**Bab III**

**Perancangan Pengujian**

## 3.1 Perancangan Awal Sistem dan Eksperimen Pengenalan Huruf Arab

Dalam tesis ini akan dibuat sebuah perangkat lunak pengenalan huruf Arab yang bisa mengenali huruf Arab tunggal dalam rangkaian kata. Masukan dari sistem ialah berupa citra berisikan kalimat tulisan Arab dan memberikan luaran berupa hasil pengenalan huruf Arab dari citra kalimat tersebut.

Gambaran umum tahapan proses pengenalan huruf Arab dari awal hingga akhir yang akan diimplementasikan dapat dilihat pada Gambar III.1



Gambar III.1 Alur proses pengenalan huruf Arab

Dokumen yang menjadi masukan perangkat lunak adalah citra tulisan Arab cetak*.* Pada penelitian ini citra tulisan Arab cetak diperoleh dengan membuat kalimat – kalimat tulisan Arab kemudian dilakukan *screenshoot* dari tulisan Arab tersebut. Pertama – tama citra dokumen ini akan masuk ke tahap praolah citra agar citra lebih berguna dan pengolahan citra pada tahap selanjutnya lebih mudah dilakukan. Hal yang dilakukan pada tahap praolah dapat dibaca pada subbab III.2.

Selanjutnya pada tahap segmentasi, hasil proses tahap praolah citra dipotong per upakata sehingga diperoleh citra baru sejumlah upakata yang ada. Hasil citra per baris ini akan melalui tahap praolah citra lagi untuk proses pencatatan objek sekunder. Kemudian hasilnya akan dikenai tahap segmentasi huruf. Detail proses pada tahap segmentasi dapat dibaca pada subbab III.3.

Tahap penipisan dilewati setelah citra upakata dipotong dan sebelum segmentasi huruf. Tahap penipisan berada diantara dua jenis tahap segmentasi. Hasil tahap penipisan adalah citra per upakata yang memiliki ketebalan hanya satu piksel. Kedua citra keluaran dan citra masukan (sebelum dikenai proses penipisan) tahap penipisan digunakan pada tahap ekstraksi fitur. Detail proses pada tahap penipisan dapat dibaca pada subbab III.4



Gambar III.2 Skema proses pengenalan huruf Arab

Tahap penipisan dilakukan setelah segmentasi upakata dan sebelum segmentasi huruf. Hasil tahap penipisan ini adalah citra per upakata (atau citra per baris) yang memiliki ketebalan 1 piksel. Kedua citra keluaran dan citra masukan  
(sebelum dikenai proses penipisan) tahap penipisan digunakan pada tahap  
ekstraksi fitur. Detail proses pada tahap penipisan dapat dibaca pada subbab III.4.

Tahap ekstraksi fitur akan diambil beberapa fitur yang akan mempenhgaruhi hasil  
pengenalan setiap huruf. Setiap citra per huruf yang telah dan yang  
belum ditipiskan melewati tahap ekstraksi fitur. Hasil dari tahap ini adalah nilai dari setiap fitur huruf yang diekstrak. Daftar fitur yang diekstrak dan metode ekstraksinya dapat dibaca pada subbab III.5.

Tahap klasifikasi adalah tahap yang menentukan citra (per huruf hasil segmentasi)  
itu huruf Arab yang mana. Masukan tahap ini adalah hasil ekstraksi fitur dari  
masing-masing citra per huruf. Detail proses tahap ini dapat dibaca pada  
subbab III.6.

Setelah setiap citra huruf pada citra dokumen dikenali, hasil pengenalan huruf  
dijadikan menjadi satu berkas teks. Berkas teks ini adalah keluaran dari perangkat  
lunak.

Skema lengkap keseluruhan proses di atas dapat diamati pada Gambar III.2.

**III.2 Tahap Praolah Citra**

Sebelum diproses lebih lanjut citra akan melewati beberapa tahap praolah yakni sebagai berikut.

**III.2.1 Binerisasi**

Binerisasi pada citra adalah proses merubah citra kedalam bentuk biner (0 dan 1). Gambar yang pada mulanya berwarna akan dirubah menjadi hitam dan putih. Citra biner adalah citra yang setiap pikselnya hanya memiliki 2 kemungkinan derajat yakni 0 dan 1. Konversi citra berwarna ke biner ini dilakukan pada saat awal citra dokumen halaman dibuka oleh perangkat lunak.

Cara yang dilakukan adalah dengan melakukan pengambangan (*threshold*) pada  
masing-masing kanal warna. Ambang yang digunakan adalah 127. Jika satu piksel  
pada kanal warna merah memiliki nilai kurang dari 127, piksel kanal itu dianggap  
hitam. Jika ada dua dari tiga kanal warna yang hitam, piksel tersebut diubah  
menjadi warna hitam. Pengecekan dua dari tiga kanal warna ini bertujuan untuk  
mengurangi derau.

**III.3 Segmentasi**

Terdapat dua segmentasi yang digunakan dalam membangun perangkat lunak ini, yakni segmentasi upakata dan segmentasi huruf.

**III.3.1 Segmentasi Upakata**

Segmentasi upakata berfungsi untuk memisahkan huruf-huruf bersambung pada  
tulisan Arab. Metode yang digunakan dalam segmentasi ini adalah metode yang  
dijabarkan oleh Amin (2000).

Tidak ada perubahan yang berarti antara metode Amin (2000) dan metode yang  
diterapkan pada perangkat lunak yang dibangun. Piksel bersambung mendatar  
dianggap menempel dengan sangkar piksel komponen piksel terhubung jika satu  
dari piksel pada piksel bersambung mendatar di sekitar sangkar komponen  
terhubung. Dengan kata lain, penerapan algoritma ini adalah dengan piksel  
bersambung delapan arah.  
Metode sederhana di atas juga tidak hanya melewati badan utama tulisan tetapi  
juga menjaring objek sekunder tulisan. Dengan demikian, segmentasi dengan  
metode sederhana tersebut juga dimanfaatkan untuk menemukan objek sekunder  
tulisan Arab seperti titik.

Setelah didapatkan kumpulan komponen piksel terhubung, masing-masing piksel  
terhubung dikelompokkan ke dalam kategori “badan utama”, “objek sekunder”,  
dan “derau” menurut ukurannya.

Piksel terhubung dianggap badan utama jika berukuran lebih dari 400 piksel atau  
sangkar piksel tersebut dilewati oleh lini basis. Piksel terhubung dengan ukuran di  
antara 40 piksel hingga 400 piksel dianggap sebagai objek sekunder selama tidak  
dilewati oleh lini basis. Sisanya dianggap sebagai derau dan diabaikan

**III.3.2 Segmentasi Huruf**

Segmentasi huruf berfungsi untuk memisahkan huruf – huruf yang berada dalam upakata. Pada penelitian ini, segmentasi huruf di implementasikan dengan algoritma Zidouri (2010). Masukan dari segmentasi huruf yang digunakan adalah citra hasil segmentasi upakata yang telah dihilangkan objek sekundernya.

**3.3 Segmentasi**

Terdapat tiga segmentasi berbeda yang digunakan dalam perangkat lunak yaitu  
segmentasi baris, segmentasi upakata, dan segmentasi huruf.

**3.3.2 Segmentasi Upa Kata**

Segmentasi upakata berfungsi untuk memisahkan huruf-huruf bersambung pada  
tulisan Arab. Metode yang digunakan dalam segmentasi ini adalah metode yang  
dijabarkan oleh Amin (2000).

Tidak ada perubahan yang berarti antara metode Amin (2000) dan metode yang  
diterapkan pada perangkat lunak yang dibangun. Piksel bersambung mendatar  
dianggap menempel dengan sangkar piksel komponen piksel terhubung jika satu  
dari piksel pada piksel bersambung mendatar di sekitar sangkar komponen  
terhubung. Dengan kata lain, penerapan algoritma ini adalah dengan piksel  
bersambung delapan arah.

Metode sederhana di atas juga tidak hanya melewati badan utama tulisan tetapi  
juga menjaring objek sekunder tulisan. Dengan demikian, segmentasi dengan  
metode sederhana tersebut juga dimanfaatkan untuk menemukan objek sekunder  
tulisan Arab seperti titik.

Setelah didapatkan kumpulan komponen piksel terhubung, masing-masing piksel  
terhubung dikelompokkan ke dalam kategori “badan utama”, “objek sekunder”,  
dan “derau” menurut ukurannya.

Piksel terhubung dianggap badan utama jika berukuran lebih dari 400 piksel atau  
sangkar piksel tersebut dilewati oleh lini basis. Piksel terhubung dengan ukuran di  
antara 40 piksel hingga 400 piksel dianggap sebagai objek sekunder selama tidak  
dilewati oleh lini basis. Sisanya dianggap sebagai derau dan diabaikan.

**3.3.3 Segmentasi Huruf**

Segmentasi huruf berfungsi untuk memisahkan huruf-huruf Arab yang menempel  
satu sama lain pada satu upakata. Masukan dari segmentasi huruf adalah citra  
hasil segmentasi upakata yang telah dihilangkan objek sekundernya. Dengan  
demikian, hanya badan utama upakara yang menjadi referensi pemotongan.  
Metode segmentasi yang digunakan adalah metode segmentasi yang dipaparkan  
oleh Zidouri (2010) dengan sedikit penyesuaian. Zidouri pada makalahnya  
merekomendasikan penggunaan citra kosong untuk menampung pita kandidat  
perpotongan huruf. Pada proses yang diimplementasikan di Tugas Akhir ini, pita  
kandidat hanya disimpan sebagai senarai saja.  
Ada empat aturan yang dipakai algoritma ini saat memilih pita kandidat, yaitu:  
1. Pilih pita jika lebar relatifnya paling besar dan *F4 = 1*

2. Pilih pita jika *F2* > *Ls* dan *F4 = 1*

3. Pilih pita jika *F2* < *Ls* dan *F3* > *Lsa* dan pita bukan pita terakhir  
4. Pilih pita jika *F1* lebih dari *Lm* dan *F4 = 1*

dengan *F1* = lebar pita, *F2* = jarak pita ke pita sebelah kanannya, *F3* = jarak pita  
ke pita kedua sebelah kanannya, dan *F4 =* posisi ditemukannya pita, bernilai satu  
di atas lini basis dan nol jika di bawah (Zidouri, 2010).

Pada (Zidouri, 2010) tidak dijelaskan bagaimana menghitung jarak pita pada fitur  
F2 dan F3. Pada implementasi perangkat lunak tugas akhir ini, jarak horizontal  
antar pita dihitung dari garis tengah masing-masing pita.

Selain empat aturan tersebut, terdapat pula aturan tambahan untuk pita kandidat  
pertama. Pita ini tetap dipilih meskipun gagal memenuhi keempat aturan di atas  
jika pita berikutnya memenuhi aturan nomor dua (Zidouri, 2010). Pada algoritma  
Zidouri ini sebenarnya tidak dijelaskan arti dari pita pertama dan pita terakhir.  
Algoritma ini diimplementasikan dengan asumsi bahwa pita pertama yang  
dimaksud pada aturan di atas adalah pita paling kanan dan pita terakhir adalah pita  
paling kiri.

Aturan di atas ditambah dengan pelonggaran keketatan aturan jika tidak ada satu  
pita pun yang memenuhi. Contohnya dengan menghilangkan pengecekan *F4*kecuali pada aturan nomor empat. Aturan tambahan ini *mungkin* diterapkan  
karena garis horizontal hasil penipisan lini basis bisa saja tidak tepat berada pada  
titik lini basis yang dihitung saat citra belum ditipiskan. Aturan tambahan ini tetap  
diterapkan pada implementasi perangkat lunak. Meskipun begitu, nilai F4 tetap  
dianggap satu meskipun posisi ditemukannya pita berada di bawah lini basis  
selama selisih kedua posisi (pita dan lini basis) tidak lebih dari tiga piksel. Nilai  
ini diambil dari fakta bahwa ketebalan lini basis pada citra baris setinggi 64 piksel  
kurang lebih adalah enam piksel.

Konstanta yang digunakan pada algoritma Zidouri ini dapat diamati pada  
Tabel III.1. Konstanta ini disesuaikan dengan tinggi baris yang dipakai yakni 64  
piksel.  
Tabel III.1 Konstanta yang digunakan pada implementasi algoritma Zidouri

|  |  |  |
| --- | --- | --- |
| **Konstanta** | **Nilai** | **Keterangan** |
| Ls | 6 | Lebar karakter terkecil |
| Lsa | 12 | Lebar dua karakter terkecil jika muncul bersamaan |
| Lm | 30 | Lebar maksimum karakter pada bentuk terisolasi |
| Theta | 6 | Lebar satu titik pada dokumen |

Sebelum segmentasi huruf dengan algoritma Zidouri ini dilakukan, dilakukan pula  
pengecekan apakah upakata yang sedang ditinjau sudah merupakan satu huruf.  
Hal ini dilakukan agar huruf tunggal tersebut tidak disegmen lagi. Hal ini  
39  
dilakukan dengan mengecek ukuran lebar upakata. Jika lebar upakata di bawah  
*Lm*, peluang upakata tersebut sudah merupakan satu huruf adalah besar. Dicek  
pula rasio aspek dari citra masukan. Bila rasio lebar per tinggi kurang dari satu,  
kemungkinan citra tersebut adalah citra huruf.

**3.4 Penipisan**

Penipisan adalah proses pengambilan tulang dari suatu pola, proses ini mengikis piksel sebanyak mungkin tanpa mempengaruhi bentuk umum. Setelah piksel di kikis, pola tersebut tetap harus dikenali. Kerangka yang diperoleh harus memiliki sifat sebagai berikut :

1. Setipis mungkin
2. Terhubung
3. Berpusat ditengah

Bila sifat ini terpenuhi, algoritma harus berhenti, berikut adalah pola dan hasil penipisan.





Gambar II.2 Pola huruf dan hasil penipisan

Penipisan berguna apabila kita tidak tertarik pada ukuran dari pola melainkan pada posisi relatif goresan dari pola. Ada beberapa algoritma yang dirancang untuk tujuan ini. Dalam penelitian ini penipisan dilakukan dengan algorima Stentiford.

Pada algoritma Stentiford ada 4 buah template yang dipakai , template 3 x 3 yaitu :



Gambar IV Template Stentiford

Langkah – langkah algoritma Stentiford :

1. Cari lokasi piksel (i,j) yang cocok dengan *template* T1. Dengan *template* T1 semua piksel di bagian atas dari citra akan dihapus. Pencocokkan *template* ini bergerak dari kiri ke kanan dan dari atas ke bawah.
2. Bila piksel tengah bukan merupakan *endpoint* dan mempunyai jumlah konektivitas 1, maka tandai *pixel* untuk kemudian dihapus.
3. Endpoint adalah piksel yang merupakan batas akhir dan hanya terhubung 1 piksel saja. Artinya, jika piksel hitam hanya memiliki satu tetangga hitam dari delapan kemungkinan tetangga.
4. Jumlah konektivitas adalah ukuran berapa banyak objek yang terhubung dengan piksel tertentu. Berikut adalah rumus untuk menghitung jumlah konektivitas.



Dimana :

Nk merupakan nilai dari 8 tetangga di sekitar *pixel* yang akan dianalisa, dan nilai S = {1,3,5,7}

N0 adalah nilai dari piksel tengah.

N1 adalah nilai dari piksel pada sebelah kanan *central pixel* dan sisanya diberi nomor berurutan dengan arah berlawanan jarum jam

1. Ulangi langkah 1 dan 2 untuk semua piksel yang cocok dengan Template T1.

semua lokasi *pixel* yang cocok dengan *template* T1.

1. Ulangi langkah 1–3 untuk *template* T2, T3 dan T4.
   1. Pencocokkan *template* T2 akan dilakukan pada sisi kiri dari obyek dengan arah dari bawah ke atas dan dari kiri ke kanan.
   2. Pencocokkan *template* T3 akan dilakukan pada sisi bawah dari obyek dengan arah dari kanan ke kiri dan dari bawah ke atas.
   3. Pencocokkan *template* T4 akan dilakukan pada sisi kanan dari obyek dengan arah dari atas ke bawah dan dari kanan ke kiri.
2. Piksel yang ditandai untuk dihapus diubah menjadi putih.

**3.4 Tahap Ekstraksi Fitur**

Pada penelitian ini terdapat 3 tahap ekstraksi fitur, yakni fitur posisi titik, fitur jumlah titik dan chaincode. Berikut adalah penjelasan dari masing – masing fitur :

**3.4.1 Chaincode**

1

8

2

3

4

5

6

7

8 Arah Chain code

Pada pengenalan pola, chaincode merupakan suatu teknik untuk menggambarkan suatu struktur dari suatu objek. Chain code diperoleh dengan cara menelusuri piksel batas objek berdasarkan arah-arah yang telah ditetapkan. Hasil dari chain code adalah angka – angka yang menunjukkan arah yang mewakili batas objek. Pencarian chain code hanya bisa dilakukan pada citra biner.

Berikut adalah cara pengambilan chain code dari suatu objek di dalam sebuah citra :

1. Cari piksel hitam yang hanya memiliki 1 tetangga dengan cara menelusuri pixel pada citra dimulai dari pojok kiri atas sampai menemukan piksel yang berwarna hitam, jika tidak ditemukan piksel hitam yang hanya memiliki 1 tetangga maka ambil piksel hitam pertama yang ditemui.
2. Lakukan iterasi pada gambar
3. Ubah piksel saat ini menjadi 0
4. Ikuti prioritas arah 1 sampai 8
5. Pindah posisi piksel
6. Tambahkan arah ke chain code

Panjang chain code dari suatu objek dapat berubah – ubah sesuai bentuk dari suatu objek tersebut. Pada penelitian ini akan dilakukan klasifikasi dengan Neural Network. Input dari neural network harus tetap, tidak dapat berubah – ubah. Oleh karena itu pada penelitian ini akan dilakukan normalisasi chain code. Yaitu membuat panjang chain code dari suatu citra tetap jumlahnya dan tidak berubah-ubah.

**3.4.1.1 Normalisasi Chaincode**

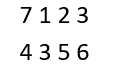
Normalisasi suatu chain code dilakukan agar jumlah chain code yang pada mulanya berubah – ubah sesuai bentuk dari suatu objek dapat dijadikan tetap. Pada penelitian ini panjang chain code dari setiap citra huruf Arab akan ditetapkan menjaidi 10.

Berikut adalah langkah – langkah dari normalisasi chain code :

1. Chain code diubah menjadi matriks 2 dimensi. Baris pertama adalah nilai dari chain code. Baris kedua adalah frekuensi terjadinya setiap angka dalam chain code.

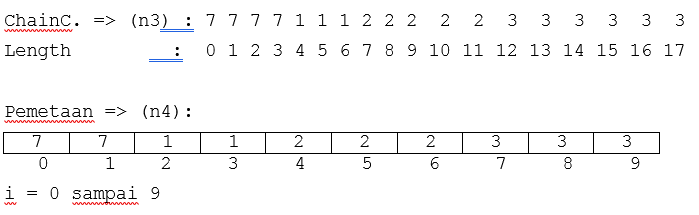
Seperti chain code berikut : 7777311122222583353333, setlah dilakukan tahap pertama dari normalisasi chain code akan menjadi 2 x 9 matriks :



1. Hilangkan semua nilai yang memiliki frekuensi hanya 1.
2. Tampilkan chaincode sesuai dengan frekuensi terjadinya :

777711122222333333

1. Lakukan pemetaan terhadap chaincode



n4[0] = n3[round(i/9 x n3length-1)]

= n3[0/9x17]

= n3[0] = 7

n4[0] = 7

n4[1] = n3[1/9x17] = n3[2] = 7

n4[2] = n3[2/9x17] = n3[4] = 1

n4[3] = n3[3/9x17] = n3[6] = 1

n4[4] = n3[4/9x17] = n3[8] = 2

n4[5] = n3[5/9x17] = n3[9] = 2

N4[6]= n3[6/9x17] = n3[11] = 2

n4[7] = n3[7/9x17] = n3[13] = 3

n4[8] = n3[8/9x17] = n3[15] = 3

N4[9]= n3[9/9x17] = n3[17] = 3

**3.4.2 Fitur Jumlah Titik**

Fitur jumlah titik merupakan fitur yang penting dalam huruf Arab, karena beberapa huruf Arab memiliki bentuk yang sama namun hanya dibedakan berdsarkan jumlah titik. Seperti huruf ب, ث, dan ت.

Jumlah titik diperoleh dengan cara melakukan iterasi pada gambar huruf dari sudut kiri atas ke kanan, kemudian kebawah, jika ditemukan titik hitam pertama hitung chaincode dari titik hitam tersebut. Huruf yang memiliki titik akan memiliki chaincode lebih dari 1. Kemudian akan diperiksa, jika chaincode yang ditemukan kurang dari 7 maka akan dihitung sebagai chaincode titik, dan dilakukan penjumlahan terhadap jumlah titik. Jika chaincode memiliki panjang lebih dari 7 maka akan dihitung sebagai chaincode dari *body* huruf.

Seperti pada huruf, ث yang memiliki 4 chaincode. Chaincode pertama adalah “60” , chaincode kedua adalah “60”, chaincode ketiga adalah “60” dan chaincode keempat adalah “66666667654545444488881818178787880”. Dari 4 chaincode tersebut didapat 3 chaincode yang memiliki panjang kurang dari 7. Kemudian setiap kemunculan chaincode yang kurang dari 7 akan dilakukan perhitungan jumlah titik. Dari 4 chaincode pada huruf ث didapat jumlah titik adalah 3.

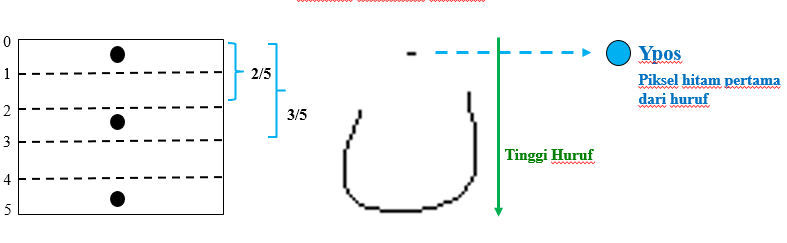
C:\Users\ainawind27\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tahoma_tsa_terpisah_zhangsuen.png

Hasil thinning huruf Tsa

**3.4.3** **Fitur Posisi Titik**

Fitur posisi titik merupakan hal yang penting pada huruf Arab. Beberapa huruf Arab memiliki bentuk dan jumlah titik yang sama, namun dibedakan berdasarkan posisi dari ttik tersebut.

Posisi titik diperoleh dengan cara menghitung piksel hitam pertama yang ada pada gambar yang disebut dengan YPos, kemudian gambar akan dibagi menjadi menjadi 5 bagian. Jika YPos berada pada posisi kurang dari 2/5 tinggi gambar maka posisi titik adalah diatas yang diwakili dengan angka 0. Jika YPos berada pada posisi kurang dari 3/5 tinggi gambar maka posisi titik berada ditengah yang diwakili dengan angka 1. Jika YPos berada pada posisi lebih dari 3/5 tinggi gambar maka posisi titik adalah dibawah yang diwakili dengan angka 2.



Cara menentukan posisi titik

**3.5 Perancangan Sistem dan Eksperimen Pengklasifikasi**

Dalam penelitian ini dikenal data latih dan data uji. Berikut penjelasan dari kedua jenis data tersebut.

1. Data latih

Data latih merupakan kumpulan data huruf arab sesuai sesuai posisinya pada kalimat yakni terpisah, di awal, di tengah, dan di akhir yang akan dilatih untuk kemudian disimpan sebagai *data set* huruf arab.

1. Data uji

Data uji merupakan kumpulan data kalimat Arab yang akan dimasukan dalam tahap pengujian untuk kemudian di segmentasi, diambil fiturnya dan dibandingkan dengan *data set* dari data latih dan ditentukan kemiripan data huruf Arab tersebut.

Jumlah huruf arab yang akan dilatih sebanyak 29 huruf arab, 1 huruf arab memiliki 4 posisi yang berbeda-beda, yaitu posisi terpisah, diawal, ditengah dan diakhir. Berikut teknik yang digunakan dalam pengambilan data latih.

1. Ketik huruf 29 hurufa Arab dari ا sampai ى menggunakan Microsoft Word , huruf arab ditulis sesuai posisinya pada kalimat, yakni posisi terpisah, di awal, di tengah dan di akhir.
2. Potong setiap huruf menggunakan Snipping Tool
3. Hasil pemotongan huruf disimpat sebagai *data set* huruf arab

Saat pengambilan data uji, dilakukan dalam tiga tipe tingkat kesulitan, yaitu:

1. Pengujian huruf

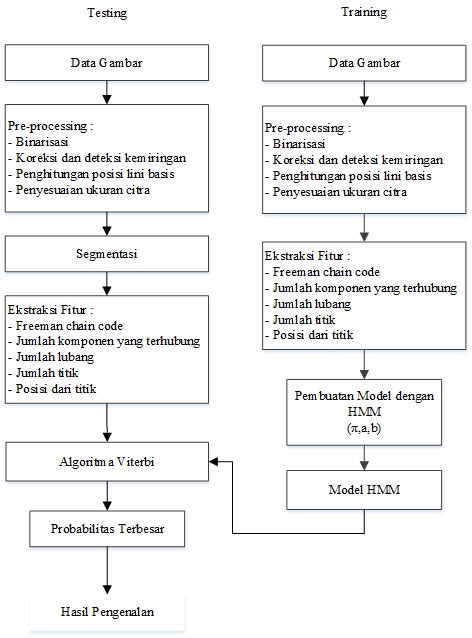
Saat pengujian huruf ini dilakukan sebanyak 29 huruf arab, dimana 1 huruf memiliki 4 posisi yang berbeda, yakni posisi terpisah, diawal, ditengah dan diakhir.

1. Pengujian kalimat

Saat mengujian kalimat dilakukan dengan memasukkan 10 kalimat yang berbeda kedalam sistem. Kalimat kemudian di segmentasi dan diambil fiturnya. Hasil pengujian kalimat berupa persentase keberhasilan segmentasi dan persentase pengenalan huruf Arab.

**3.5.1 Tahap Pelatihan dan Pengujian dengan Metode Hidden Markov Model**

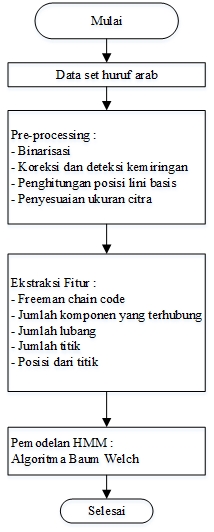
Dalam penelitian ini menggunakan Hidden Markov Model (HMM) untuk tahap klasifikasi. Secara umum, diagram blok dari proses pengenalan huruf arab menggunakan metode Hidden Markov Model dalam Gambar 3.2.



Gambar 3.2 Proses pengenalan huruf arab menggunakan Hidden Markov Model

1. Tahap pelatihan dengan metode Hidden Markov Model

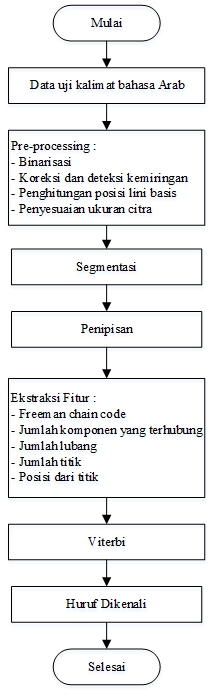
Tahap pelatihan dilakukan untuk menentukan parameter estimasi, sehingga  
terbentuk model markov tersembunyi berupa . Berikut diagram alir pelatihan dengan metode model markov tersembunyi.



Gambar 5.11 Diagram alir pelatihan dengan metode *Hidden Markov Model* (HMM)

1. Tahap pengujian dengan metode model markov tersembunyi

Pengujian dilakukan untuk mengetahui seberapa besar nilai peluang data citra uji  
terhadap kecocokannya dengan data latih*.* Proses pengujian atau evaluasi ini menggunakan algoritma *viterbi* karena yang ingin dicari hanya nilai peluang dari data uji di setiap basis data hasil pelatihan. Nilai peluang yang paling tinggi, menentukan kecocokan antara data uji terhadap basis data latih. Berikut algoritma *viterbi* yang digunakan dalam tahap pengujian dengan metode HMM.



Gambar 5.12 Diagram alir pengujian dengan Hidden Markov Model

Tahapan pertama adalah memasukkan gambar rangkaian kata dalam huruf Arab. Terhadap gambar tersebut dilakukan pemrosesan awal guna membersihkan gambar huruf dari kerusakan atau karena posisi yang tidak sesuai serta mengeliminasi hal-hal yang tidak diperlukan. Warna gambar kemudian diubah ke dalam warna abu- *(gray scale)*, dan selanjutnya diubah lagi ke dalam warna hitam-putih, selanjutnya dilakukan proses *thinning* dengan algoritma *Zhang Suen* yang berguna untuk mendapatkan jalur huru, kemudian dilakukan tahapan segmentasi dari kalimat menjadi huruf. Huruf yang telah berhasil di segmentasi dimasukkan kedalam metode *Freeman Chain code* dalam tahap ekstraksi fitur pengambilan ciri masing-masing huruf. Hasil dari ekstraksi fitur ini akan digunakan sebagai masukan untuk tahap klasifikasi dengan *Hidden Markov Model* (HMM) yang merupakan proses identifikasi masing-masing karakter dan menetapkan kedalam kelasnya sendiri.