-nilainya dapat dilihat pada gambar 3.7.

data_demod1 <1x4513 double>

	1	2	3	4	5	6	7	8	9	10
1	246	206	80	139	42	56	255	4	242	175

Gambar 3.7 Nilai-nilai *pixel* 'Setyahadi.bmp' yang telah didemodulasi dengan 256 QAM

6. RLE *Decoding*

Pada proses ini nilai-nilai *pixel* yang telah terkompresi akan dikembalikan ke nilai awalnya. Didapatkan nilai-nilainya pada gambar 3.8 sebagai berikut :

data_terima1 <178x287 double>

	1	2	3	4	5	6	7	8	9	10
1	246	246	246	246	246	246	246	246	246	246
2	246	246	246	246	246	246	246	246	246	246
3	246	246	246	246	246	246	246	246	246	246
4	246	246	246	246	246	246	246	246	246	246
5	246	246	246	246	246	246	246	246	246	246
6	246	246	246	246	246	246	246	246	246	246
7	246	246	246	246	246	246	246	246	246	246
8	246	246	246	246	246	246	246	246	246	246
9	246	246	246	246	246	246	246	246	246	246
10	246	246	246	246	246	246	246	246	246	246
11	246	246	246	246	246	246	246	246	246	246
12	246	246	246	246	246	246	246	246	246	246
13	246	246	246	246	246	246	246	246	246	246
14	246	246	246	246	246	246	246	246	246	246
15	246	246	246	246	246	246	246	246	246	246

Gambar 3.8 Nilai-nilai *pixel* 'Setyahadi.bmp' yang telah didecoding dengan RLE

7. Identifikasi *Hidden Markov Model*

Proses identifikasi citra tanda tangan dengan *Hidden Markov Model* akan dijalaskan pada sub bab 3.2

3.1 Proses Pentransmisian

Pada proses pentransmisian melewati kanal yang diasumsikan *flat fading Rayleigh* [10],

$$y = hx + n \qquad \text{.....(3.1)}$$

dimana,

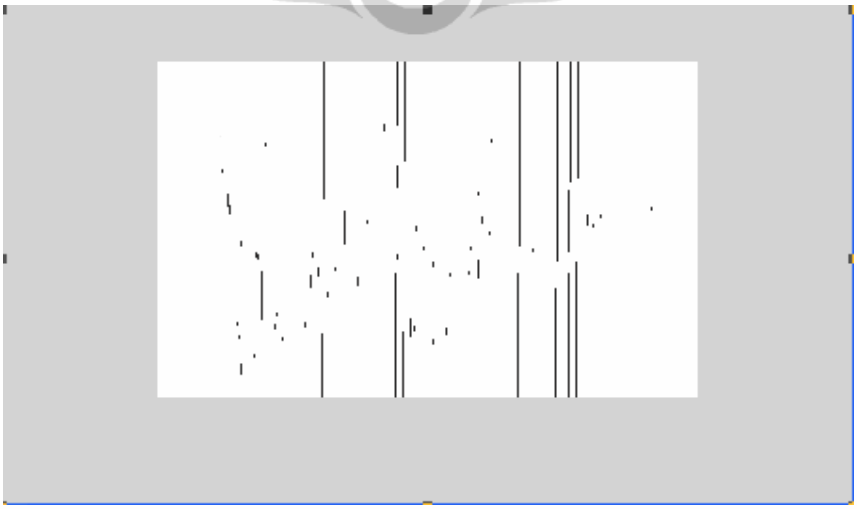
y adalah simbol yang diterima,

h dimodelkan sebagai variabel komplek Gaussian dengan *mean* ’0’ dan *variance* σ^2,
x adalah simbol yang ditransmisikan,
n adalah *Additive White Gaussian Noise* (AWGN)

Dimana algoritma pentransmisiannya adalah sebagai berikut :

```
  masukkan nama_tanda_tangan = nama_tanda_tangan;
% tahap encoding
  [run data] = rle(nama_tanda_tangan);
% tahap modulasi 256 QAM
  data_mod = qammod(data,256);
% kanal Rayleigh
  data_terima=data_mod*h + 10^(-Eb_No_dB/20)*n;
  demodulasi = qamdemod(data_terima,256);
  decoding = rle_decoding(demodulasi);
```

Hasil pentransmisian citra tanda tangan jika menggunakan kompresi RLE dapat dilihat pada Gambar 3.9. Sedangkan jika tidak menggunakan kompresi RLE dapat dilihat pada Gambar 3.10.



Gambar 3.9 Tampilan citra tanda tangan yang ditransmisikan dengan menggunakan

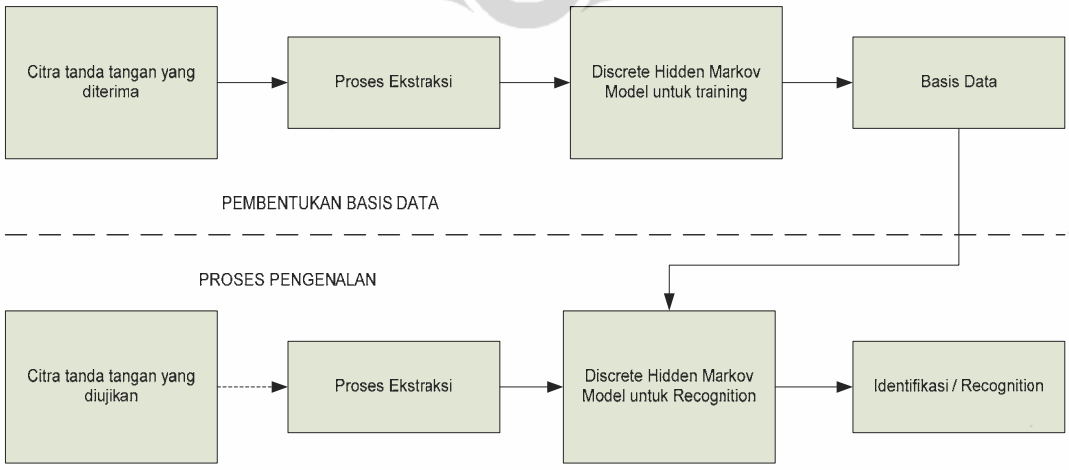
kompresi RLE

Gambar 3.10 Tampilan citra tanda tangan yang ditransmisikan dengan tidak menggunakan kompresi RLE

3.2 Proses Identifikasi

Proses identifikasi citra tanda tangan ini secara garis besar dibagi menjadi dua tahap utama, yaitu proses pembentukan *database* dan proses pengenalan. Citra tanda tangan akan dianalisa dengan menggunakan metode *Hidden Markov Model* (HMM).

Hubungan antara proses pembentukan *database* dengan proses pengenalan ditunjukkan oleh Gambar 3.3 di bawah ini :



Gambar 3.11 Hubungan antara proses pembentukan *database* dan pengenalan

3.3 Pembentukan Basis Data

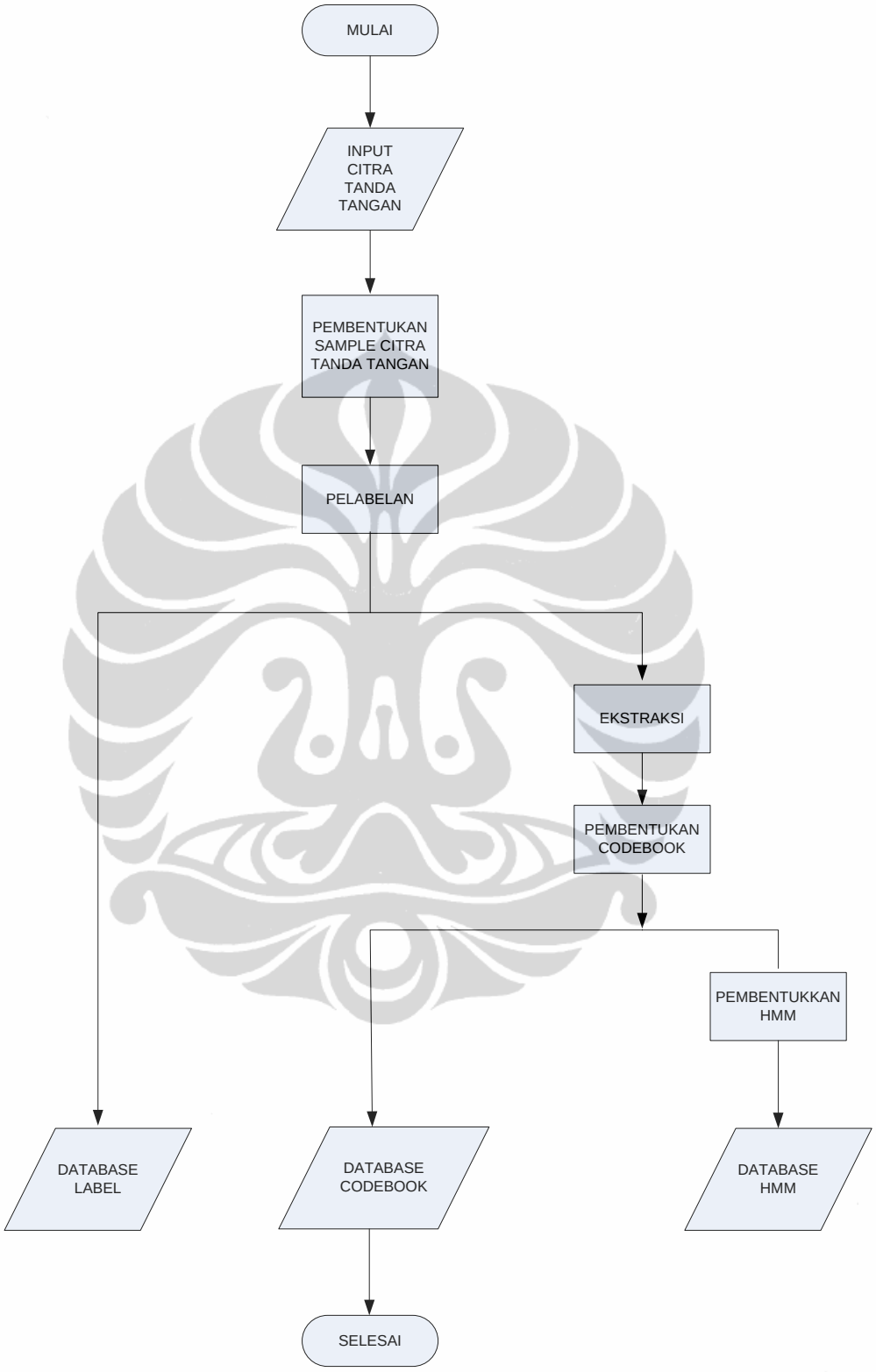
Sebelum dilakukan proses pengenalan, dilakukan proses pembentukan basis data untuk memperoleh data sebagai acuan. Terdapat 3 tahapan proses utama dalam pembentukan *database*, yaitu:

(1) Proses pelabelan

(2) Proses pembuatan *Codebook*

(3) Proses pembentukan HMM

Diagram alir pembuatan *database* dapat dilihat pada Gambar 3.12. Citra tanda tangan yang dimasukkan dibentuk menjadi *sample* citra tanda tangan dengan melibatkan proses pengolahan citra. Proses pengolahannya terdiri dari *cropping*, dan *reshapping*. Hasil *sample* citra yang telah diolah tersebut yang selanjutnya akan dipakai pada proses-proses utama dalam pembentukkan *database*.



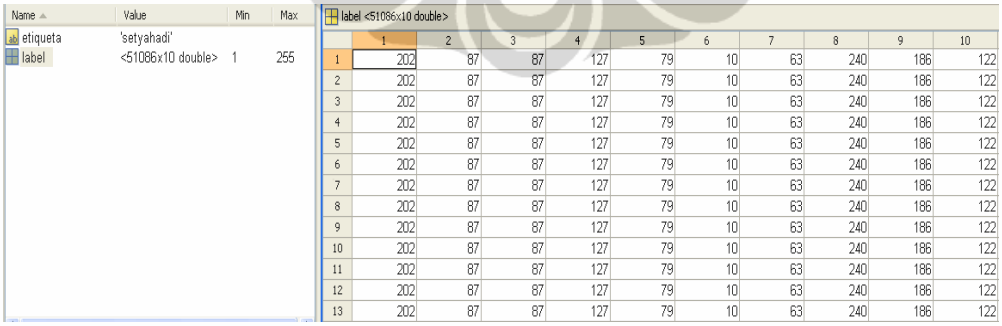
Gambar 3.12 Diagram alir pembentukan *database*

3.3.1 Proses Pelabelan

Pada proses pelabelan ini, masing-masing citra tanda tangan yang akan didaftarkan pada *database* diberi label sesuai dengan nama tanda tangan tersebut. Sebagai contoh, tanda tangan ”Setyahadi” diberi label1, tanda tangan ”Siska” diberi label 2 dan seterusnya. Nama Label inilah yang nantinya akan menjadi keluaran pada proses pengenalan tanda tangan.

Pada program pelabelan terdapat tiga masukkan, yakni urutan jumlah label (*index*), jumlah data yang dimasukkan ke tiap label (*training*), dan nama label. Jumlah *training* diisi dengan angka yang diinginkan dan nama label diisi sesuai dengan nama tanda tangan yang dimasukkan dalam *data base*. Proses pelabelan ini dilakukan dengan menekan tombol *start* pada tampilan program proses pelabelan. Tombol *start* ini kemudian akan memanggil fungsi pelabelan.

Untuk melakukan proses pelabelan dibutuhkan *file-file* gambar dari masing-masing tanda tangan dalam format ”`*.bmp`”. Pada proses labelisasi, citra dari tiap tanda tangan diubah ke bentuk matriks N x M (ukuran matriks N x M akan sama dengan ukuran citra yang dimasukkan). Keluaran dari pelabelan ini adalah kumpulan matriks-matriks kolom dari tiap tanda tangan dengan jumlah kolom sebanyak jumlah *training.* Dapat dilihat pada Gambar 3.13 dibawah ini bahwa ukuran matrik dari label ”Setyahadi” adalah 51086 x 10 menunjukkan besarnya data sebesar 51086 hasil *reshape* MN x 1 dengan jumlah training 10 buah.



Name	Value	Min	Max
etiqueta	'setyahadi'		
label	<51086x10 double>	1	255

label <51086x10 double>

	1	2	3	4	5	6	7	8	9	10
1	202	87	87	127	79	10	63	240	186	122
2	202	87	87	127	79	10	63	240	186	122
3	202	87	87	127	79	10	63	240	186	122
4	202	87	87	127	79	10	63	240	186	122
5	202	87	87	127	79	10	63	240	186	122
6	202	87	87	127	79	10	63	240	186	122
7	202	87	87	127	79	10	63	240	186	122
8	202	87	87	127	79	10	63	240	186	122
9	202	87	87	127	79	10	63	240	186	122
10	202	87	87	127	79	10	63	240	186	122
11	202	87	87	127	79	10	63	240	186	122
12	202	87	87	127	79	10	63	240	186	122
13	202	87	87	127	79	10	63	240	186	122

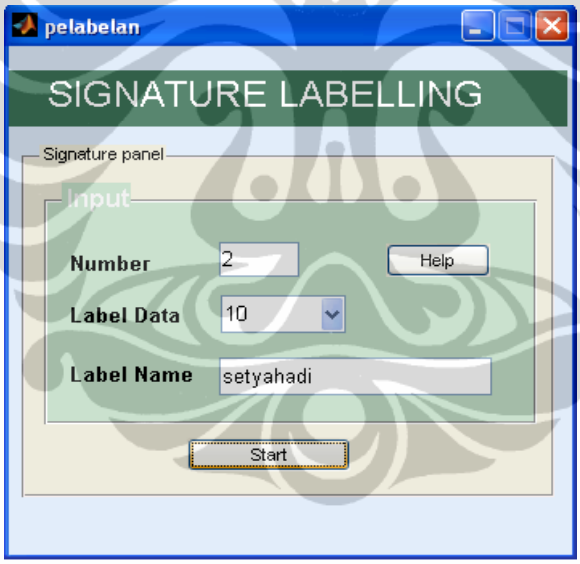
Gambar 3.13 Tampilan matrik untuk label ”Setyahadi”

Dalam proses pelabelan ini akan dibuat 5 buah label yang terdiri dari label 1 sampai dengan 5.

Algoritma dari proses pelabelan adalah :

```
Mulai
 Untuk I=1 sampai N
 Input nama tanda tangan
 Tulis nama tanda tangan
 Tentukan jumlah data dari tiap label = M
 Ubah nama tanda tangan menjadi bentuk matrik
 kolom
 Nama label tanda tangan[i]=nama tanda tangan
 Kembali
Selesai
```

Tampilan program proses pelabelan dapat dilihat pada Gambar 3.14 dibawah ini



Gambar 3.14 Tampilan program pelabelan

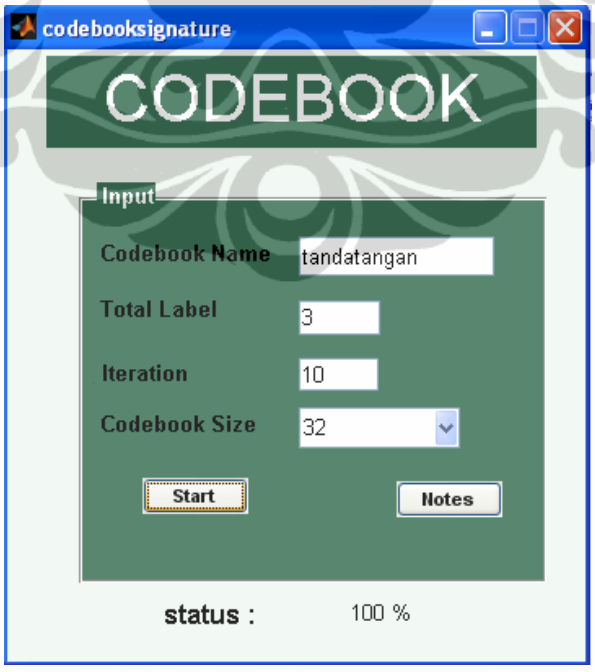
3.3.2 Proses Pembuatan Codebook

Setelah dilakukan proses pelabelan, maka langkah selanjutnya adalah melakukan proses penggabungan dari semua label yang telah dibuat ke dalam sebuah *file codebook*. Pada pembuatan *codebook* ini juga dilakukan proses *Vector Quantization* yang telah dijelaskan pada Bab 2.

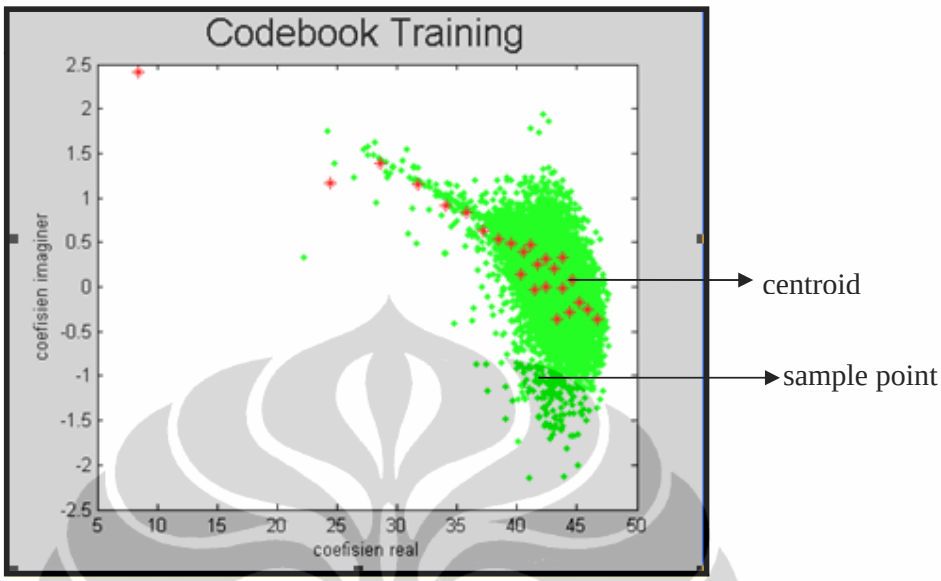
Pada program ini terdapat empat buah *input*, yakni nama *file codebook*, jumlah label, jumlah iterasi, dan ukuran *codebook*. Nama *file codebook* dimasukkan sesuai

dengan nama *file* yang diinginkan yang nantinya akan tersimpan dalam format "`*.mat`" di dalam folder yang sama dengan *codebook.m*. Misalnya *codebook* ini diberi nama file "`tandatangan`", maka hasil dari proses pembuatan *codebook* ini akan tersimpan dengan nama *file* "`tandatangan.mat`".

Ukuran *codebook* yang tersedia dalam program ini adalah 32, 62, dan 128 bit. Dimana keempat ukuran *codebook* ini akan dijadikan bahan perbandingan untuk melihat berapa nilai *codebook* yang paling akurat pada proses pengenalan citra tanda tangan. Jumlah iterasi merupakan banyaknya proses pengulangan yang dilakukan dalam menentukan *centroid* guna mendapatkan *centroid* yang cukup presisi. Semakin besar jumlah iterasinya, maka akan semakin presisi letak *centroid* yang didapat, namun dengan mengambil iterasi yang sangat tinggi proses pembuatan *codebook* akan berjalan sangat lambat. Dalam tesis ini ditentukan *default* untuk besarnya iterasi adalah 10 [12] dengan harapan letak *centroid* yang diperoleh cukup presisi dan waktu proses relatif cepat. Berturut-turut Gambar 3.15 dan Gambar 3.16 dibawah ini adalah tampilan pembuatan *codebook* untuk ukuran 32 dan *vector quantization training map* ukuran 32.



Gambar 3.15 Tampilan pembuatan codebook ukuran 32



Gambar 3.16 Hasil tampilan *VQ training map* ukuran 32

Berdasarkan Gambar 3.16 terdapat dua buah titik, dimana titik yang berwarna hijau menunjukkan interpretasi data hasil proses segementasi (*sample points*) data dan titik yang berwarna merah merupakan data hasil VQ atau *centroid*, yang akan mewakili dari sekian banyak jumlah data. Semakin besar ukuran *codebook* nya, maka semakin banyak jumlah *centroid* dan semakin berhimpitan letaknya. Letak centroid ini akan menentukan tingkat kepresisian dalam identifikasi.

Algoritma dari proses pebuatan *codebook* adalah sebagai berikut:

```
Mulai
Tentukan besar nilai N
Untuk I = 1 sampai N
      Load label [I];
Kembali;
Gabungkan label;
Hitung FFT untuk setiap sample[i];
Sample point[i] = nilai FFT;
Tentukan cluster;
```

Untuk j =1 sampai cluster

Hitung centroid dengan LBG;

Simpan centroid[j] berdasarkan urutan labelnya;

Kembali

Selesai

3.3.3 Proses Pembuatan Parameter HMM

Proses pembentukan HMM ini dilakukan dengan menggunakan program *hmm.m*. Pada program ini terdapat tiga masukan, yakni *file* hmm, *file codebook*, dan jumlah iterasi. *File* HMM dimasukkan sesuai dengan keinginan dimana hasil dari proses parameter HMM akan disimpan dengan nama *file* tersebut dalam format *"*.mat"* di dalam folder yang dengan sama *hmm.m* . Misalnya, akan dimasukkan *file* hmm dengan nama "ttdhmm", maka hasil dari proses pembentukan hmm ini akan disimpan dalam file "ttdhmm.mat". *File* codebook dimasukkan sesuai dengan nama file *codebook* yang telah diisi sebelumnya pada proses pembuatan *codebook.*

Sama halnya pada proses pembuatan *codebook*, dalam pembentukan hmm juga ditentukan *default* untuk besarnya iterasi sebesar 10 dengan harapan probabilitas yang didapat bernilai tinggi dan waktu proses relatif cepat.

Algoritma dari proses pembentukan HMM yang bertujuan untuk memperoleh nilai parameter dari setiap label berdasarkan *file codebook* yang telah dibuat adalah sebagai berikut :

Mulai

Ambil *file* *.mat dari *codebook* yang telah dibuat;

Ekstraksi file label citra tanda tangan;

Mencari nilai matrik observasi dari tiap label;

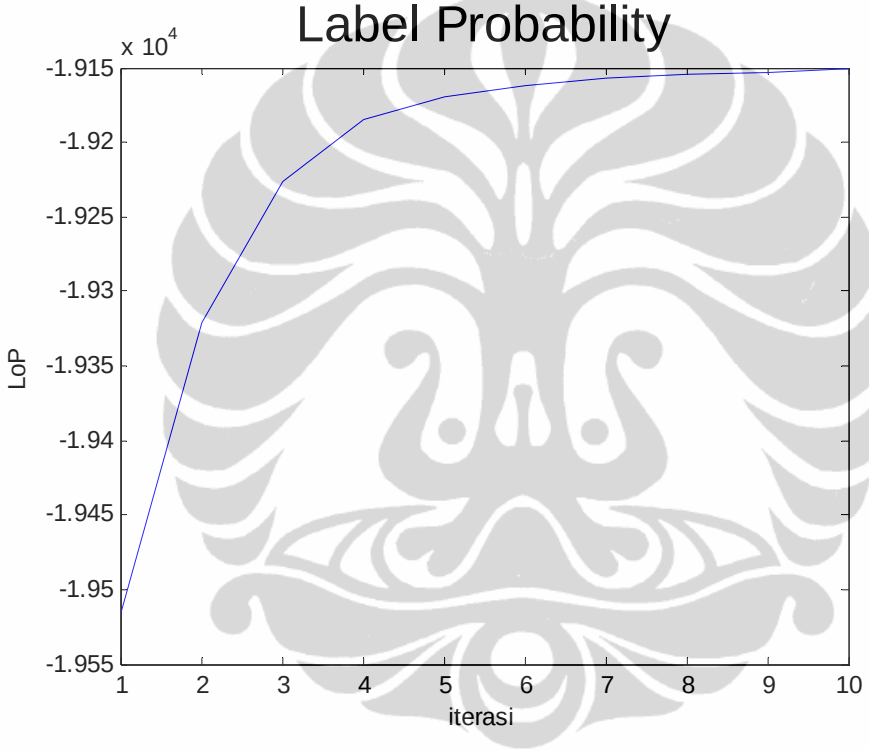
Inisialisasi matriks A, B dan π dengan nilai acak;

Pelatihan menggunakan algoritma *baum-welch* dengan memasukkan semua data *image* dihitung nilai A, B, π sampai nilai matriks tidak berubah;

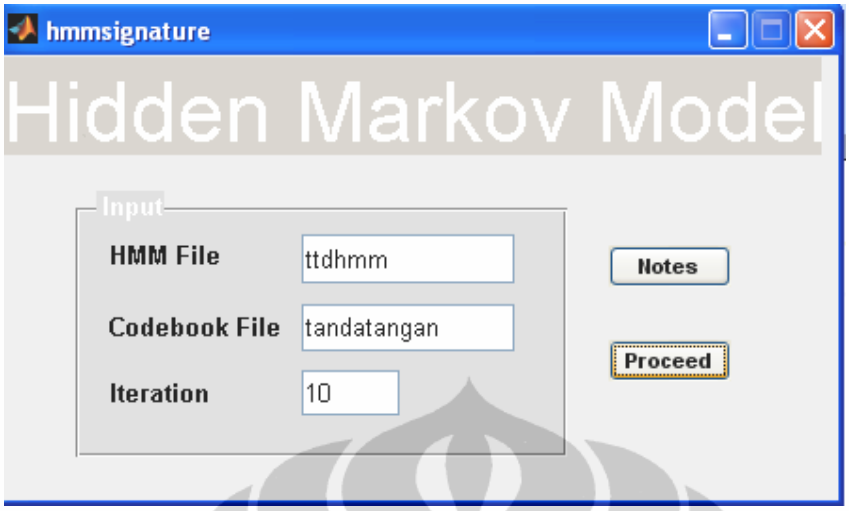
Hitung probabilitas observasi HMM untuk setiap

```
image;
Simpan hasil dalam file format *.mat;
Selesai
```

Gambar 3.17 dibawah ini adalah tampilan grafik probabilitas dalam pembuatan model HMM :



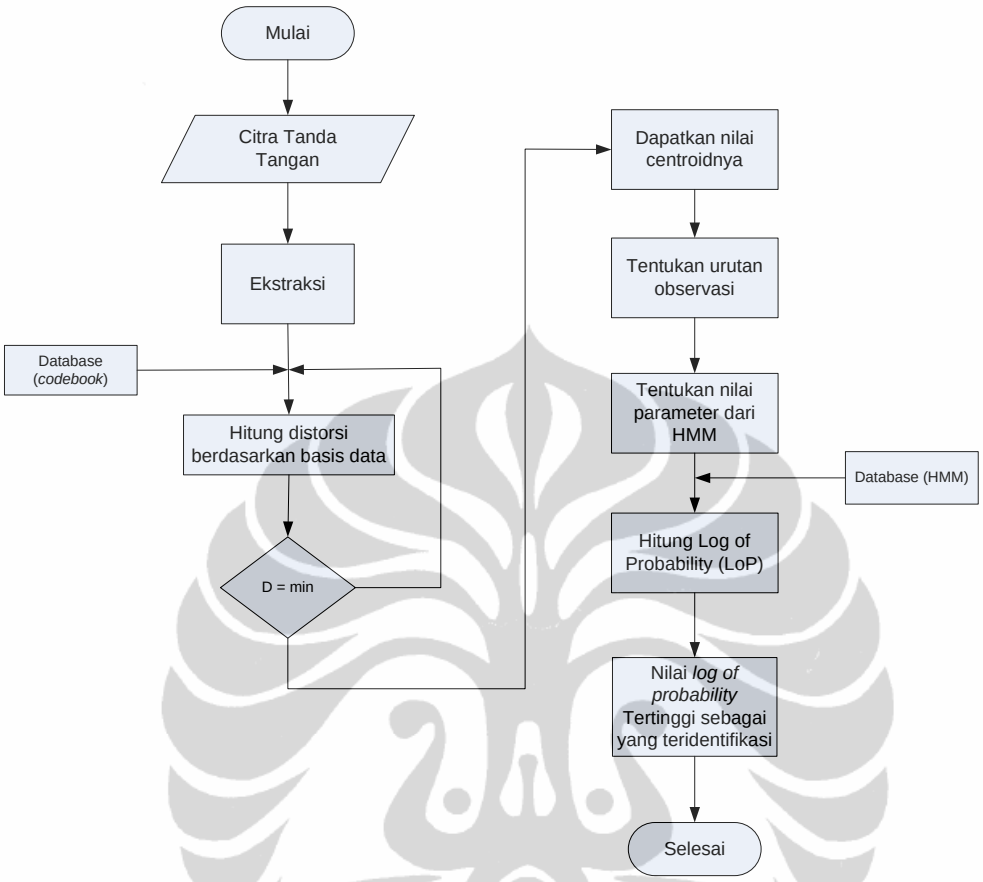
Gambar 3.17. Probabilitas *Log of Probability* 5 label tanda tangan dengan 10 *training*



Gambar 3.18. Tampilan pembentukan parameter HMM

3.4 Proses Pengenalan Citra Tanda Tangan

Setelah basis data dibuat, maka proses pengenalan pada tanda tangan bisa dilakukan. Terlebih dahulu di *input* kan file *codebook*, parameter HMM, citra tanda tangan yang akan diuji, kemudian ditekan tombol *proceed* agar citra tanda tangan yang diujikan mengalami proses pentransmisian. Gambar 3.19 adalah diagram alir dari proses pengenalan.



Gambar 3.19. Diagram alir proses pengenalan

Pada diagram alir pengenalan Gambar 3.19 diatas dapat dilihat prosesnya serupa dengan proses pembentukkan basis data hanya pada proses pengenalan tidak dilakukan proses pembelajaran (training). Pada proses pengenalan, masukan berupa citra tanda tangan yang akan diuji diekstraksi terlebih dahulu kemudian diubah ke dalam domain frekuensi dengan menggunakan FFT. Hasilnya dalam domain frekuensi membentuk titik-titik sample yang bernilai vektor *rea*l dan *imaginer* yang akan dipetakan dalam *codebook* dalam bentuk *sample points* dan dicari nilai *centroid*nya dalam basis data. Berdasarkan nilai *centroid* tersebut dihitung besar *log of probability* (LoP) untuk semua nilai parameter HMM yang diperoleh. Demikian pula dengan nilai parameter HMM dari semua jenis tanda tangan yang terdapat pada basis data dihitung LoP-nya. Dari nilai LoP untuk semua tanda tangan dalam basis data dan LoP tanda tangan yang diuji dicari nilai yang terbesar. Nilai LoP yang terbesar adalah nilai LoP

yang dicari. Dengan demikian maka didapatkan hasil identifikasi berupa nama label dari *file* yang dikenali sesuai nilai LoP tersebut.

Pada proses pengenalan titik *sampel* dari citra tanda tangan yang akan diuji dicari nilai *codeword*nya pada basis data yang kemudian diperoleh jenis citra tanda tangan yang sesuai yang terdapat pada basis data. Dari data citra tanda tangan tersebut ditentukan parameter HMM nya dan selanjutnya dihitung nilai LoP seperti contoh di bawah ini :

tanda tangan 1 $\longrightarrow$ (w1, w2, w2, w1, w1) = $a_{12}*b_1*a_{22}*b_2*a_{21}*b_2*a_{11}*b_1$

tanda tangan 2 $\longrightarrow$ (w1, w2, w1, w3, w1) = $a_{12}*b_1*a_{21}*b_2*a_{13}*b_1*a_{31}*b_3$

tanda tangan x $\longrightarrow$ (w4, w5, w4, w5, w4) = $a_{45}*b_4*a_{54}*b_5*a_{45}*b_4*a_{54}*b_5$

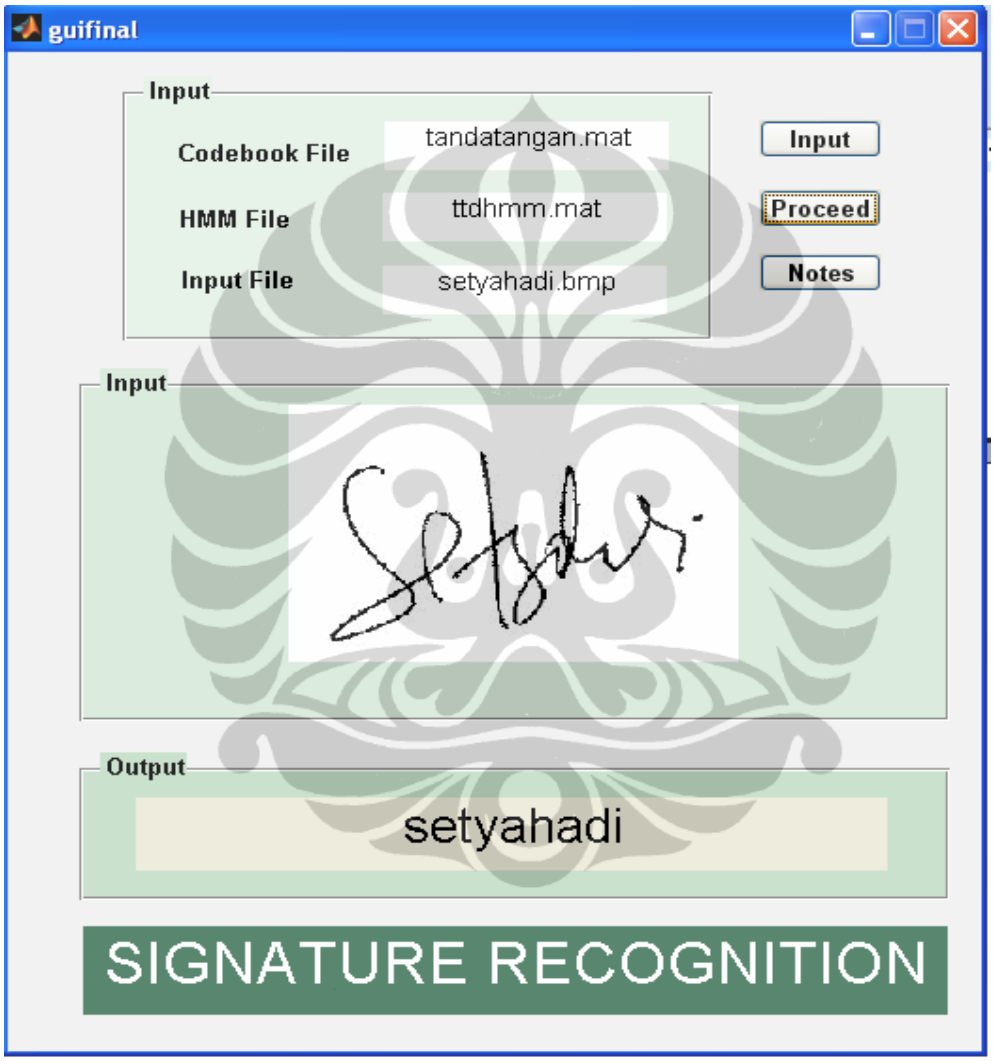
dimana w1, w2, w2 dan seterusnya menyatakan keadaaan (*state*) yang diambil berdasarkan jumlah *centroid* yang ada dalam basis data, adapun nilai a_{ij} pada contoh a_{12} peluang transaksi (*probability of transaction*) dan nilai b_i adalah peluang munculnya state ke i. Proses ini diulang untuk semua jenis citra tanda tangan dan selanjutnya dicari nilai LoP yang paling besar.

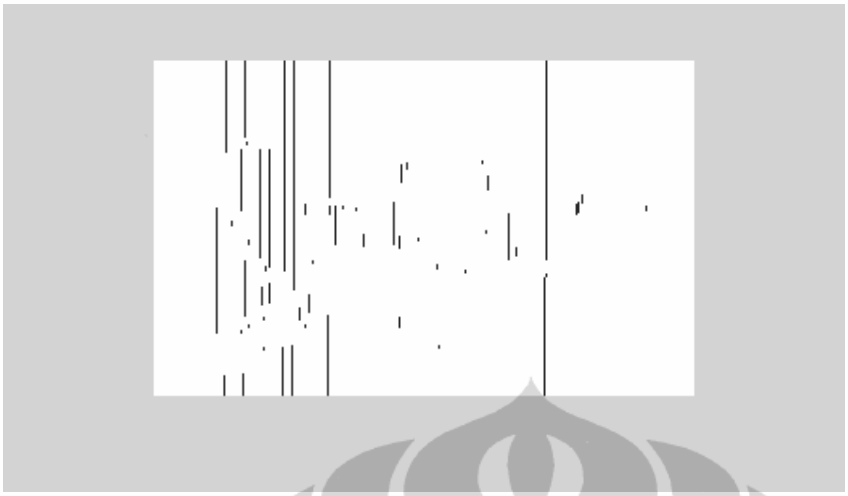
Algoritma dari proses pengenalan dengan HMM dapat ditulis sebagai berikut:

```
Mulai
imread tanda_tangan.bmp;
Hitung FFT;
Karakter tanda_tangan = matriks codeword tanda_tangan;
Cari centroid codeword_tanda_tangan di basis data;
Tentukan nilai observasi;
Tentukan parameter HMM tanda_tangan;
untuk h = 1 sampai  jml_label
    Baca data parameter HMM untuk jenis karakter tanda      tangan
    lainnya dari basis data;
   Hitung log of probability (LoP) untuk semua label
   tanda_tangan;
   LoP[jml_label] = LoP;
   Cari LoP[jumlah_label] = tertinggi;
   Tentukan nama label untuk LoP tertinggi;
```

```
Kembali
Selesai
```

Gambar 3.20 adalah tampilan dari pengenalan tanda tangan yang telah ditransmisikan dengan menggunakan kompresi RLE melalui kanal *fading Rayleigh.*





Gambar 3.20 Tampilan pengenalan tanda tangan yang telah ditransmisikan dengan kompresi RLE melalui kanal *fading Rayleigh*