BAB IV

ANALISIS DAN PERANCANGAN SISTEM

4.1. Analisis Kebutuhan Sistem

Analisis kebutuhan sistem terdiri dari deskripsi umum sistem, batasan dan asumsi, analisis masukan sistem, model proses sistem dan analisis keluaran sistem berupa perangkat lunak.

4.1.1. Deskripsi Umum Sistem

Pada penelitian ini penulis membangun suatu perangkat lunak Pengenalan Huruf Tulisan Tangan Menggunakan *Fuzzy Feature extraction* Dengan Pendekatan *Radial Basis Function Neural Network*. Secara umum aplikasi ini dibagi menjadi beberapa tahap yaitu :

a. Melakukan pra-pemrosesan data
b. Melakukan ekstraksi ciri
c. Menghasilkan matriks baru yaitu matriks input JST

Langkah pertama dari sistem ini adalah menerima masukan citra huruf tulisan tangan hasil *scan*, kemudian melakukan pra-pemrosesan data masukan berupa penghalusan citra. Hasil dari citra biner kemudian dilakukan penipisan objek dengan ketebalan 1 *pixel*, selanjutnya hasil penipisan citra ini akan ditentukan koordinat titik-titik ujung (*end points)* dan koordinat titik-titik percabangan (*intersection points)* untuk mengetahui segmen yang terbentuk.

Langkah kedua adalah melakukan ekstraksi ciri dari citra hasil segmentasi yang dikonversi ke dalam aturan *bit*, dengan tipe segmen

tertentu. Hasilnya berupa matriks *value* yang terbentuk dan matriks *interrelationship*. Setelah matriks *interrelationship* dan matriks *value* didapat, langkah berikutnya ialah mengubah kedua matriks tersebut menjadi matriks yang hanya memiliki dimensi 1 x 80 (berasal dari matriks *value*) dan 1 x 100 (berasal dari matriks *interrelationship*).

Langkah ketiga adalah menggabungkan kedua matriks tersebut menjadi sebuah matriks baru yaitu matriks *input* JST yang nantinya menjadi *input* untuk proses Jaringan Syaraf Tiruan *Radial Basis Function* (matriks ini berdimensi 1 x 180). Pada langkah ini dihasilkan penentuan huruf yang dikenali.

4.1.2 Batasan dan Asumsi Analisis Sistem

Berdasarkan analisis yang telah dilakukan, beberapa asumsi yang akan diterapkan dalam mengimplementasikan perangkat lunak yang akan dibangun antara lain:

1. Sistem pengenalan huruf tulisan tangan dilakukan secara tidak langsung. Artinya responden tidak menulisnya langsung pada sistem berupa gerakan *mouse*, tetapi berupa tulisan tangan dalam kertas yang kemudian di *scan*. Hasil scan disimpan dalam format .bmp dengan resolusi 106 x 114 *pixel*.
2. *Input* dari sistem adalah citra digital dan *output*-nya adalah informasi jenis huruf tulisan tangan yang ada pada citra tersebut.
3. Tugas akhir ini hanya terbatas pada masalah pengenalan huruf tulisan tangan yang terdiri dari huruf besar dan huruf kecil.

4. Perangkat lunak ini tidak memiliki bagian *error recovery* (pemulihan kesalahan), dan *error repair* (perbaikan kesalahan). Bagian yang berhubungan dengan *error* hanyalah sebatas *error detect* (pendeteksian kesalahan).

4.1.3. Masukan Sistem

Masukan sistem berupa data yang akan dikembangkan membutuhkan masukan berupa *data collection*, hal ini bertujuan agar pemrosesan yang dilakukan perangkat lunak dapat berjalan dengan lancar. *Data collection* yang menjadi masukan memiliki syarat-syarat sebagai berikut :

1. Data citra masukan berupa huruf tulisan tangan hasil *scan*..
2. Citra masukan memiliki ukuran 106x114 *pixel* dan memiliki format .bmp. Hal tersebut dilakukan agar mempercepat proses pengenalan.

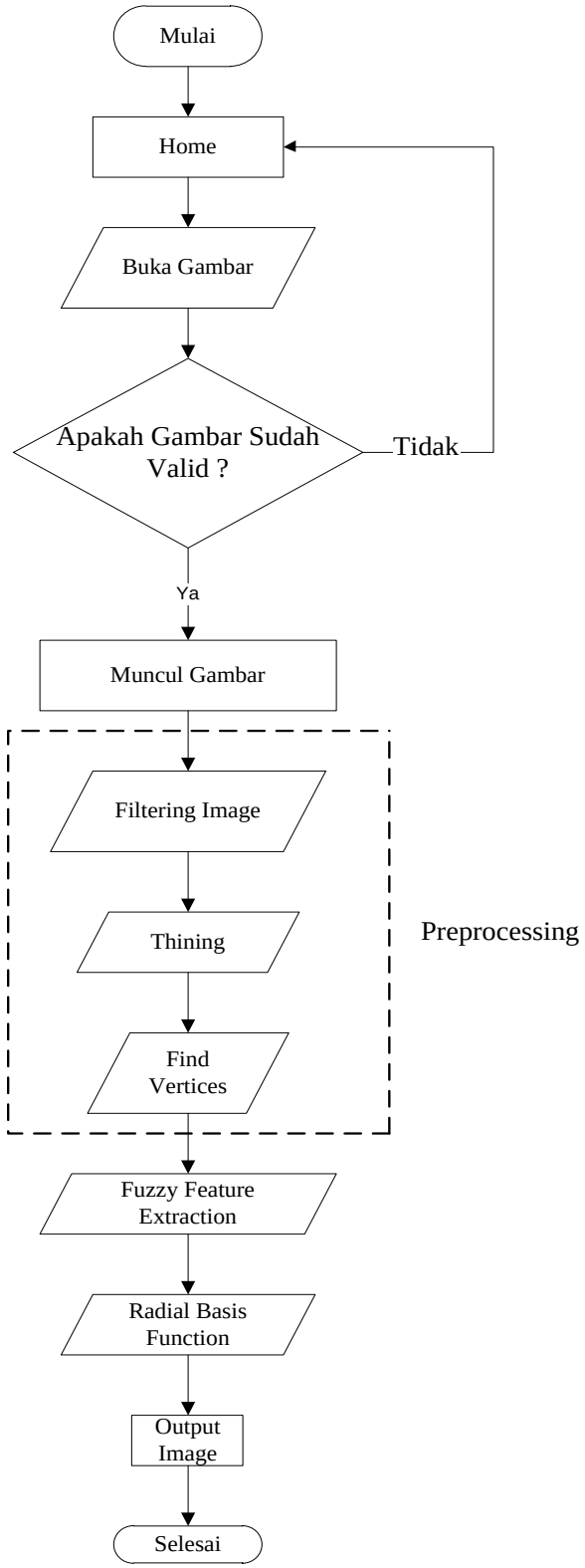
4.1.4. Keluaran Sistem

Keluaran sistem berupa identifikasi bentuk huruf yang dikenali. Hasil proses tersebut terdiri atas :

1. Citra hasil proses (citra masukan, citra biner, penipisan citra).
2. Hasil dari matriks *value*, matriks *interrelationship*, matriks *input* JST.
3. Hasil perhitungan dari algoritma JST *Radial Basis Function*.
4. Waktu proses yang terbentuk pada proses pengenalan suatu citra karakter huruf dan deskripsi objek keluaran.

4.2. Desain Sistem

Pada tahapan ini, penulis melakukan proses desain sistem. Desain sistem tergambar dalam *flowchart*, sebagai berikut :



Gambar 4.1 *Flow Chart* Sistem

Pada proses filtering gambar masukan yang diterima oleh sistem akan diubah menjadi sebuah matrik yang berukuran sesuai dengan ukuran pixel masukan gambar yaitu 106 kolom x 114 *pixel*. Kemudian pada proses *thining*, merupakan proses penghilangan *pixel* terluar dengan cara *iterative boundary erotion process* hingga menghasilkan sebuah *pixel* tulang (skeleton). Gambar masukan yang telah dilakukan proses *thining* selanjutnya akan dicari titik ujung (*end points*) dan titik percabangan (*intersection points*). Kemudian, pada *Fuzzy Feature Extraction* bertujuan untuk mendapatkan karakteristik suatu karakter yang membedakan dari karakter yang lain, yang disebut *feature*. Hasil perhitungan keseluruhan dari JST *Input* dengan metode *Radial Basis Function* menghasilkan matrik keluaran yang kemudian dieksekusi menjadi sebuah huruf yang diinginkan.

4.2.1. Pemahaman Algoritma

Pada subbab ini akan dilakukan analisis terhadap penerapan algoritma yang digunakan dalam membangun perangkat lunak Pengenalan Huruf Tulisan Tangan sebagai berikut :

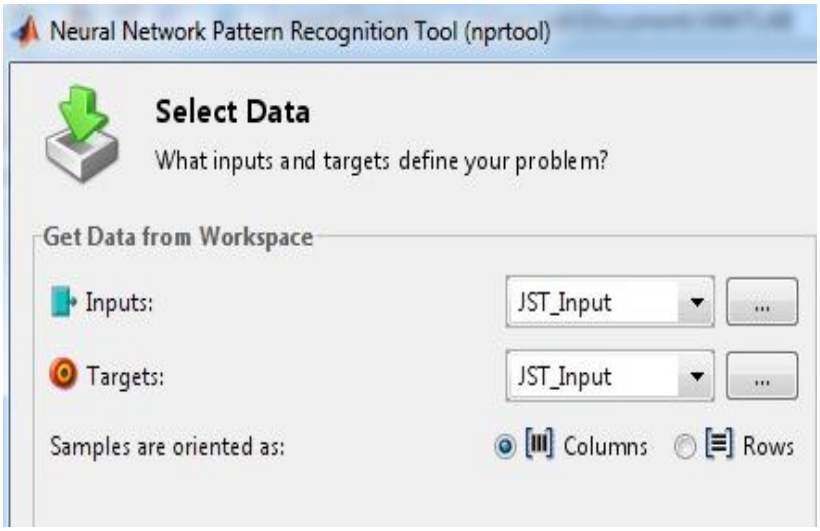
4.2.1.1. *Training* (Pelatihan)

Sebelum pengenalan tulisan tangan dilakukan, penulis akan menjelaskan proses pelatihan. Pada proses pelatihan ini, sampel data yang berjumlah 260 huruf yang masing-masing memiliki 5 sampel huruf, didapat dari :

a. Huruf Kecil = 26 x 5 sampel huruf = 130 huruf
b. Huruf Kapital = 26 x 5 sampel huruf = 130 huruf

Sampel data yang berjumlah 260 huruf memberikan masukan data sampel yang akan disimpan kedalam *storage* data yaitu JST *Input* dan JST Target. Pada *storage* data ini bukan merupakan *data base sistem* yang memerlukan *software* pendukung untuk menyimpan data sampel huruf, melainkan hanya penyimpanan biasa yang tergabung dalam satu folder penyimpanan dalam aplikasi MATLAB itu sendiri. Berikut akan dijelaskan bagaimana proses pelatihan ini bekerja.

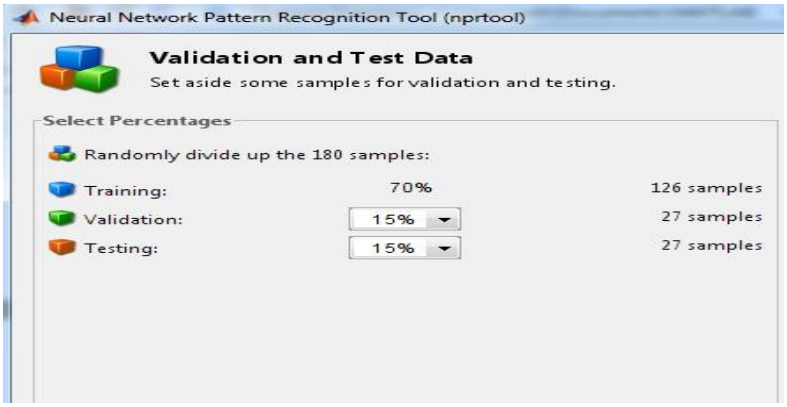
1. Proses *training* ini diawali dengan memasukkan data JST *Input* dan data JST Target pada Select data Neural Network Pattern Recognition Tool (nprtool).



Gambar 4.2 Proses memasukkan data JST*Input* dan data JST Target

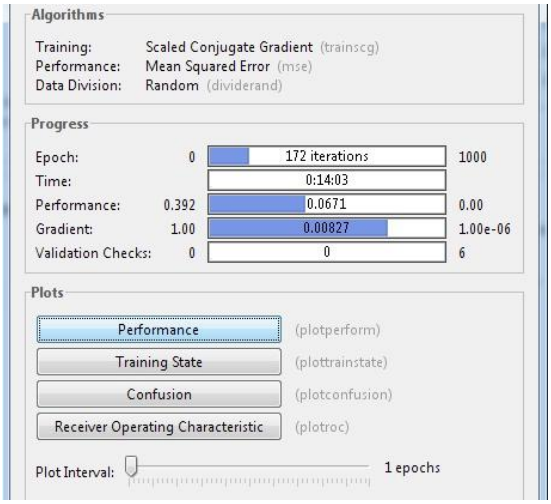
2. Setelah kedua data tersebut dimasukkan, kemudian dimulai proses training dengan memvalidasi dan tes data sebanyak 260 sampel dengan proses training sebanyak 70%, *validation* 15%, dan *testing* 15%. Hal ini dilakukan karena dengan proses validasi dan testing lebih kecil bertujuan untuk mendapatkan proses pelatihan yang lebih besar. Jika kedua proses dinaikkan beberapa persen, maka

proses pelatihan akan turun. Akibatnya data sampel pelatihan menjadi lebih sedikit.



Gambar 4.3 Ilustrasi proses validasi dan tes data

3. Kemudian dimulailah proses pelatihan. Berikut ilustrasi gambar proses pelatihan tersebut.



Gambar 4.4 Proses terjadinya pelatihan sampel huruf

Setelah proses pelatihan dilakukan, maka tahapan selanjutnya adalah proses pengenalan data uji. Data uji merupakan data baru yang akan dibandingkan dengan data training pada tahapan sebelumnya. Proses pengenalan data uji melibatkan metode *Radial Basis Function*. Pada metode ini melakukan proses eksekusi dari *supervised ke*

unsupervised. Supervised yaitu metode pembelajaran dari sekumpulan pola masukan dan keluaran, sehingga pada saat pelatihan diperlukan pola yang terdiri dari vektor masukan dan vektor target yang dinginkan. Sementara itu *unsupervised* yaitu metode pembelajaran yang menggunakan data tak berlabel, pada metode ini juga tidak memerlukan pengawasan dari luar.

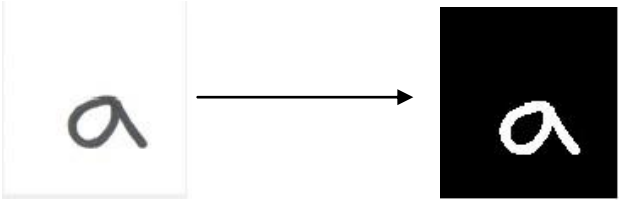
Berdasarkan hal di atas dapat disimpulkan bahwa pada metode *Radial Basis Function,* proses training dilakukan dengan data sampel pelatihan yang sedikit dengan melakukan proses eksekusi data lebih besar.

4.2.1.2. Preprocessing

Preprocessing merupakan langkah awal sebelum dilakukan segmentasi terhadap huruf yang akan dikenali. Ada beberapa proses dalam *preprocessing* ini yakni: *filtering*, *thining*, *search vertices*. Berikut akan penulis jelaskan lebih mendalam mengenai proses pada implementasi tersebut.

4.2.1.2.1. Filtering

Gambar masukan yang diterima oleh sistem akan diubah menjadi sebuah matrik yang berukuran sesuai dengan ukuran pixel masukan gambar tersebut yaitu 106 kolom x 114 *pixel*. Kemudian akan dilakukan perubahan menjadi *binary matrik* (*matrik* yang hanya mempunyai dua nilai yakni: 0 dan 1). Perhatikan gambar berikut:



(a) **(b)**

Gambar 4.5 (a). Input gambar, (b) *Filtering*

Pada tahap ini juga akan dilakukan pembersihan terhadap *noise* yang ada pada gambar. Hal ini bertujuan agar menghindari distorsi pada tahap segmentasi. Namun, pada tahap pembersihan *noise* itu sendiri kecil kemungkinan untuk terhapus semua, sehingga perlu adanya *software* tambahan untuk membersihkan *noise* yaitu *software* yang berkaitan dengan *editing image* seperti Adobe Photoshop. Model *filtering* yang digunakan pada aplikasi ini adalah *median filter*.

4.2.1.2.2 *Thining*

Proses *thining* merupakan proses penghilangan *pixel* terluar dengan cara *iterative boundary erotion process* hingga menghasilkan sebuah *pixel* tulang (skeleton). Secara garis besar algoritma *thining* sebagai berikut:

1. Nilai *pixel* pada point p bernilai satu.

$$f_1(p) = 1 \qquad (4.1)$$

2. Jumlah 8 tetangga yang berhubungan dengan point p lebih besar atau sama dengan dua dan lebih kecil atau sama dengan enam.

$$f_2(p) = \sum_{i=1}^{8} n_i \qquad (4.2)$$

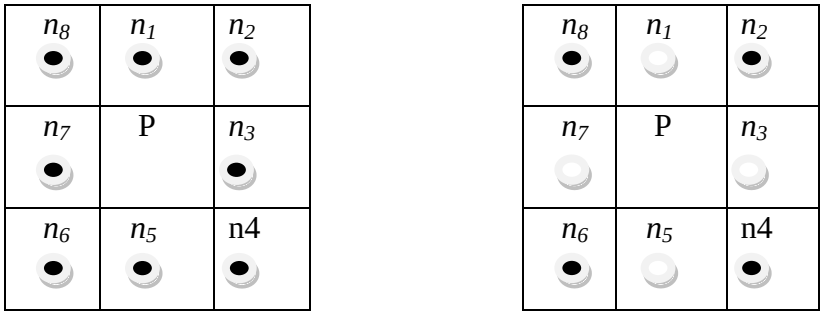
$$2 \le f_2(p) \le 6$$

3. Terdapat *white pixel* yang terletak sebelum *black pixel* untuk 8 tetangganya, dan 8 tetangganya dilihat dengan urutan n_3, n_2, n_1, n_8, n_7, n_6, n_5 and n_4, yang sama dengan crossing number X_R (dihitung dengan melihat jumlah transisi dari point pada pattern/obyek ke point pada background dan sebaliknya saat mengelilingi 8 tetangganya berlawanan arah jarum jam). Nilainya adalah genap karena transisi muncul secara berpasangan. *Crossing number* yang valid ada dua :

$$f_3(p) = X_R(p) = \sum_{i=1}^{8} |n_{i+1} + n_i| \qquad (4.3)$$

$$f_3(p) = 2$$

Pada gambar 4.6 dibawah ini adalah contoh konfigurasi *pixel* yang berbeda dan *crossing number* nya.



$X_R(p) = 0$ (a)

$X_R(p) = 8$ (b)

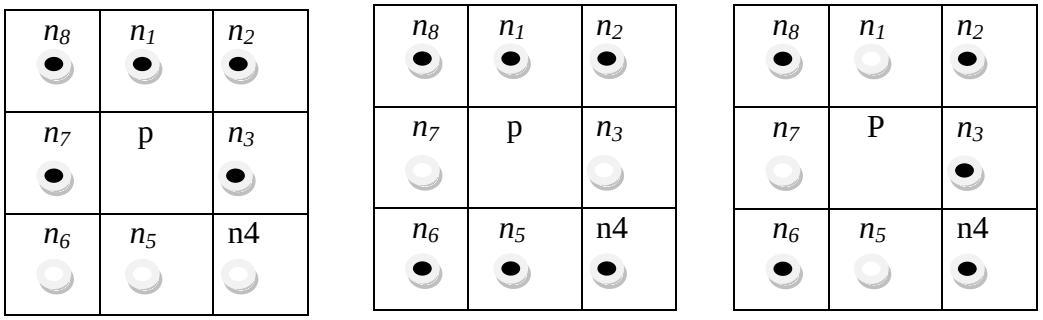
Gambar 4.6 contoh konfigurasi *pixel* yang berbeda dan *crossing number*

Pada gambar 4.6 (a) semua 8 tetangganya adalah *black pixel*, sedangkan pada gambar 4.6 (b) *pixel* n_2, n_4, n_6 and n_8 adalah *black*

pixel. Algoritma ini tidak akan menerima contoh pada gambar 4.6 (a) karena bertentangan dengan dengan kondisi 4.2 yaitu,

$$f_2(p) = \sum_{i=1}^{8} n_i$$

$$2 \le f_2(p) \le 6$$



$X_R(p) = 2$ $X_R(p) = 4$ $X_R(p) = 6$

(a) (b) (c)

Gambar 4.7 contoh *crossing number*

Ketiga contoh diatas memiliki crossing number dua, empat dan enam. *Crossing number* akan bernilai sama jika *white pixel* dan *black pixel* ditukar satu sama lain.

4. Salah satu dari n_1, n_3 dan n_7 dari 8 tetangganya adalah *background*

$$f_4(p) = n_1 \times n_3 \times n_7 \quad (4.4)$$

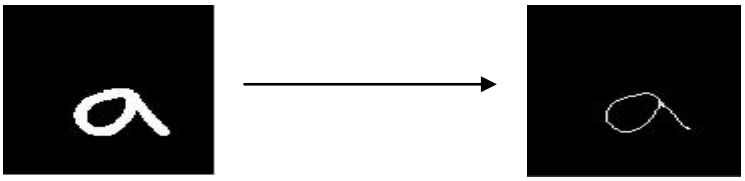
$$f_4(p) = 0$$

5. Salah satu dari n_1, n_7 and n_5 dari 8 tetangganya adalah *background*

$$f_5(p) = n_1 \times n_7 \times n_5 \quad (4.5)$$

$$f_5(p) = 0$$

Pada Gambar 4.8 di bawah ini terlihat gambar yang dihasilkan dari proses *thining.*

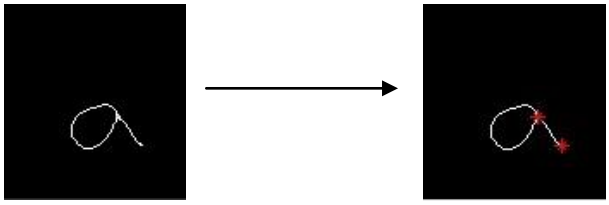


Gambar 4.8 Proses *thining*

4.2.1.2.3. *Search vertices*

Gambar masukan yang telah dilakukan proses *thining* selanjutnya akan dicari titik ujung (*end points*) dan titik percabangan (*intersection points*). Secara garis besar algoritma *search vertices* sebagai berikut:

1. Telusuri gambar secara horizontal, mulai dari kiri atas hingga kanan bawah.
2. Cek pada setiap *pixel p* jumlah tetangga (x1, x2, x3,..., x8 adalah *value* untuk kedelapan tetangga dari *p*, pengurutan dimulai dari tetangga sebelah kanan dan seterusnya mengikuti arah jarum jam) yang ia miliki.
3. Jika *p* setidaknya tiga tetangga, maka kategorikan sebagai titik percabangan (*intersection point*) kemudian simpan ke dalam *matrik intersection.*
4. Jika *p* hanya memiliki satu tetangga, maka dikategorikan sebagai titik ujung (*end points*) kemudian simpan ke dalam *matrik end*.



(a) (b)

Gambar 4.9 (a) Gambar *thining*, (b) Gambar verteks-verteks (*found vertices)*

Pada proses *thining* dan *found vertices* akan didapat nilai keluaran berupa nilai koordinat *End Points* dan nilai koordinat *Intersection Points*. Berikut akan kita hitung jumlah titik-titik piksel yang akan ditemukan oleh sistem ini untuk menentukan koordinat sehingga nantinya akan menghasilkan nilai *end point* dan *intersection point*. Berikut akan kita hitung jumlah titik-titik piksel yang akan ditemukan oleh sistem ini untuk menentukan koordinat sehingga nantinya akan menghasilkan nilai *end point* dan *intersection point*.

Tabel 4.1 Nilai titik koordinat x dan y pada segmen pertama

Nilai (x,y) pada segmen pertama									
68,73	67,73	66,74	65,74	64,74	63,74	62,73	61,72	61,71	60,70
59,69	59,68	59,67	58,66	58,65	58,64	57,63	57,62	56,61	56,60
56,59	55,58	55,57	55,56	55,55	54,54	53,53	53,52	52,51	51,50

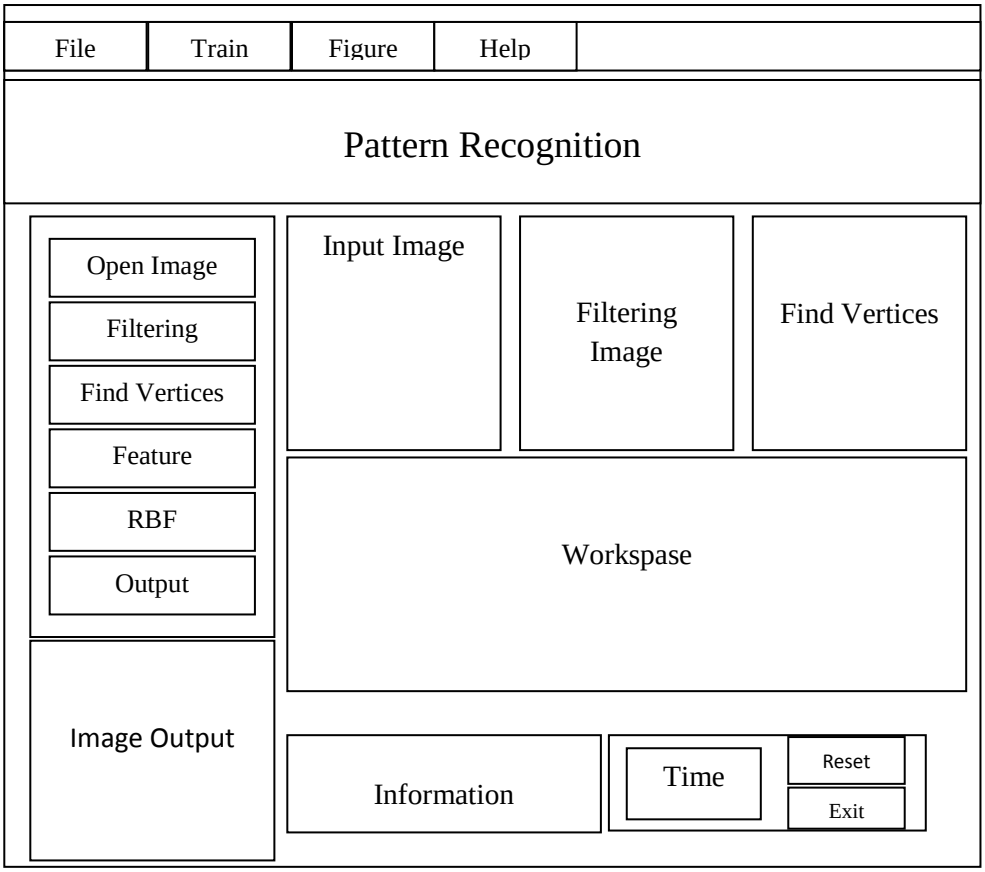
Tabel 4.1 Nilai titik koordinat x dan y pada segmen pertama terdiri atas 30 gambar, sebagai contoh nilai 68,73 diperoleh dari algoritma *find vertices* untuk nilai 68 nya, sedangkan nilai desimal (,73) diperoleh dengan mengklik *Figure, Skeleton Image,* dan klik pada titik-titik *end point*nya. Kelemahan dari algoritma ini adalah memasukkan inputnya masih membutuhkan campur tangan manusia (manual) agar hasilnya sesuai dengan nilai *End Points* dan

Intersection Points. Sebagai contoh nilai 68,73 berhubungan dengan gambar 4.11 dan gambar 4.12.

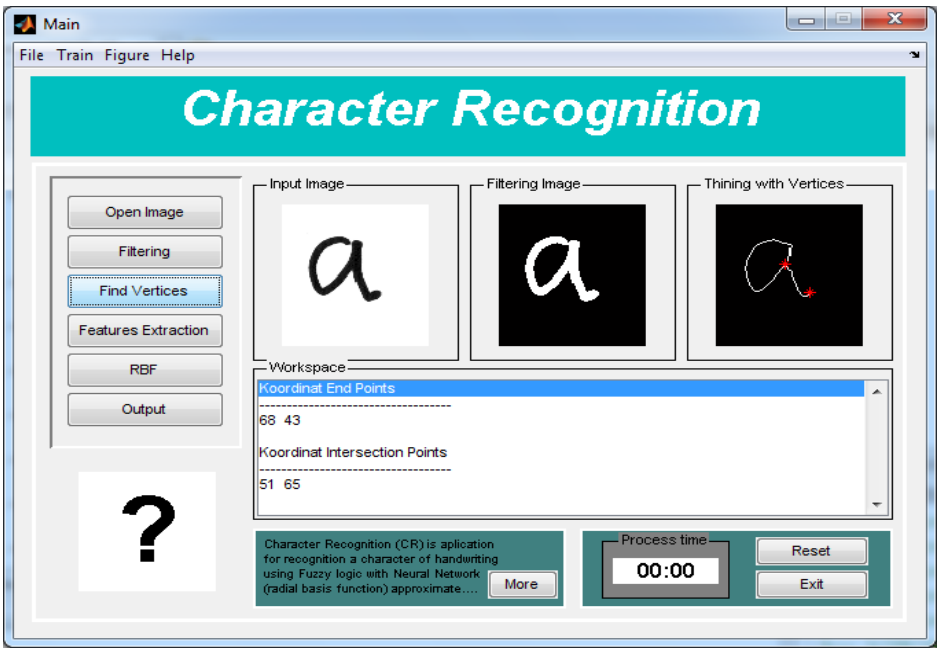
Tabel 4.2 Nilai titik koordinat x dan y pada segmen kedua

Nilai (x,y) segmen kedua									
51,50	50,51	49,52	48,53	47,54	47,55	46,56	45,57	44,58	43,59
43,60	42,61	41,62	40,63	39,64	38,65	37,66	36,67	35,68	34,69
33,69	32,70	31,70	30,70	29,70	28,69	27,68	27,67	26,66	25,65
25,64	25,63	24,62	24,61	24,60	23,59	23,58	23,57	23,56	23,55
22,54	22,53	22,52	22,51	23,50	23,49	23,48	23,47	23,46	24,45
24,44	25,43	25,42	26,41	26,40	27,39	27,38	28,37	28,36	29,35
30,34	31,33	32,33	33,32	34,32	35,31	36,31	37,31	38,30	39,30
40,30	41,30	42,29	43,29	44,30	45,30	46,30	47,31	48,31	49,32
50,32	51,32	52,32	53,33	54,33	55,34	55,35	54,36	54,37	54,38
54,39	54,40	54,41	54,42	53,43	53,44	53,45	53,46	53,47	52,48
52,49									

Pada gambar 4.10 merupakan Rancangan Perangkat Lunak Pengenalan Huruf Tulisan Tangan Menggunakan *Fuzzy Feature Extraction* Dengan Pendekatan *Radial Basis Function*.

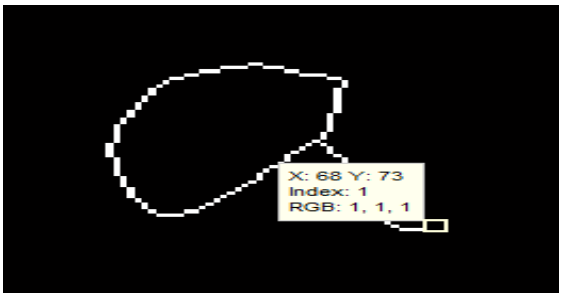


Gambar 4.10 Rancangan Perangkat Lunak



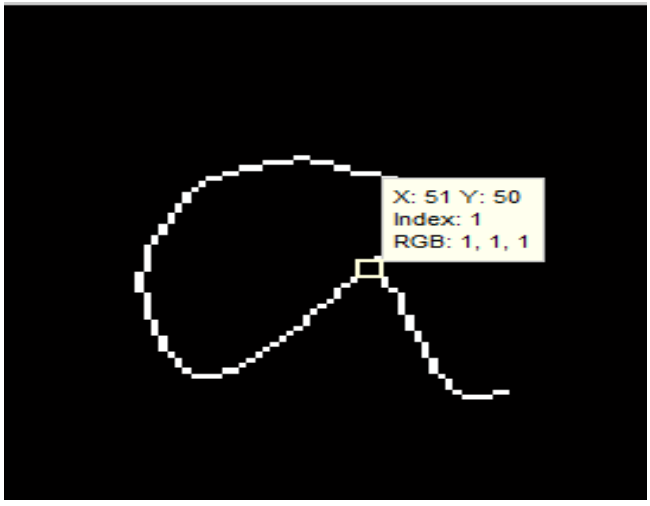
Gambar 4.11 Contoh nilai Koordinat *End points* dan Koordinat *Intersection Points* untuk huruf ‘a’

Nilai x pada segmen pertama di dapat dari nilai 68 dari Koordinat *End Points* seperti pada gambar di atas, sedangkan nilai y nya dapat di ketahui dengan membuka figure kemudian skeleton image lalu muncullah gambar seperti pada gambar 4.12 di bawah ini.



Gambar 4.12 Gambar pencarian nilai x dan y pada segmen pertama (68, 73)

Demikian juga dengan pencarian x pada segmen kedua didapat nilai 51 dari Koordinat Intersection Points seperti pada gambar 4.8, sedangkan nilai y nya dapat di ketahui dengan membuka figure kemudian skeleton image lalu muncullah gambar seperti pada gambar 4.13 di bawah ini.



Gambar 4.13 Gambar pencarian nilai x dan y pada segmen kedua (51, 50)

4.2.1.3. Fuzzy Feature Extraction

Fuzzy Feature Extraction bertujuan untuk mendapatkan karakteristik suatu karakter yang membedakan dari karakter yang lain, yang disebut *feature*. Karakteristik ini dapat berupa titik koordinat awal dan akhir dari suatu goresan dan fragmen garis penyusun karakter. *Feature* yang diperoleh digunakan untuk mendefinisikan kelas suatu karakter. *Feature* merupakan *high-level-attribute* dari data goresan karakter. Setelah proses *feature extraction* selesai, maka didapat *feature* dari sebuah huruf. *Feature* ini mempresentasikan informasi struktural dari sebuah huruf. Untuk mendapatkan *fragmen* sesuai kategori, suatu pola tulisan tangan dipisahkan pada *segmentation point*-nya.

Berikut akan dijelaskan tahapan-tahapan proses sistem *fuzzy feature extraction.* Tahapan pertama adalah segmentasi terhadap huruf, selanjutnya dilakukan penentuan tipe dari masing-masing segmen tersebut. Dari langkah ini dihasilkan suatu nilai keluaran yakni suatu matrik yang diterjemahkan oleh Jaringan Syaraf Tiruan melalui metode *Radial Basis Function*. Algoritma pada proses *fuzzy feature extraction* dijelaskan sebagai berikut :

***Input data*:**

a. *m x n binary image matrik* (yang telah dilakukan *thining*).

b. Koordinat *vertices* (titik ujung dan titik percabangan).

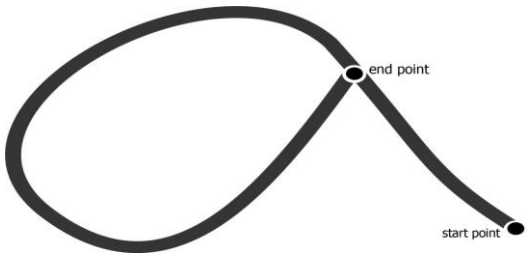
***Output data*:**

a. *Matrik* fungsi dan huruf.

b. *Matrik interrelationship* (*matrik* keterhubungan antar segmen).

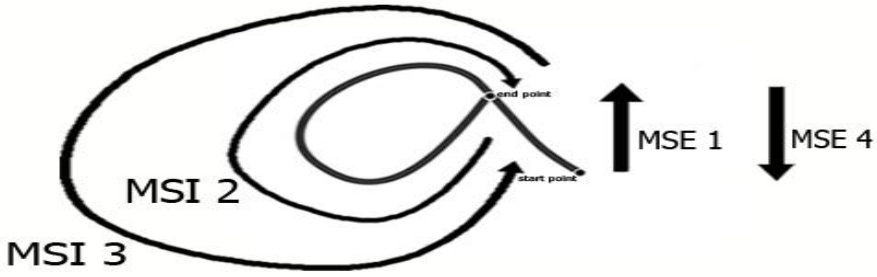
Langkah 1:

Pada langkah ini dilakukan penelusuran dengan dimulai dari tiap-tiap titik ujung (*end points*) dari objek. Titik ujung akan menjadi *start_point* (yang merupakan koordinat awal dari sebuah segmen). Penelusuran dilakukan dengan mencari koordinat tetangga yang belum ditelusuri dari setiap *state* (posisi koordinat pada saat ini berada). Hal ini dilakukan secara berulang hingga mendapatkan kondisi untuk berhenti. Kondisi berhenti ditentukan ketika saat *state* yang merupakan koordinat percabangan (*intersection points*). *State-state* yang ditemukan selanjutnya disimpan dalam suatu *matrik* segmen ujung MSE (*Matrik Segmen End point*).



Gambar 4.14 Gambar MSE (*Matrik Segmen End point*)

Disisi lain lakukan penelusuran serupa yang dimulai dari setiap titik-titik percabangan (*intersection points*). Hal ini terus dilakukan sampai ditemukan *state* yang merupakan titik percabangan (intersection points) atau titik ujung (end point). Kemudian simpan semua *state* yang telah ditelusuri tadi kedalam *matrik* segmen percabangan (MSI). Setelah itu lakukan penyeleksian terhadap *matrik* MSI agar tidak terdapat duplikasi segmen (menghilangkan segmen yang ditulis lebih dari satu kali).



Gambar 4.15 Gambar MSI (Matrik Segmen Intersection)

Langkah 2:

Berdasarkan objek yang terdapat pada gambar 4.12, terdapat dua potongan segmen yang nantinya akan ditentukan oleh sistem aplikasi ini sendiri yaitu berupa matrik *value* dan matrik *interrelationship*.

(a)

Klasifikasi Tipe Segmen

	line	*right*	*left*	*loop*
horizontal				
vertical				
right slope				
left slope				

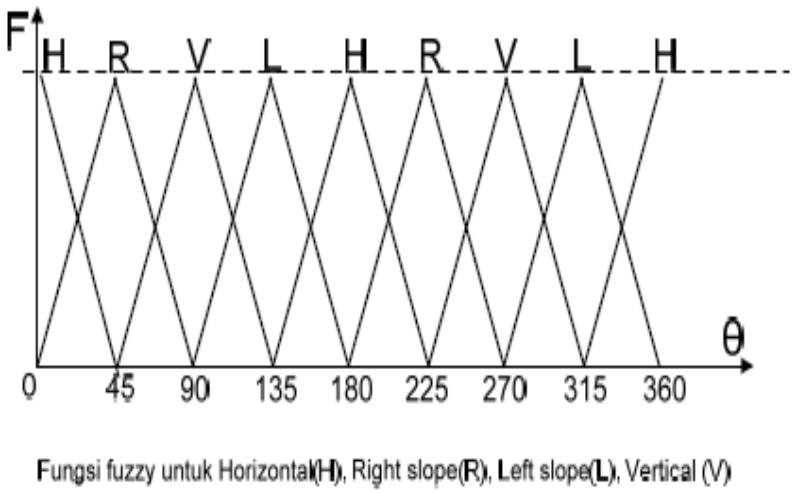
(b)

Gambar 4.16 (a) potongan huruf a (b) klasifikasi tipe segmen

Pada gambar 4.16 (a) huruf disegmentasi sesuai dengan tipe-tipe segmennya.

Langkah 3:

Mencari nilai fungsi keanggotaan fuzzy untuk keanggotaan Horizantal, Right Slope, Left Slope, atau Vertikal.



Gambar 4.17 Fungsi Fuzzy H,R,V,L tergantung pada sudut θ yang dibentuk.

(Sumber : Gilewski, 1997)

$$
\begin{aligned}
m &= (y2-y1) / (x2 - x1) \\
&= (50 - 73) / (51 - 68) \\
&= (-23/-17) \\
&= 1{,}3529
\end{aligned}
$$

m digunakan untuk menghitung θ yaitu sudut yang bukan bentuk *loop*, dimana (x_1,y_1) dan (x_2,y_2) adalah pasangan koordinat *start point* dan *end point* dari gambar 4.9 dan 4.10 yaitu (68,73) dan (51,50).

$$
\begin{aligned}
\theta &= \arctan(m) \\
&= \arctan(1{,}3529) \\
&= 53{,}5299 \approx 54
\end{aligned}
$$

Berdasarkan koordinat *start point* dan *end point tadi*, maka kita dapat mencari apakah berbentuk loop atau tidak dengan menentukan nilai *gradien* (*m*) dengan memperhitungkan k yaitu jumlah state yang ada pada segmen tersebut, d yaitu jarak yang dibentuk oleh sudut *α* (Kurva atau Loop). Untuk mengetahui apakah segmen merupakan sebuah *loop*, maka segmen tersebut harus memenuhi kondisi berikut ***m* <= (*k* + *d*)* *α* / 360**
Berikut perhitungan nilai gradiennya :

Segmen pertama :	Segmen kedua :
$m = (k + d)\ (\alpha/360)$	$m = (k + d)\ (\alpha/360)$
$= (30 + 29)\ (53/360)$	$= (101 + 11)\ (40/360)$
$= (59)\ (0{,}14722)$	$= 12{,}4432$
$= 8{,}68598$	$d <= m$
$d <= m$	loop = true
loop = true	else
else	loop = false
loop = false	loop = 10000000

Dan dari *gradien* tersebut kita dapat melakukan klasifikasi pertama dari segmen *horizontal* (*H*), *vertical* (*V*), *Right slope* (*R*), *Left slope* (*L*) dengan fungsi keanggotaan *fuzzy* berikut :

FH(θ) = keanggotaan Fungsi fuzzy untuk segmen *horizontal*

FH(θ) = 1- *min*{*min*[|a|, |b|, |c|]/45,1}

Jika θ = 54

a = (0 + θ)/45 = (0 + 54)/45 = 54/45 = 1,2

b = (180 - θ)/45 = (180 - 54)/45 = 2,8

c = (360 - θ)/45 = (360-54)/45 = 6,8

FH(54) = 1- *min*{*min*[|a|, |b|, |c|]/45,1}

= 1- *min*{*min* |1,2|, |2,8|, |6,8|,1}

=.1- *min*{*min*[|1,2|,1}

FH (54) = 1 – 1 = 0

FV(θ) = keanggotaan Fungsi fuzzy untuk segmen *vertical*

FV(θ) = 1-*min*{*min*[|d|, |e|]/45,1}

Jika θ = 54

d = (90 - θ)/45 = (90 - 54)/45 = 0,8

e = (270 - θ)/45 = (270 - 54)/45 = 4,8

FV(54) = 1- *min*{*min*[|d|, |e|]/45,1}

= 1- *min*{*min* |0,8|, |4,8|,1}

=.1- *min*{*min*[|0,8|,1}

FV (54) = 1 – min = 1 – 0,8 = 0,2 ≈ 0

FR(θ) = keanggotaan Fungsi fuzzy untuk segmen *Right slope*

FR(θ) = 1-*min*{*min*[|f|, |g|]/45,1}

Jika θ = 54

f = (45 - θ)/45 = (45 - 54)/45 = -0,2 = 0,2

g = (225 - θ)/45 = (225 - 54)/45 = 3,8

FR(54) = 1- *min*{*min*[|f|, |g|]/45,1}

= 1- *min*{*min* |0,2|, |3,8|,1}

=.1- *min*{*min*[|0,2|,1}

FR (54) = 1 – min = 1 – 0,2 = 0,8 ≈ 0

FL(θ) = keanggotaan Fungsi fuzzy untuk segmen *Left slope*

FL(θ) = 1-*min*{*min*[|h|, |i|]/45,1}

h = (135 - θ)/45 = (135 - 54)/45 = 1,8

i = (315 - θ)/45 = (315 - 54)/45 = 5,8

FL(54) = 1- *min*{*min*[|h|, |i|]/45,1}

= 1- *min*{*min* |1,8|, |5,8|,1}

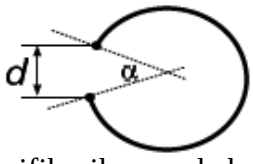
=.1- *min*{*min*[|1,8|,1}

FL (54) = 1 – min = 1 – 1 = 0

Setelah didapat nilai maksimal yaitu nilai FL sebesar 1,8222. Dengan demikian, nilai matrik yang di dapat yaitu 00001. Sedangkan untuk nilai matrik disebelah kanan dikarenakan hasil dari nailai FR dan FL bernilai nol (0), maka dengan otomatis nilai *baseline* bernilai 1 karena harus ada nilai 1 disebelah kanan. Dengan begitu nilai matrik yang didapat yaitu 100, sehingga nilai matrik keseluruhan adalah 00001100.

Langkah 4:

Untuk mengetahui apakah segmen merupakan sebuah *loop*, maka segmen tersebut harus memenuhi kondisi berikut ***m* <= (*k* + *d*)* *α* / 360**, *d* merupakan jarak antara *start point* ke *end point* (jarak antar koordinat awal segmen dengan koordinat akhir segmen), *k* merupakan jumlah *state* yang ada pada segmen tersebut (jumlah baris dari *matrik* segmen), dan *α* merupakan *angle threshold* (besar batas sudut yang masih bisa dikatakan sebagai sebuah *loop*).



Gambar 4.18 Mengklasifikasikan apakah sebuah segmen merupakan loop

(Sumber : Gilewski, 1997)

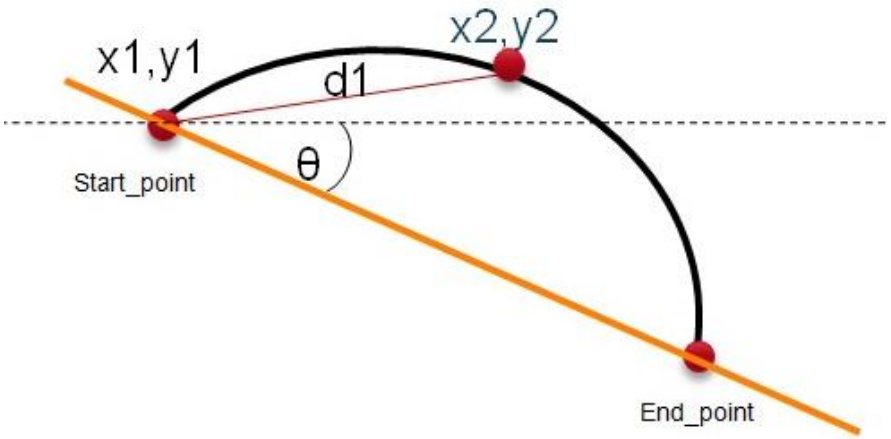
Setelah ditentukan oleh sistem titik koordinatnya, maka akan ditentukan titik mana yang akan dihitung. Pada segmen pertama yaitu berupa garis miring, akan dihitung berapa nilai *d*. Sedangkan pada segmen kedua yang berupa segmen seperti lingkaran, penulis akan melakukan perpotongan untuk menentukan nilainya. Berikut akan dihitung nilai dari tiap segmen.

$$d1^2 = (x2 - x1)^2 + (y2 - y1)^2 \qquad (4.6)$$

Dimana:

d1 merupakan garis perpotongan untuk membentuk yang sudut menghubungkan dengan garis *d2*.

x1, *x2*, *y1*, *y2* merupakan titik sudut untuk menentukan garis *d1*.



Gambar 4.19 Ilustrasi pencarian *d1*

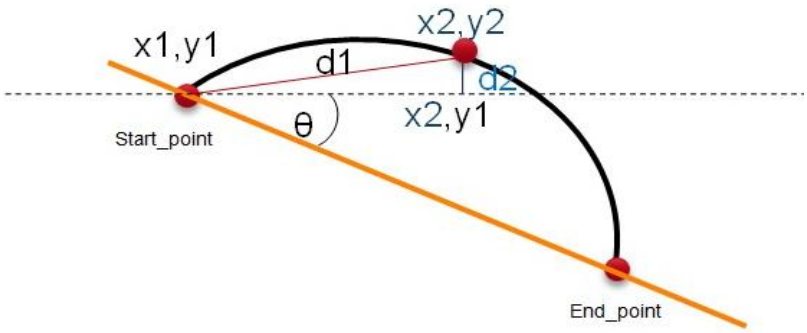
Segmen pertama :	Segmen kedua :
$d1^2 = (y2-y1)^2 + (x2-x1)^2$	$d1^2 = (y2-y1)^2 + (x2-x1)^2$
$= (50-73)^2 + (51-68)^2$	$= (43-50)^2 + (59-51)^2$
$= (-23)^2 + (-17)^2$	$= (-7)^2 + (8)^2$
$= \sqrt{529} + \sqrt{289}$	$= (49)^2 + (64)^2$
$= \sqrt{818}$	$= \sqrt{113}$
$= 28{,}6006 \approx 29$	$= 10{,}6301 \approx 11$

$$d2^2=(y2-y1)^2 \quad (4.7)$$

Dimana:

d2 merupakan garis deviasi yang membentuk sudut untuk menentukan derajat α.

x2,*y1* merupakan titik yang membentuk sudut perpotongan garis *d2*.



Gambar 4.20 Ilustrasi pencarian *d2*

Segmen pertama :	Segmen kedua :
$d2^2 = (y2-y1)^2$	$d2^2 = (y2-y1)^2$
$= (50-73)^2$	$= (43-50)^2$
$= (-23)^2$	$= (49)^2$
$= \sqrt{529}$	$= 7$
$= 23$	

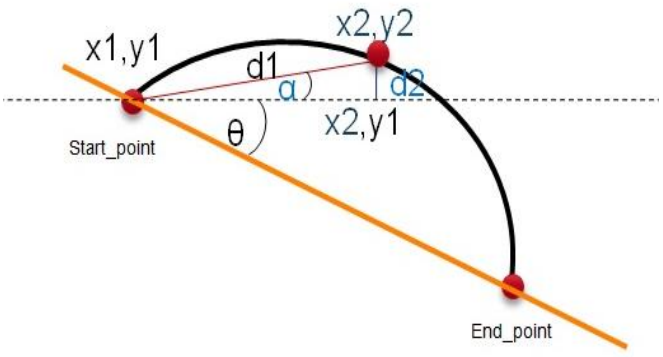
Sebelum ditemukannya apakah segmen berupa kurva atau *loop*, maka terlebih dahulu ditentukan sudut *alfa* (α). Sehingga nantinya akan ditemukan gradien (m) dan selisih dari gradien (m) dan d akan menentukan apakah segmentersebut berupa kurva atau *loop*.

$$\alpha = \arctan\left(\frac{d2}{\sqrt{d1^2 - d2^2}}\right) \tag{4.8}$$

Dimana:

α merupakan sudut antara *state i* terhadap *baseline.*

d1, d2 merupakan hasil dari garis perpotongan antara *state i* terhadap *base line.*

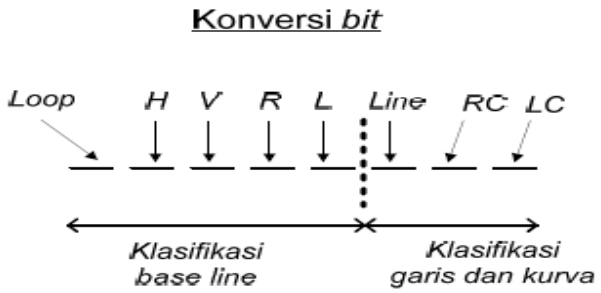


Gambar 4.21 ilustrasi pencarian α

Segmen pertama:	Segmen kedua :
α = arctan$\left(\frac{d2}{\sqrt{d1^2-d2^2}}\right)$ = arctan $\left(\frac{23}{\sqrt{29^2-23^2}}\right)$ = arctan $\left(\frac{23}{\sqrt{841-529}}\right)$ = arctan (1,3021) = 52,4760 ≈ 53	α = arctan$\left(\frac{d2}{\sqrt{d1^2-d2^2}}\right)$ = arctan $\left(\frac{7}{\sqrt{11^2-7^2}}\right)$ = arctan $\left(\frac{7}{\sqrt{121-49}}\right)$ = arctan 0.8249 = 39,5192 ≈ 40

Langkah 5:

Pada langkah ini akan dilakukan konversi dari tipe segmen (hasil klasifikasi segmen mulai dari langkah 1 hingga 5) menjadi nilai *bit* yang memiliki panjang 8 digit untuk setiap tipe segmen. Berikut aturan pengkonversian tersebut:



Gambar 4.22 Aturan Konversi *bit*
(Sumber : Gilewski, 1997)

Berdasarkan gambar di atas, dapat diketahui bahwa 5 digit pertama merupakan hasil klasifikasi dari *back line*, dan 3 digit terakhir adalah pengklasifikasian jenis kurva dan garis. Dari 5 digit pertama tadi hanya boleh ada satu digit yang memiliki nilai 1, dan posisinya akan

menunjukkan tipe yang manakah ia tergolong (*Loop, H, V, R, L*). Begitupula dengan 3 digit terakhir hanya boleh ada satu yang bernilai 1 serta posisinya akan menunjukkan tipe (*Line, RC, RL*). Berikut hasil konversi *bit* yang diharapkan pada potongan huruf yang telah diklasifikasi:

Pada segmen pertama dengan rasio nilai maksimum diantara keempat segmen FH, FV, FR, dan FL akan dapat ditentukan nilai maksimalnya.

Value = max (FH, max (FV, max(FR, FL)))

(4.9)

Setelah didapat nilai maksimal yaitu nilai FL sebesar 1,8222. Dengan demikian, nilai matrik yang di dapat yaitu 00001. Sedangkan untuk nilai matrik disebelah kanan dikarenakan hasil dari nailai FR dan FL bernilai nol (0), maka dengan otomatis nilai *baseline* bernilai 1 karena harus ada nilai 1 disebelah kanan. Dengan begitu nilai matrik yang didapat yaitu 100, sehingga nilai matrik keseluruhan adalah 00001100.

Seperti yang sudah diketahui apabila nilai $d <= m$, maka sudah bisa ditentukan nilai *loop*-nya yaitu berupa matrik yang bernilai 10000000. Apabila sudah ditemukan nilai *loop* pada segmen kedua, maka tidak perlu lagi dilakukan perhitungan seperti yang dilakukan perhitungan pada segmen pertama.

Langkah 6:

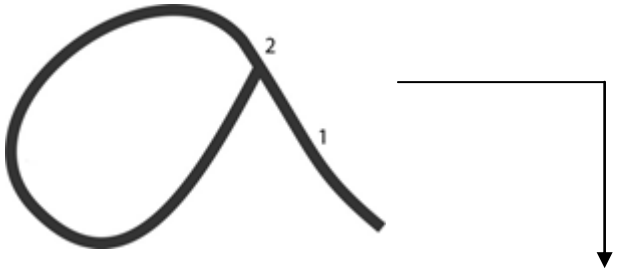
Ulangi langkah 1 hingga 6 pada semua segmen dari huruf yang akan dikenali. Simpan semua nilai konversi *bit* segmen tersebut kedalam sebuah matrik *Value* (matrik yang memiliki dimensi 2 x 1, jumlah baris

menunjukkan jumlah segmen yang dimiliki oleh sebuah huruf, dan jumlah kolom menunjukkan panjang digit dari nilai bit). Berikut nilai matrik *Value* setelah dilakukan konversi *bit*.

Tabel 4.3 Nilai matrik *value*

	Nilai Matrik *Value*							
	Loop	H	V	R	L	Line	RC	LC
Segmen pertama	0	0	0	0	1	1	0	0
Segmen kedua	1	0	0	0	0	0	0	0

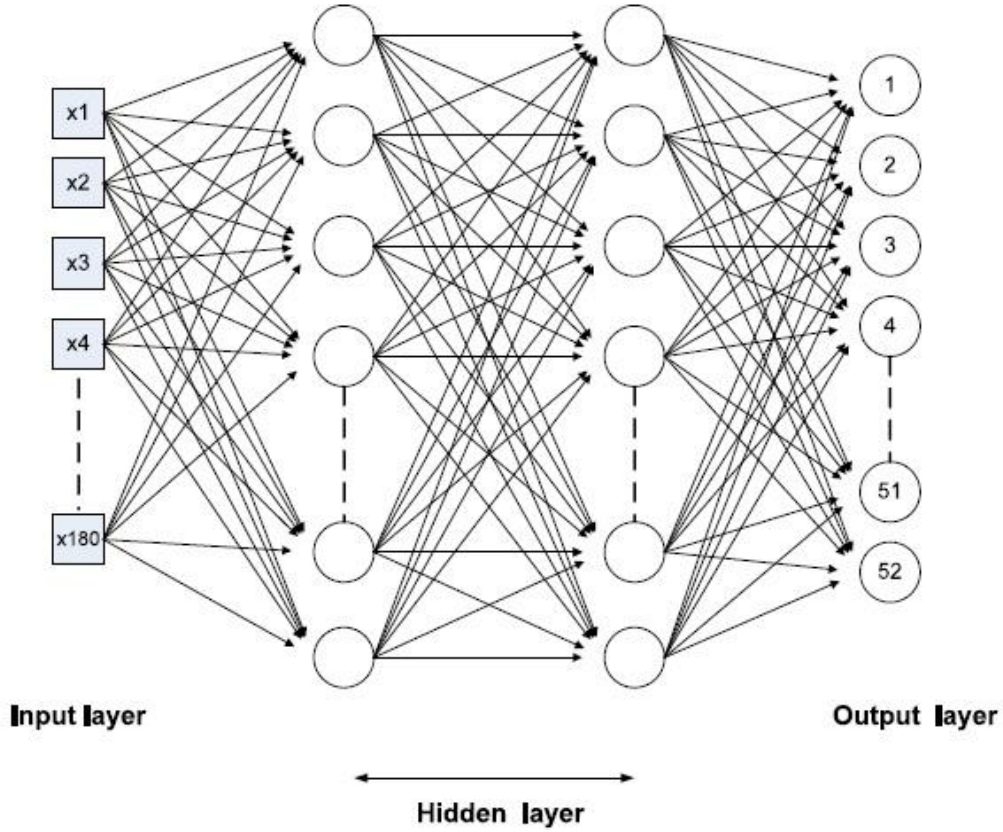
Untuk mengetahui keterhubungan antar segmen maka perlu kita definisikan menjadi sebuah matrik *interrelationship* (*matrik* berdimensi 10 x 10, yang menunjukkan ada maksimal 10 segmen dalam tiap hurufnya).



Tabel 4.4 Hasil perhitungan *interrelationship* dalam bentuk matrik 10 x 10

Segmen	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

Setelah matrik *interrelationship* dan *matrik Value* didapat, langkah berikutnya adalah mengubah ke dua *matrik* tersebut menjadi *matrik* yang hanya memiliki dimensi 1 x 80 (berasal dari *matrik Value*) dan 1 x 100 (berasal dari *matrik interrelationship*). Setelah itu gabung kedua *matrik* tersebut menjadi sebuah *matrik* baru (*matrik JST input*) yang nantinya akan menjadi *input* untuk proses *Radial Basis Function Network* (*matrik* ini berdimensi 1x 180).



Gambar 4.23 Struktur JST *Radial Basis Function* dengan 3 masukan

Input yang digunakan pada *Radial Basis Function* ini adalah JST*Input* (*matrik* 1 x 180) dan *output*-nya adalah *matrik Output* (52 x 1). Fungsi yang digunakan pada setiap *layer* ialah fungsi *radbas*, $f(x) = exp^{(-n \wedge 2)}$ yang nilai keluarannya antara 0 dan 1. Hasil perhitungan keseluruhan dari JST *Input* dengan metode *Radial Basis Function* menghasilkan matrik keluaran yang kemudian dieksekusi menjadi sebuah huruf yang diinginkan. Berikut ini matrik hasil dari perhitungan *Radial Basis Function*.

Berikut adalah potongan *source code* fungsi pemanggilan *output RBF*

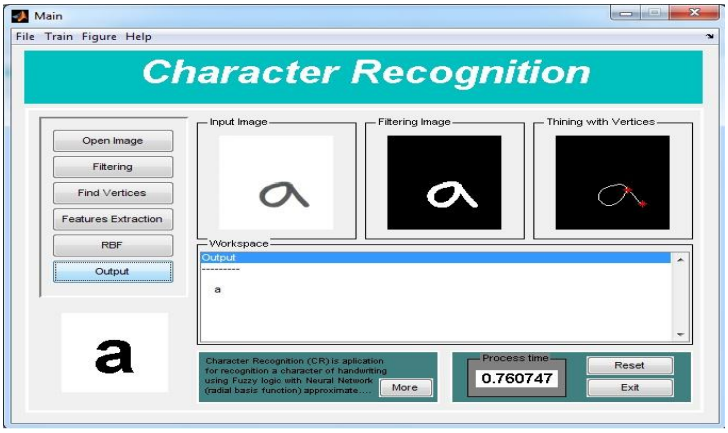
```
function NN_radialbasisfunction(JSTInput,)

RBF = newrb(minmax(JSTInput),
[180,180,1],{'radbas','gaussmf','purelin'},'spread','goal','trainr');
net.train
```

Berdasarkan fungsi pemanggilan *output Radial Basis Function di atas, maka* nilai *output* dari jaringan *Radial Basis Function* adalah 0 dan 1 seperti pada table 4.5.

Tabel 4.5 Nilai Output Radial Basis Function

Nilai Output Radial Basis Function			
0.00000	0.66667	0.00000	0.00000
0.00000	0.00000	0.33340	0.00000
0.00000	0.00066	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
0.00001	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000

Berikut hasil output nilai huruf yang diharapkan.



Gambar 4.24 Hasil *output* huruf yang diharapkan

4.3. Implementasi

4.3.1. Perancangan Perangkat Lunak

Perancangan perangkat lunak terdiri dari perancangan data dan perancangan antarmuka.

4.3.1.1. Perancangan Data

Perangkat lunak yang dikembangkan tidak menggunakan basis data melainkan matriks penyimpanan data. Matriks-matriks yang digunakan pada perangkat lunak pengenalan huruf tulisan tangan adalah sebagai berikut.

1. Matriks citra biner

   Matriks ini menyimpan data hasil konversi citra RGB ke dalam citra biner. Matriks terdiri dari 106 kolom dan 114 *pixel*. Matriks ini bernilai 0 dan 1.

2. Matriks *Filtering Image*

   Matriks ini menyimpan data dari hasil konversi biner ke dalam *filtering image*. Matriks terdiri dari 106 kolom dan 114 *pixel*. Matriks ini bernilai 0 dan 1.

3. Matriks *Find Vertices*

   Matriks ini menyimpan nilai koordinat *end points* dan nilai koordinat *Intersection Points* yang terbentuk setelah dikonversi menggunakan aturan *bit*. Matriks ini berukuran 2 x1 kolom.

4. Matriks *Value*

   Matriks ini menyimpan nilai keterhubungan antar satu segmen dengan segmen dengan segmen lainnya. Matriks akan bernilai 1 jika suatu segmen terhubung dengan segmen yang lainnya dan 0 jika tidak terhubung.

5. Matriks *Interrelationship*

   Matriks ini menyimpan nilai keterhubungan antar satu segmen dengan segmen lainnya. Matriks akan bernilai 1 jika suatu segmen terhubung dengan segmen lainnya dan 0 jika tidak terhubung. Matriks ini berukuran *n* baris *x n* kolom, dimana *n* adalah jumlah maksimal segmen yang terbentuk dari suatu huruf.

6. Matriks *input* JST

   Matriks ini merupakan akumulasi dari penjumlahan matriks *value* dan matriks *interrelationship*. Matriks ini memiliki ukuran *n x* 1 ukuran matriks, dimana n adalah hasil akumulasi dari matriks *value* dan matriks *interrelationship*, n adalah 180. Matriks ini digunakan sebagai *input* untuk proses pengujian atau pengenalan dengan algoritma Jaringan Syaraf Tiruan *Radial Basis Function*.

7. Matriks JST_Target

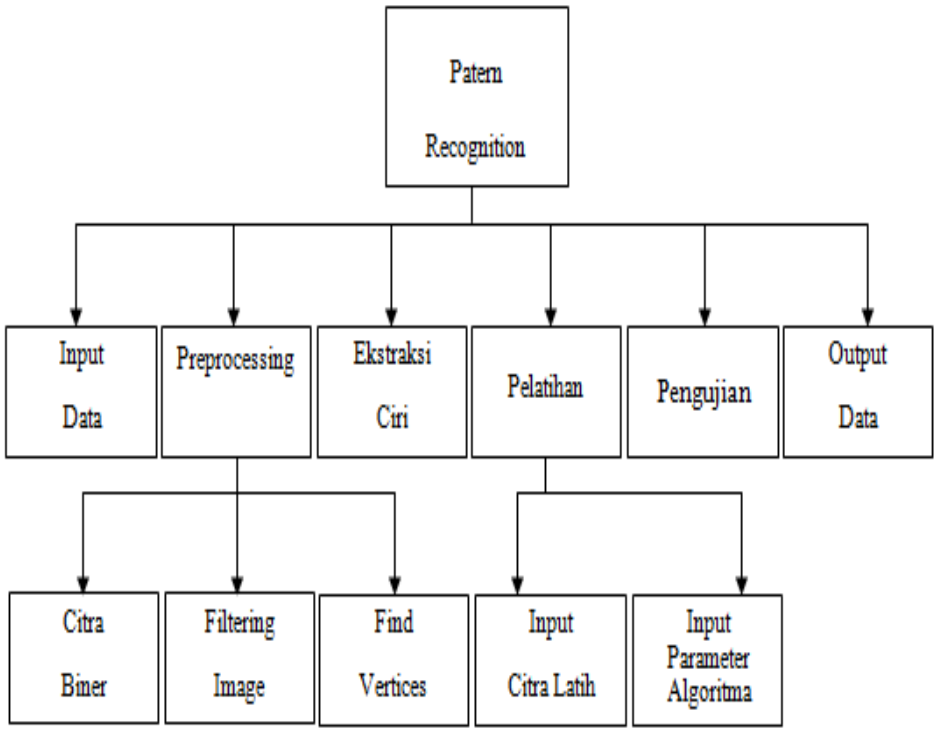
   Matriks ini menyimpan nilai dari target suatu huruf. Matriks ini bernilai 0.0180 sampai 0.9360.

8. Matriks JST_Radial Basis Function

   Matriks ini menyimpan *file* hasil dari model pelatihan untuk digunakan sebagai acuan pada proses pengenalan atau pengujian.

4.3.1.2. Perancangan Antarmuka Perangkat Lunak

Perangkat lunak Pengenalan Huruf Tulisan Tangan memiliki beberapa Pengguna. *Dialog Chart* pada gambar 4.17 di bawah ini menujukkan menu-menu yang dimiliki oleh perangkat lunak ini.



Gambar 4.25 Dialog Chart

Antarmuka yang dibangun menggunakan properti yang terdapat dalam GUIDE Matlab. Tabel 4.6 pada halaman berikut ini memuat properti-properti yang digunakan pada implementasi antarmuka Pengenalan Huruf Tulisan Tangan.

Tabel 4.6 Properti Antarmuka

Nama Properti	Keterangan Penggunaan Properti
MPanel	*Panel* yang menampung objek untuk membuka *file*, menampung objek untuk proses pengenalan, menampung objek untuk menampilkan hasil pembangunan pengetahuan
Mbutton	Tombol untuk menampilkan proses

	yang terjadi pada properti Maxes, Mtext, Mlistbox
Maxes	Area untuk menampilkan hasil praproses
Medittext	Kotak untuk memasukkan suatu nilai
MListbox	Area untuk menampilkan hasil ekstraksi ciri dan perhitungan menggunakan algoritma *Radial Basis Function*
Mtext	Area untk menampilkan hasil keluaran dan waktu proses pengenalan
Mpopupmenu	Kotak untuk menampilkan pilihan secara *dropdown*

BAB V

HASIL DAN PEMBAHASAN

Dari hasil analisis dan perancangan yang telah dilakukan, tahapan selanjutnya dalam membangun perangkat lunak adalah dengan cara mengimplementasikan hasil dan perancangan tersebut.

5.1. Implementasi Data

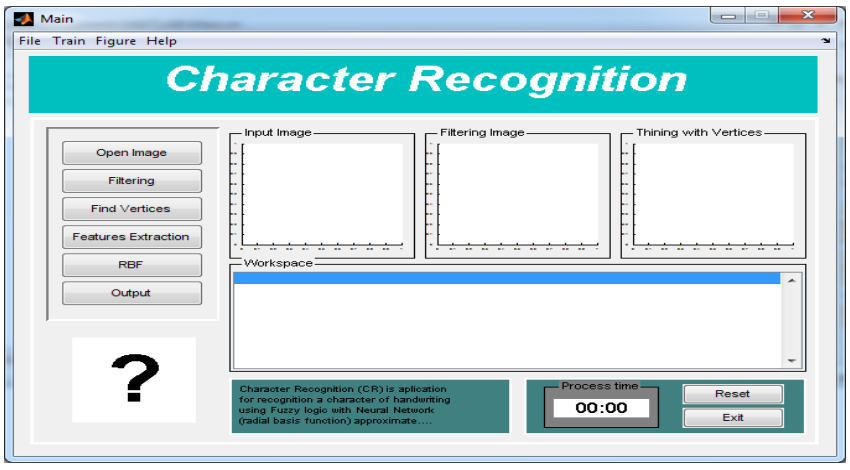
Aplikasi yang dikembangkan tidak menggunakan *database*, sehingga bagian ini hanya menjelaskan mengenai implementasi data yang perhitungannya menggunakan matriks.

5.2. Implementasi Modul Program

Dari hasil analisis perancangan yang telah dilakukan, dibuat bentuk *coding* dengan cara mengimplementasikan hasil analisis dan perancangan tersebut ke dalam bahasa pemrograman matlab.

5.3. Implementasi Antarmuka

Implementasi antarmuka yang dibangun dapat dilihat pada gambar 5.1 di bawah ini :



Gambar 5.1 Implementasi Perangkat Lunak

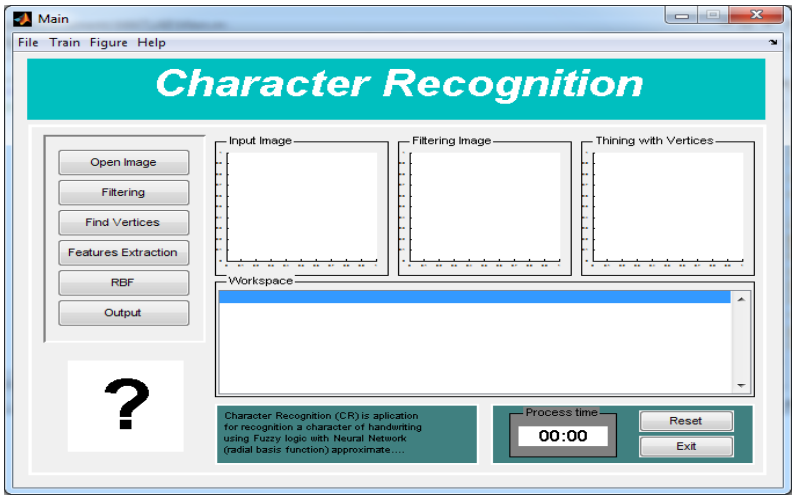
5.4. Pengujian

Setelah melakukan langkah analisis, perancangan, dan implementasi, perangkat lunak Pengenalan Huruf Tulisan Tangan, maka langka selanjutnya adalah pengujian. Pengujian dilakukan untuk mengetahui akurasi penerapan dari algoritma yang digunakan dalam perangkat lunak ini. Skenario pengujian pada penelitian ini adalah sebagai berikut :

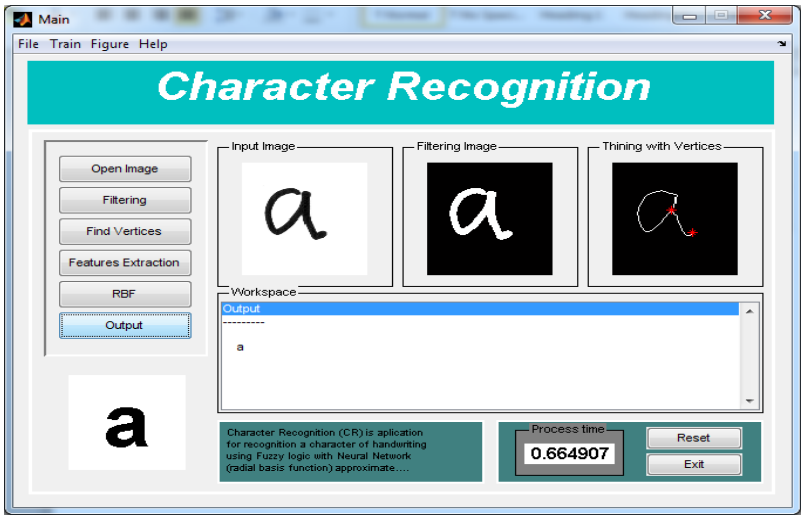
1. Memeriksa kesesuaian antara implementasi perangkat lunak dengan hasil analisis dan perancangan perangkat lunak.
2. Mengetahui akurasi penerapan *Fuzzy Feature Extraction* pada proses ekstraksi ciri suatu huruf.
3. Mengetahui akurasi penerapan algoritma Jaringan Syaraf Tiruan *Radial Basis Function* pada proses pelatihan.
4. Mengetahui akurasi penerapan algoritma Jaringan Syaraf Tiruan *Radial Basis Function* pada proses pengujian dan faktor-faktor yang mempengaruhi nilai akurasi.
5. Mengetahui waktu proses pengenalan huruf tulisan tangan.

5.4.1. Kesesuaian Implementasi Perangkat Lunak dengan Hasil Analisis dan Perancangan Perangkat Lunak

Di bawah ini adalah gambar implementasi Perangkat Lunak Pengenalan Huruf Tulisan Tangan Menggunakan *Fuzzy Feature extraction* Dengan Pendekatan *Radial Basis Function Neural Network.*



Gambar 5.2 Tampilan Utama Perangkat Lunak



Gambar 5.3 Tampilan Hasil Pengenalan Huruf

Berdasarkan gambar 5.3 didapatlah data uji sesuai dengan huruf yang diharapkan.

5.4.2 Analisis Hasil Pengujian Penerapan Algoritma *Fuzzy Feature Extraction*

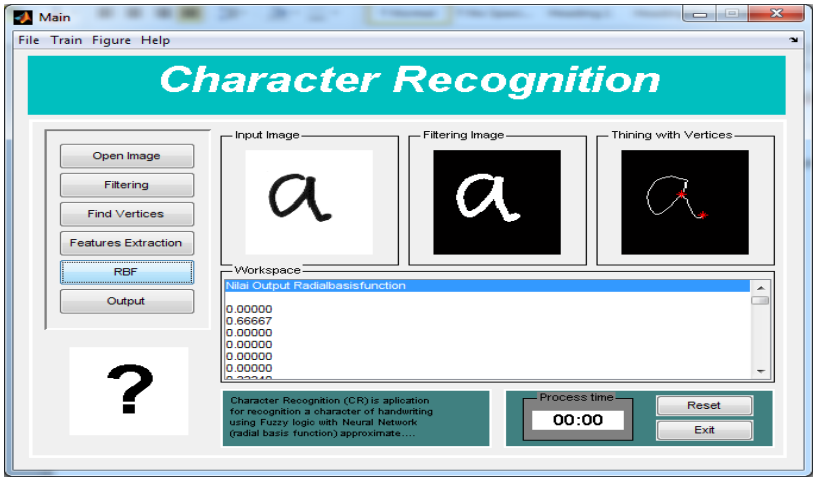
Perbandingan hasil penerapan algoritma ekstraksi ciri ini dilakukan terhadap subset huruf tulisan tangan, yaitu 520 sampel dimana masing-masing huruf memiliki 10 pola. Proses ekstraksi ciri (*feature extraction*) ideal dilakukan terhadap huruf dalam bentuk *hand printed* yang telah

diubah menjadi bentuk citra yang berukuran sama dengan dimensi citra pada sampel huruf tulisan tangan. Bentuk huruf ideal yang dipilih adalah *font Times New Roman*. *Font* ini dipilih karena bentuknya yang umum diterapkan. Sehingga, dengan menggunakan *font* ini, perbandingan yang dilakukan cukup sepadan.

Proses ekstraksi ciri pada citra ideal ini tidak dilakukan secara manual, tetapi mengikuti prosedur yang ada pada perangkat lunak sampai tahapan ekstraksi ciri, hal ini bertujuan untuk mengetahui tipe segmen yang terbentuk dari silabel huruf citra ideal tersebut, hasilnya akan dibandingkan dengan sampel tulisan pada salah satu responden.

5.4.3 Analisis Hasil Pengujian Penerapan Algoritma Jaringan Syaraf Tiruan *Radial Basis Function*

Hasil perhitungan keseluruhan dari JST *Input* dengan metode *Radial Basis Function* menghasilkan matrik keluaran yang kemudian dieksekusi menjadi sebuah huruf yang diinginkan. Berikut ini gambar matrik hasil dari perhitungan *Radial Basis Function* pada Perangkat Lunak Pengenalan Huruf Tulisan Tangan Menggunakan *Fuzzy Feature extraction* Dengan Pendekatan *Radial Basis Function Neural Network*.



Gambar 5.4 Tampilan Hasil Perhitungan *Radial Basis Function*

5.4.4 Analisis Hasil Akurasi dari Model Pelatihan yang Dibangun

Proses pembangunan model pelatihan dibuat dengan menggunakan *Neural Network Toolbox* (*nntool*) yang terdapat pada Matlab. Skenario pengujian dilakukan sebagai berikut :

Data eksekusi yang digunakan sebanyak 520 citra huruf yang berasal dari 10 (sepuluh) responden, masing-masing responden menuliskan 52 huruf besar dan huruf kecil. Pengujian dilakukan dengan melatih 260 citra latih untuk mendapatkan model huruf yang menjadi target dalam pengenalannya. Kemudian data yang telah dilatih tersebut dijadikan sebagai acuan untuk mengenali huruf yang belum dilatih. Hasil dari analisis akurasi pengujian dapat disajikan dalam tabel berikut.

Tabel 5.1 Data Hasil Recognition Rate Huruf Besar

Input Huruf	Output Huruf Besar										Huruf yang Dikenali	Total Sampel	Recognition Rate
	1	2	3	4	5	6	7	8	9	10			
A	A	A	A	A	A	-	A	-	A	-	7	10	70%
B	B	B	B	B	B	-	-	B	B	-	7	10	70%
C	C	C	C	-	C	C	C	C	C	C	9	10	90%
D	-	D	D	D	D	D	D	-	-	-	6	10	60%
E	E	E	E	E	E	E	E	E	-	E	9	10	90%
F	F	F	F	-	F	F	F	-	-	-	6	10	60%
G	G	-	G	G	-	G	-	G	G	G	7	10	70%
H	H	H	H	H	H	H	-	H	H	H	9	10	90%
I	I	I	-	I	I	-	I	I	-	I	7	10	70%
J	J	J	J	J	J	J	J	-	J	J	9	10	90%
K	K	K	-	-	K	K	-	-	-	-	4	10	40%
L	L	L	L	L	L	L	L	L	L	L	10	10	100%
M	-	M	M	-	M	-	-	M	M	M	6	10	60%
N	N	N	N	N	N	N	N	N	N	N	10	10	100%
O	-	-	O	-	O	-	-	-	-	-	2	10	20%
P	-	-	-	-	-	-	P	-	-	-	1	10	10%
Q	-	Q	Q	Q	Q	Q	Q	Q	-	-	7	10	70%
R	R	R	R	R	R	R	R	R	-	-	8	10	80%
S	S	S	S	S	S	S	S	S	S	S	10	10	100%
T	T	T	T	T	T	T	T	T	T	T	10	10	100%
U	-	-	-	U	U	-	-	-	-	-	2	10	20%
V	V	-	V	-	-	-	V	-	V	-	4	10	40%
W	W	W	W	-	-	W	-	-	-	W	5	10	50%
X	X	X	X	X	X	X	X	X	X	X	10	10	100%
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	10	10	100%
Z	Z	Z	-	-	-	-	-	-	-	-	2	10	20%
Jumlah													68.076%

Dari tabel 5.1 didapatlah rata-rata pengenalan huruf besar sebesar 68,076% yang berasal dari 10 (sepuluh) responden, masing-masing responden menuliskan 26 huruf besar.

Tabel 5.2 Data Huruf Besar dalam Jumlah Waktu (detik)

Input Huruf	Output Huruf Besar										Total	Rata-Rata Waktu Huruf Tiap (detik)
	1	2	3	4	5	6	7	8	9	10		
A	0.9154	0.9375	1.0858	1.0069	0.6756	1.1327	0.8543	0.9795	0.7658	0.875	9.2285	0.92285
B	1.0284	1.0777	1.1965	1.0396	1.1182	1.3284	1.128	0.9698	0.8704	1.0692	10.8262	1.08262
C	0.7291	0.8721	0.9352	0.8794	0.7476	0.8154	0.8006	0.7517	0.7418	0.8442	8.1171	0.81171
D	0.8774	1.0299	0.7467	1.2148	0.8804	1.0973	0.9833	0.8617	0.8634	1.1446	9.6995	0.96995
E	0.8039	1.0447	0.9627	1.1525	0.9078	0.8979	1.0559	0.9129	0.8068	0.8951	9.4402	0.94402
F	0.8713	0.9847	0.9442	0.9422	0.8637	0.7772	0.9342	0.8092	0.5843	0.8705	8.5815	0.85815
G	0.7881	1.0593	1.2677	1.0271	0.9387	1.0669	0.8548	0.8678	0.9577	1.0581	9.8862	0.98862
H	0.8359	1.0068	0.8552	1.0139	0.854	0.9663	0.6325	0.8636	0.8307	0.5131	8.372	0.8372
I	0.6116	0.6362	0.4521	0.6244	0.4264	0.5895	0.5959	0.5994	0.5744	0.6246	5.7345	0.57345
J	0.7666	0.9063	0.7883	0.9667	0.763	0.8206	0.7934	0.8814	0.7987	0.8118	8.2968	0.82968
K	0.5461	1.0054	0.88	1.0175	0.8399	0.9049	0.8968	0.8743	0.4229	0.8096	7.2284	0.81974
L	0.4932	0.8639	0.7946	0.7905	0.7233	0.7546	0.6917	0.7462	0.6978	0.7461	7.3019	0.73019
M	0.8249	0.3958	0.9583	1.1704	0.9725	0.9699	0.8457	1.0606	1.0251	1.0202	9.2434	0.92434
N	0.7886	1.0126	0.8626	1.0381	0.8317	0.9802	0.8836	1.0037	0.7279	0.8238	8.9528	0.89528
O	0.7924	0.9357	0.8857	0.8764	0.6766	0.7204	0.9527	0.9379	0.6927	0.8114	8.2819	0.82819
P	0.8433	1.0055	0.9101	1.0709	0.7622	0.8313	0.7546	0.9432	0.8321	0.8759	8.8291	0.88291
Q	1.0042	1.1189	0.0694	1.1382	0.8764	1.1312	0.9659	1.1054	0.9569	0.9683	9.3348	0.93348
R	0.7986	1.1774	1.1311	1.1059	0.8536	0.984	0.8213	1.0096	0.9489	0.8505	9.6809	0.96809
S	0.7755	1.0299	0.9917	1.0158	0.9095	0.8727	0.8307	0.7697	0.6565	0.8493	8.7013	0.87013
T	0.7276	0.8673	0.8539	0.8262	0.8424	0.7865	0.917	0.8547	0.7888	0.7093	8.1737	0.81737
U	0.822	0.9082	0.853	1.0289	0.8661	0.9659	0.8283	0.8046	0.6348	0.8674	8.5792	0.85792
V	0.7133	0.803	0.8041	0.9353	0.6547	0.7998	0.7213	0.7294	0.657	0.6978	7.5157	0.75157
W	0.6831	1.1792	0.7581	1.1151	0.9344	1.0388	0.966	0.8396	0.7221	0.8876	9.124	0.9124
X	0.6723	0.8206	0.8257	0.8456	0.6903	0.6965	0.7573	0.779	0.7259	0.7674	7.5806	0.75806
Y	0.7038	0.7856	0.7615	0.7713	0.7867	0.7243	0.6828	0.7389	0.6749	0.6949	7.3247	0.73247
Z	0.8615	0.9818	1.0929	1.0935	0.9267	0.8104	0.8839	1.101	0.8576	1.0198	9.6291	0.96291
Jumlah Rata-rata Waktu (detik)												0.86397

Dari table 5.2 didapatlah jumlah rata-rata waktu pengenalan huruf besar selama 0,86397 per detik.

Tabel 5.3 Data Hasil Recognition Rate Huruf Kecil

Input Huruf	Output Huruf Kecil										Huruf yang Dikenali	Total Sampel	Recognition Rate
	1	2	3	4	5	6	7	8	9	10			
A	a	a	a	a	a	a	a	a	a	-	9	10	90%
B	b	b	b	b	b	b	b	b	b	b	10	10	100%
C	c	-	c	c	c	-	c	c	c	c	8	10	80%
D	-	d	d	d	d	d	-	-	d	d	7	10	70%
E	e	e	e	e	e	e	e	e	e	e	10	10	100%
F	f	-	f	f	-	f	f	f	f	-	7	10	70%
G	g	g	g	g	g	g	g	g	g	g	10	10	100%
H	h	h	h	h	h	h	h	h	h	h	10	10	100%
I	i	i	i	i	i	i	-	i	-	i	8	10	80%
J	j	j	j	j	j	j	j	j	j	j	10	10	100%
K	k	k	k	k	k	k	k	k	k	k	10	10	100%
L	l	l	l	l	l	l	l	l	l	l	10	10	100%
M	m	m	m	m	m	m	m	m	m	m	10	10	100%
N	n	n	n	n	n	n	n	n	n	n	10	10	100%
O	o	o	o	o	o	o	o	o	o	o	10	10	100%
P	p	-	-	-	-	-	-	-	-	-	1	10	10%
Q	q	q	-	-	-	q	-	q	q	q	6	10	60%
R	r	r	r	r	r	r	r	r	r	r	10	10	100%
S	s	s	s	s	s	s	s	s	s	s	10	10	100%
T	t	t	t	t	t	t	t	t	t	t	10	10	100%
U	u	u	u	u	u	u	u	u	u	u	10	10	100%
V	v	v	v	v	v	v	v	v	v	v	10	10	100%
W	w	w	w	w	w	w	w	w	w	w	10	10	100%
X	x	x	x	x	x	-	-	-	-	-	5	10	50%
Y	y	y	y	y	y	y	y	y	y	-	9	10	90%
Z	z	z	z	z	-	-	-	-	-	-	4	10	40%
Jumlah													86.153%

Dari table 5.3 didapatlah rata-rata pengenalan huruf kecil sebesar 86,153% yang berasal dari 10 (sepuluh) responden, masing-masing responden menuliskan 26 huruf kecil.

Tabel 5.4 Data Huruf Kecil dalam Jumlah Waktu (detik)

Input Huruf	Output Huruf Kecil										Total	Rata-rata waktu huruf tiap (detik)
	1	2	3	4	5	6	7	8	9	10		
a	0,7904	0,6798	0,9355	0,7748	0,8209	0,8894	0,8733	0,8666	0,9419	0,8681	8,4407	0,84407
b	0,6842	0,7559	0,9056	0,7396	0,755	0,7715	0,7755	0,6727	0,8384	0,7805	7,6789	0,76789
c	0,6085	0,6359	0,7972	0,8649	0,7418	0,6228	0,6454	0,6721	0,6919	0,6159	6,8964	0,68964
d	0,795	0,6692	0,9589	0,6526	0,7405	0,727	0,8348	0,8477	0,8116	0,8251	7,8624	0,78624
e	0,7411	0.5632	0,9316	0,7471	0,809	0,8028	0,6871	0,8351	0,9116	0,7224	7,751	0,7751
f	0,6655	0.8762	0.7681	0,6854	0.6543	0,7281	0.7812	0,7355	0.8125	0.8764	7,5832	0,75832
g	0,7694	0,7171	0,4831	0,6664	0,7978	0,8136	0,7858	0,6777	0,8911	0,7783	7,3803	0,73803
h	0,6416	0,6105	0,7218	0,6161	0,7162	0,7059	0,6704	0,6743	0,7023	0,6982	6,7573	0,67573
i	0,5055	0,5063	0,5899	0,4885	0,6135	0,5175	0,5012	0,513	0,5139	0,5146	5,2639	0,52639
j	0,6057	0,546	0,675	0,5946	0,8543	0,6683	0,8831	0,6154	0,8564	0,624	6,9228	0,69228
k	0,6678	0,6871	0,6957	0,6871	0.7321	0,704	0,5321	0,6978	0,705	0,6609	6,7696	0,67696
l	0,5437	0,5724	0,7049	0,5457	0,5276	0,5778	0,5447	0,5963	0,8351	0,5237	5,9719	0,59719
m	0,7316	0,6704	0,8704	0,7284	0,8866	0,6679	0,7809	0,6796	0,7782	0,7243	6,1195	0,61195
n	0,6738	0,6544	0,7827	0,6329	0,6826	0,6525	0,6605	0,6967	0,6793	0,6628	6,7782	0,67782
o	0,6249	0,5656	0,6804	0,5703	0,7803	0,6023	0,5678	0,6224	0,6758	0,6178	6,3076	0,63076
p	0,6792	0,7451	0,8123	0,6573	0,8564	0,7195	0,6514	0,6878	0,7561	0,7809	7,346	0,7346
q	0,7975	0,6671	0,8334	0,6694	0,6996	0,7319	0,7746	0,8831	0,8104	0,7519	7,6189	0,76189
r	0,5708	0,5442	0,7981	0,549	0,5654	0,5857	0,4531	0,6031	0,5694	0,6644	5,9032	0,59032
s	0,8598	0,6557	0,7396	0,6425	0,5589	0,6142	0,5819	0,5971	0,6865	0,6196	6,5558	0,65558
t	0,657	0,5676	0,7475	0,6289	0,6505	0,6543	0,7047	0,8321	0,4321	0,7658	6,6405	0,66405
u	0,6694	0,8913	0,8119	0,5944	0,6781	0,5909	0,6267	0,5652	0,7799	0,6294	6,8372	0,68372
v	0,5544	0,5442	0,7619	0,5734	0,5504	0,5697	0,6159	0,5938	0,6636	0,5651	5,9924	0,59924
w	0,7191	0,6406	0,8969	0,714	0,5436	0,5871	0,7008	0,7931	0,7695	0,7268	7,0915	0,70915
x	0,6161	0,5722	0,7239	0,5793	0,5784	0,5858	0,609	0,6127	0,6764	0,6037	6,1575	0,61575
y	0,7539	0,7295	0,9202	0,5643	0,6321	0,8159	0,7771	0,7417	0,732	0,7321	7,3988	0,73988
z	0,6783	0,5888	0,966	0,6483	0,749	0,6255	0,6599	0,7109	0,7596	0,7592	7,1455	0,71455
Jumlah Rata-rata Waktu (detik)												0,68911

Dari table 5.4 didapatlah jumlah rata-rata waktu pengenalan huruf kecil selama 0,68911 per detik.

Berdasarkan hasil pengujian diatas, dapat diketahui bahwa proses pengenalan huruf besar dan kecil memiliki *recognition rate* sebesar 77.1145 %. Hasil yang didapat tergolong kedalam kategori baik untuk melakukan pengenalan huruf tulisan tangan yang berada dalam rentang (75%-84%) . Ada beberapa masalah yang bisa ditemui dalam proses pengenalan huruf tulisan tangan ini antara lain:

1. Pola yang dimiliki oleh setiap huruf tidak selalu identik, dengan kata lain diperlukan sebuah parameter baru (selain *value* huruf dan matrik

interrelationship) yang diperlukan untuk menjamin setiap huruf memiliki pola yang identik.

2. Penggunaan Jaringan Syaraf Tiruan akan lebih efektif jika pola yang dimiliki pada saat proses pembelajaran (*training*) semakin banyak dan begitupula sebaliknya akan kurang efektif apabila pola yang dimiliki terbatas.
3. Munculnya "kail" pada ujung suatu huruf yang ditulis responden sehingga jumlah segmen yang terbentuk menjadi tidak sama dengan citra ideal, karena adanya penambahan segmen.
4. Proses pengambilan citra ketika gambar tersebut di-*edit* ke dalam sebuah gambar yang sesuai dengan syarat yang telah ditetapkan harus benar-benar rapi dan bersih. Ini untuk membantu proses ekstraksi ciri sehingga hasil *output* yang diinginkan dapat berhasil.

Dari tabel 5.4 dapat disimpulkan bahwa waktu proses rata-rata dihitung hanya dari proses *Radial Basis Function* dalam mengenali setiap huruf besar dan kecil. Sehingga didapat waktu proses rata-rata dalam mengenali setiap huruf besar dan kecil adalah 0.77654 detik. Sehingga dapat disimpulkan pula bahwa waktu proses rata untuk mengenali huruf besar dan kecil cukup baik meskipun *output* yang dihasilkan kurang memuaskan. Beberapa hal yang dapat mempengaruhi lama tidaknya waktu proses:

1. Ukuran gambar (*image*) *input*-an akan mempengaruhi kecepatan pemrosesan, yakni semakin kecil ukuran *image input* akan semakin cepat waktu yang diperlukan, begitupula sebaliknya semakin besar ukuran *image input* maka semakin lama waktu proses yang diperlukan.

2. Jumlah segmen pada tiap huruf akan mempengaruhi waktu proses, yakni semakin banyak jumlah segmen maka akan semakin lama waktu proses dan demikian pula sebaliknya, semakin sedikit jumlah segmen maka akan semakin cepat waktu proses.

BAB VI

PENUTUP

6.1. Kesimpulan

Kesimpulan yang dapat diambil dari proses penelitian yang dilakukan adalah sebagai berikut :

1. Metode *Fuzzy Feature Extraction* dengan pendekatan Jaringan Syaraf Tiruan RBF (*Radial Basis Function*) dapat digunakan untuk aplikasi pengenalan huruf tulisan tangan.
2. Sistem *Pattern Recognition* yang dibangun dengan menggunakan metode *Fuzzy Feature Extraction* dan Jaringan Syaraf Tiruan *Radial Basis Function* memiliki *recognition rate* sebesar 77.1145 %, dengan waktu proses yang diperlukan rata-rata sebesar 0.77654 detik untuk tiap huruf. Dengan data training : data uji = 260 : 520 = 1 : 2 .
3. Besarnya *recognition rate* akan dipengaruhi oleh identik tidaknya pola yang dimiliki oleh tiap huruf yang akan dikenali.
4. Waktu proses akan dipengaruhi setidaknya oleh dua faktor yakni : ukuran *huruf*, dan jumlah segmen yang dimiliki oleh huruf yang akan dikenali.

6.2. Saran

Berikut saran yang penulis ajukan guna pengembangan pembangunan sistem penegenalan huruf tulisan tangan menggunakan metode *Fuzzy Feature Extraction* dengan pendekatan Jaringan Syaraf Tiruan *Radial Basis Function* :

1. Sistem akan lebih bermanfaat bila mampu mengenali bukan hanya satu huruf saja, melainkan kata ataupun kalimat sehingga aplikasi ini dapat lebih bermanfaat.
2. Pola yang akan dikenali sebaiknya dalam bentuk huruf dengan titik-titik penghubungnya harus terhubung sempurna. Jika tidak, akan mengarah dengan dikenalinya sebagai huruf lain.
3. Sistem akan lebih baik bila mampu menangani *noise* secara langsung.

DAFTAR PUSTAKA

Amin. (1996). Hand printed Arabic Character Recognition System Using an Artificial Neural Networks.

Gilewski, Phillips, P., Yanushkevich, & Popel. (1997). Education Aspect: Handwriting recognition-Neural Network-Fuzzy Logic. In *Proceedings of the IAPR International Conference On Pattern Recognition And Information Processing-PRIP'97* (pp. 39-47). vol. 1.

Gonzalez, R., & Woods, R. (2002). Digital image processing (2nd ed.). Prentice-Hall Inc.

Gorgel, P., & Oguzhan, O. (2007). Handwritten Character Recognition System Using Artificial Neural Networks. In *Jurnal of Electrical and Electronics Engineering.*

Jaeger, S. (2003). The State of Art in Japanese Online Handwriting Recognition Compared to Techniques in Western Handwriting Recognition.

K.C, S. (2009). A Comperhansive Survey On On-Line Handwriting Recognition Technology And Its Real Application To The Nepalese Natural Handwriting. *Kathmandu University Journal of Science Engineering and Technology*.

Kusumadewi, S. (2002). Analisis & Desain Sistem. Yogyakarta: Graha Ilmu.

Kusumadewi, S. (2003). In *Artificial Intelligence (Teknik dan Aplikasinya).* Jakarta: Graha Ilmu.

Kusumoputro, B., & Emanuel, P. (2001). Pengenalan Huruf Tulisan Tangan Menggunakan Ekstraksi Ciri Berbasis Fuzzy dan Jaringan Syaraf Tiruan. In *Jurnal Komputer dan Teknologi Informas.*

Lee, W. (1996). Off-Line Recognition Of Totally Unconstrained Handwritten Numerical Using Multilayer Cluster Neural Network. In *Jurnal IEEE Transaction On Pattern Analysis And Machine Intelligence.*

Martin, J. (1990). Information Engineering. In *Book II : Planning and analysis.* London.

Pavlidis, T. (1992). Algorithm for Graphics and Image. Addison-Wessley.

Santosh. (2009). A Comperhansive Survey On On-Line Handwriting Recognition Technology And Its Real Application To The Nepalese Natural

Handwriting. *Kathmandu University Journal of Science Engineering and Technology*.

Santosh, K. (2009). A Comperhansive Survey On On-Line Handwriting Recognition Technology And Its Real Application To The Nepalese Natural Handwriting. Kathmandu University Journal of Science Engineering and Technology.

Setiawan, W., & Sri, A. (2005). Aplikasi Jaringan Syaraf Tiruan Perambatan Balik Pada Pengenalan Angka Tulisan Tangan. Staff Pengajar Teknik Elektro Universitas Udayana dan Staff Teknik Elektro Politeknik Negeri Bali Kampus Bukit Jimbaran Bali.

Siregar, D. M. (2009). Pengenalan Huruf Tulisan Tangan dengan Metode Ekstraksi Ciri Windowing Menggunakan Jaringan Syaraf Tiruan Backpropagation. Fakultas Teknik Universitas Andalas.

Sugiyono. (2009). Metode Penelitian Pendidikan, Pendekatan Kuantitatif, Kualitatif, dan. Bandung: Alfabeta.

Suyanto. (2007). Artificial Intelligence : Searching, Reasoning, dan Learning. Bandung.

LAMPIRAN

LAMPIRAN A

Lampiran A-1 Source Code

Main.m :

```
function varargout = Main(varargin)
% MAIN M-file for Main.fig
%      MAIN, by itself, creates a new MAIN or raises the existing
%      singleton*.
%
%      H = MAIN returns the handle to a new MAIN or the handle to
%      the existing singleton*.
%
%      MAIN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAIN.M with the given input arguments.
%
%      MAIN('Property','Value',...) creates a new MAIN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Main_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Main_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Main

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Main_OpeningFcn, ...
                   'gui_OutputFcn',  @Main_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Main is made visible.
function Main_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no edit2 args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Main (see VARARGIN)

% Choose default command line edit2 for Main
handles.edit2 = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Main wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Main_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning edit2 args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line edit2 from handles structure
varargout{1} = handles.edit2;



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox2


% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Open.
function Open_Callback(hObject, eventdata, handles)
% hObject    handle to Open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
global namafile panjang lebar filename pathname;
[filename, pathname] = uigetfile({'*.bmp';'*.jpg';'*.gif';'*.*'}, 'Pick an Image
File');
 if filename~=0
   S = imread([pathname,filename]);
   axes(handles.axes1);
   imshow(S);
   handles.S = S;
  % guidata(hObject, handles);

   dimensi=size(S);
```

```
   panjang = dimensi(1,2);
   lebar = dimensi (1,1);
   namafile=filename ;

 end


set(handles.listbox2,'String',{['Nama File Input = ' filename ] ['Ukuran Image    = '
num2str(panjang) 'x' num2str(lebar)]});



% --- Executes on button press in Filter.
function Filter_Callback(hObject, eventdata, handles)
% hObject    handle to Filter (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
global namafile lebar panjang FilterImg2 filename pathname;
if filename==0
   open('Exception2.fig');
else
       I = imread([pathname,filename]);
       dim = size(namafile);        % size of input image
       leng = length(dim);          % Change input image to gray scale

       if leng == 3
          I = rgb2gray(I);
       end
       imgBW = im2bw(I, 0.90455);      %% Binary Image
       FilterImg = medfilt2(imgBW,[3 1]);
       for i = 1 : lebar
             for j = 1 : panjang
                if FilterImg(i,j) == 1
                  FilterImg(i,j) = 0;
                else
                  FilterImg(i,j) = 1;
                end
             end
       end

       %  figure,imshow(FilterImg);
        FilterImg2=flipud(FilterImg);
        axes(handles.axes2);
        imshow(FilterImg);
       %  figure,imshow(FilterImg2);
       % BW2 = bwmorph(FilterImg,'thin',inf);
       % figure, imshow(BW2);
```

```
end


% --- Executes on button press in Vertices.
function Vertices_Callback(hObject, eventdata, handles)
% hObject    handle to Vertices (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
global  ends intsc input_skeleton_image filename namafile lebar panjang
pathname;

if filename==0
   open('Exception2.fig');
else
      I = imread([pathname,filename]);
      dim = size(namafile);       % size of input image
      leng = length(dim);         % Change input image to gray scale

      if leng == 3
         I = rgb2gray(I);
      end
      imgBW = im2bw(I, 0.90455);      %% Binary Image
      FilterImg = medfilt2(imgBW,[3 1]);
      for i = 1 : lebar
            for j = 1 : panjang
               if FilterImg(i,j) == 1
                  FilterImg(i,j) = 0;
               else
                  FilterImg(i,j) = 1;
               end
            end
      end

      %  figure,imshow(FilterImg);
       FilterImg2=flipud(FilterImg);
       axes(handles.axes2);
       imshow(FilterImg);
      %  figure,imshow(FilterImg2);
      % BW2 = bwmorph(FilterImg,'thin',inf);
      % figure, imshow(BW2);
end
input_skeleton_image = bwmorph(FilterImg2,'thin',inf);
find_vertices(input_skeleton_image);
axes(handles.axes3);
skeleton_image=bwmorph(FilterImg,'thin',inf);
imshow(skeleton_image);
hold on;
```

```
a=size(intsc,1);
b=size(ends,1);
c=ends;
d=intsc;
e=size(skeleton_image,1);
if a~=0 && b~=0
   for i=1:a
     d(i,2) = e - d(i,2);
   end
   for i=1:b
     c(i,2) = e - c(i,2);
   end
   plot(d(:,1),d(:,2),'r*');
   plot(c(:,1),c(:,2),'r*');
   x_ends=ends(:,1);
   Y_ends=ends(:,2);
   koor_ends=num2str([x_ends Y_ends]);
   x_intsc=intsc(:,1);
   Y_intsc=intsc(:,2);
   koor_intsc=num2str([x_intsc Y_intsc]);

end

if a~=0 && b==0
   for i=1:a
     d(i,2) = e - d(i,2);
   end
   plot(d(:,1),d(:,2),'r*');
   koor_ends=num2str([]);
   x_intsc=intsc(:,1);
   Y_intsc=intsc(:,2);
   koor_intsc=num2str([x_intsc Y_intsc]);
end

if a==0 && b~=0
   for i=1:b
     c(i,2) = e - c(i,2);
   end
   plot(c(:,1),c(:,2),'r*');
   x_ends=ends(:,1);
   Y_ends=ends(:,2);
   koor_ends=num2str([x_ends Y_ends]);
   koor_intsc=num2str([]);
end

if a==0 && b==0
   koor_ends=num2str([]);
   koor_intsc=num2str([]);
```

```
end
set(handles.listbox2,'String',{['Koordinat End Points'] [ '-------------------------------
----'] [koor_ends] [] ['Koordinat Intersection Points'] [ '-------------------------------
--'] [koor_intsc]});

% display('Intersection Point');
% intsc
% display('End Point');
% ends


% --- Executes on button press in FE.
function FE_Callback(hObject, eventdata, handles)
% hObject    handle to FE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
warning off all
global ends intsc current_state input_skeleton_image Status  JSTin;

Vertices_Callback(hObject, eventdata, handles);
Value1=[];
Value2=[];
Value=[];
% ------------------------------------------------------------------------
if size(intsc,1)==0 && size(ends,1)~=0
   MSE = Segmen_from_ends(ends,intsc,input_skeleton_image);
   Value = Konfersi_bit(MSE);
   mi=zeros(10,10);
   mi(1,2)=1;mi(2,1)=1;
   MI=mi;
end
% ------------------------------------------------------------------------
if size(intsc,1)>=1 && size(ends,1)~=0
   MSE = Segmen_from_ends(ends,intsc,input_skeleton_image);
   MSI=Segmen_from_intsc(intsc,ends,input_skeleton_image);
   Value1 = Konfersi_bit(MSE);
   Value2 = Konfersi_bit(MSI);
   Value= [Value1;Value2];
   MI = Matrik_interrelationship(MSE,MSI);
end

% ------------------------------------------------------------------------
if size(ends,1)==0,size(intsc,1)==0
   loop=1;
   Value = [1 0 0 0 0 0 0 0 ];
   MI = zeros(10,10);
end
```

```
JSTin = JST_input(Value,MI);
set(handles.listbox2,'String',{['Value'] [ '----------------------------------']
[num2str(Value)] [] ['Matrik Interrelationship'] [ '----------------------------------']
[num2str(MI)] [] ['Matrik Input JST'] [ '----------------------------------']
[num2str(JSTin)]});


% --- Executes on button press in BP.
function BP_Callback(hObject, eventdata, handles)
% hObject    handle to BP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
warning off all
global ends intsc current_state input_skeleton_image Status  JSTin ;

load NN_Radialbasisfunction;
FE_Callback(hObject, eventdata, handles);
input=JSTin';
y=sim(NetBP,input);
set(handles.listbox2,'String',[{'Nilai Output Radialbasisfunction',' ',
num2str(y,'%2.5f')}]);


% --- Executes on button press in Output.
function Output_Callback(hObject, eventdata, handles)
% hObject    handle to Output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
warning off all
tic
global ends intsc current_state input_skeleton_image Status  JSTin;

load NN_Radialbasisfunction;
FE_Callback(hObject, eventdata, handles);
input=JSTin';
y=sim(NetBP,input);
Character = Output(y);
time=toc;
set(handles.time,'String',time);
set(handles.listbox2,'String',{['Output'] ['---------'] [] [ [ '    ' ] Character]});
set(handles.text5,'string',Character);
% set(handles.edit2,'String',Character);

% --- Executes on button press in more.
function more_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to more (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('../Fix CR/Dokumentasi/Sistem.htm');

% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
% hObject    handle to Reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
cla(handles.axes1);
cla(handles.axes2);
cla(handles.axes3);
% set(handles.axes1,'XTick',str2num(''));
% set(handles.axes1,'YTick',str2num(''));
% set(handles.axes2,'XTick',str2num(''));
% set(handles.axes2,'YTick',str2num(''));
% set(handles.axes3,'XTick',str2num(''));
% set(handles.axes3,'YTick',str2num(''));

handles.metricdata.text5 = '?';
handles.metricdata.listbox2 = ' ';
handles.metricdata.time = '00:00';
set(handles.text5, 'String', handles.metricdata.text5);
set(handles.time, 'String', handles.metricdata.time);
set(handles.listbox2, 'String', handles.metricdata.listbox2);


% --- Executes on button press in Exit.
function Exit_Callback(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Close;

% --------------------------------------------------------------------
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
global namafile panjang lebar pathname;
[filename, pathname] = uigetfile({'*.bmp';'*.jpg';'*.gif';'*.*'}, 'Pick an Image
File');
 if filename~=0
    S = imread([pathname,filename]);
    axes(handles.axes1);
```

```
   imshow(S);
   handles.S = S;
  % guidata(hObject, handles);


dimensi=size(S);
panjang = dimensi(1,2);
lebar = dimensi (1,1);
namafile=filename ;
 end

% --------------------------------------------------------------------
function exit_Callback(hObject, eventdata, handles)
% hObject    handle to exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Close;

% --------------------------------------------------------------------
function file_Callback(hObject, eventdata, handles)
% hObject    handle to file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function train_Callback(hObject, eventdata, handles)
% hObject    handle to train (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function IP_Callback(hObject, eventdata, handles)
% hObject    handle to IP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fig1=openfig('InputPattern.fig');
handles=guihandles(fig1);
guidata(fig1,handles)

% --------------------------------------------------------------------
function training_Callback(hObject, eventdata, handles)
% hObject    handle to training (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fig2=open('Train.fig');
handles=guihandles(fig2);
```

```
guidata(fig2,handles);

% --------------------------------------------------------------------
function figure_Callback(hObject, eventdata, handles)
% hObject    handle to figure (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function filter_Callback(hObject, eventdata, handles)
% hObject    handle to filter (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global namafile lebar panjang FilterImg2 filename pathname;
if filename==0
   open('Exception2.fig');
else
   I = imread([pathname,filename]);
   dim = size(namafile);        % size of input image
   leng = length(dim);          % Change input image to gray scale

   if leng == 3
      I = rgb2gray(I);
   end
   imgBW = im2bw(I, 0.90455);      %% Binary Image
   FilterImg = medfilt2(imgBW,[3 1]);
   for i = 1 : lebar
         for j = 1 : panjang
            if FilterImg(i,j) == 1
               FilterImg(i,j) = 0;
            else
               FilterImg(i,j) = 1;
            end
         end
   end

   figure,imshow(FilterImg);
end

% --------------------------------------------------------------------
function skeleton_Callback(hObject, eventdata, handles)
% hObject    handle to skeleton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global namafile lebar panjang FilterImg2 filename pathname;
if filename==0
   open('Exception2.fig');
```

```
else
   I = imread([pathname,filename]);
   dim = size(namafile);        % size of input image
   leng = length(dim);          % Change input image to gray scale

   if leng == 3
      I = rgb2gray(I);
   end
   imgBW = im2bw(I, 0.90455);      %% Binary Image
   FilterImg = medfilt2(imgBW,[3 1]);
   for i = 1 : lebar
         for j = 1 : panjang
            if FilterImg(i,j) == 1
               FilterImg(i,j) = 0;
            else
               FilterImg(i,j) = 1;
            end
         end
   end

   BW2 = bwmorph(FilterImg,'thin',inf);
   figure, imshow(BW2);
end

% --------------------------------------------------------------------
function vertices_Callback(hObject, eventdata, handles)
% hObject    handle to vertices (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function help_Callback(hObject, eventdata, handles)
% hObject    handle to help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to proses (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('../Fix CR/Dokumentasi/Proses.pps')

% --------------------------------------------------------------------
function about_Callback(hObject, eventdata, handles)
% hObject    handle to about (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes2


% --- Executes during object deletion, before destroying properties.
function axes2_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on mouse press over axes background.
function axes2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Base_line :

function [m,C] = Base_line(MS,end_points)

a=MS(1,:);
b=end_points;
m = (b(1,2)-a(1,2))/(b(1,1)-a(1,1));
C = a(1,2)-(m*a(1,1));

Cek_status :

function [Status,end_points]=Cek_status(position)
global intsc ends;
b_intsc = size(intsc,1);
b_ends = size(ends,1);
b_position = size(position,1);
Status=0;
end_points = [];
for i=1:b_position
   for k=1:b_intsc
```

```
        if position(i,:) == intsc(k,:)
            Status=1;
            end_points = intsc(k,:);
           break
        end
    end
    if Status==1,
          break
    end
end
if Status==0
for l=1:b_position
        for j=1:b_ends
            if position(l,:) == ends(j,:)
                Status=1;
                end_points= ends(j,:);
                break
            end
        end
        if Status==1
          break
        end
end
end

if b_position==0
    Status=1;
    end_points=[];
end
```

Deviasi :

```
function [D,Threshold] = Deviasi(str_pts,m,MS)

k = size(MS,1);
a = str_pts(1,1);
b = str_pts(1,2);
teta = Sudut(m);
if teta >= 90
   teta = 180-teta;
else
    teta=teta;
end

% sudut1 = round(teta)
MD=[];
for i=2:(k-2),
```

```
    d0=sqrt((MS(i,1)-a)^2+(MS(i,2)-b)^2);
    d1=sqrt((MS(i,2)-b)^2);
    d2=sqrt((MS(i,1)-a)^2);
    alpha = atand(d1/d2);

    if teta == 90
        jarak = d2;
        MD=[MD;jarak];
    end

    if teta == 0
        jarak = d1;
        MD=[MD;jarak];
    end

    if teta >= alpha
        sudut = teta - alpha;
        jarak = sind(sudut)*d0;
        MD=[MD;jarak];
    else
        sudut = alpha - teta;
        jarak = sind(sudut)*d0;
        MD=[MD;jarak];
    end
end
l=size(MD,1);
D=sum(MD)/l;
% ------------------------------------------------------------------------

Xmax=max(MS(:,1));
Xmin=min(MS(:,1));
Ymax=max(MS(:,2));
Ymin=min(MS(:,2));

Panjang=Xmax-Xmin;
Lebar=Ymax-Ymin;

diagonal=sqrt(Panjang^2+Lebar^2);
Threshold=0.1*diagonal;

Include_distance:

%
%
%  Inputs       :
%             Coord1,Coord2 - Coordinates to be calculated. Each of them in [X,Y]
%
%  Outputs      :
```

```
%             distance - The calculated distance between each pair of coordinates
%
%  Description   :
%       Calculates the euclidean_distance between 2 points
%
%  To Run >> distance = euclidean_distance(Coord1,Coord2)
%
%  Example >> distance = euclidean_distance([2,4;5,6],[4,6;5,7]);

function distance = euclidean_distance(Coord1,Coord2)
if isempty(Coord1) | isempty(Coord2)
   error('No data input');
end
[r1,c1] = size(Coord1);
[r2,c2] = size(Coord2);
if ([r1,c1] ~= [r2,c2]) | (c1~=2) | (c2 ~= 2) | (r1 ~= r2)
   error('Invalid matrix dimensions');
end
distance = sqrt((Coord1(:,1)-Coord2(:,1)).^2 + (Coord1(:,2)-Coord2(:,2)).^2);
```

exception1 :

```
function varargout = Exception1(varargin)
% EXCEPTION1 M-file for Exception1.fig
%      EXCEPTION1, by itself, creates a new EXCEPTION1 or raises the existing
%      singleton*.
%
%      H = EXCEPTION1 returns the handle to a new EXCEPTION1 or the handle
to
%      the existing singleton*.
%
%      EXCEPTION1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in EXCEPTION1.M with the given input
arguments.
%
%      EXCEPTION1('Property','Value',...) creates a new EXCEPTION1 or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Exception1_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Exception1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Exception1
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Exception1_OpeningFcn, ...
                   'gui_OutputFcn',  @Exception1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Exception1 is made visible.
function Exception1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Exception1 (see VARARGIN)

% Choose default command line output for Exception1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Exception1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Exception1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Close;
```

Exception2 :

```
function varargout = Exception2(varargin)
% EXCEPTION2 M-file for Exception2.fig
%      EXCEPTION2, by itself, creates a new EXCEPTION2 or raises the existing
%      singleton*.
%
%      H = EXCEPTION2 returns the handle to a new EXCEPTION2 or the handle
to
%      the existing singleton*.
%
%      EXCEPTION2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in EXCEPTION2.M with the given input
arguments.
%
%      EXCEPTION2('Property','Value',...) creates a new EXCEPTION2 or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Exception2_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Exception2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Exception2

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Exception2_OpeningFcn, ...
                   'gui_OutputFcn',  @Exception2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Exception2 is made visible.
function Exception2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Exception2 (see VARARGIN)

% Choose default command line output for Exception2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Exception2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Exception2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Close;
```

Find_vertices :

```
%  File and function name : find_vertices.m

%  Inputs       :
%            input_skeleton_image - binary image of 1 island representing the
%                                   skeleton.
%            'not testing','testing' (Optional) - Enables or disables testing mode
%                                                 (Default - 'not testing')
%
%  Outputs      :
%            intersecting_pts - [X,Y] coordinates of where the intersection points
%                               of the skeleton are.
%            terminating_pts - [X,Y] coordinates of where the ends of the skeleton
are.
%
%  Description   :
%      Determines the locations of the intersection points in the given skeleton.
%      Determines the locations of end points in the given skeleton.
%
%  To Run >> intersecting_pts = find_skel_intersection(input_skeleton_image)
%
% Example >> I = imread(filename);
%          input_skeleton_image = bwmorph(I,'thin',inf);
%          find_vertices(input_skeleton_image);
%
%
%

function vertices_pts = find_vertices(input_skeleton_image,varargin)
option = 'not testing';
global ends intsc;
if ~isempty(varargin)
   for n = 1:1:length(varargin)
      if strcmp(varargin{n},'testing') | strcmp(varargin{n},'not testing')
         option = varargin{n};
      else
         error('Error in input option');
      end
   end
end

%Extract the relative
[Y,X] = find(input_skeleton_image ~= 0);
skeleton_coord = [X,Y];
end_pts = [];
% Relative vectors to the side borders from the current pixel
border_vector = [-1 -1;...
                 0 -1;...
                 1 -1;...
```

```
                1  0;...
                1  1;...
                0  1;...
               -1  1;...
               -1  0];
for n = 1:1:length(skeleton_coord(:,1))
   if strcmp(option,'testing')
      disp(strcat('find_skel_ends :',num2str(n),'-of-
',num2str(length(skeleton_coord(:,1)))));
   end
   %Select the current coordinate to be tested
   current_coord = skeleton_coord(n,:);
   %Determine the coordinates of the pixels around this pixel
   border_coord = [border_vector(:,1) + current_coord(1),border_vector(:,2) +
current_coord(2)];
   pos = find(ismember(border_coord,skeleton_coord,'rows') == 1);
   if isempty(pos)
      % If this pixel is an edge to a island of 1 pixel
      % Save it.
      end_pts = [end_pts;current_coord];
   else
      %Default assumption: pixel is an edge unless otherwise stated
      present = 0;
      %Test all the pixels around this current pixel
      for m = 1:1:length(pos);
         %For each surrounding pixel, test if there is a corresponding pixel on the
other side
         continunity = [4,5,6] + pos(m);
         g = find(continunity > 8);
         if ~isempty(g)
            continunity(g) = mod(continunity(g),9)+1;
         end
         if any(ismember(continunity',pos,'rows'))
            % If any pixels that are on the opposite side of the current pixel
            % are present then this pixel is not a terminating pixel
            present = 1;
            break;
         end
      end
      % If the current pixel does not fullfill the above condition, then it is an edge
      if present == 0
         end_pts = [end_pts;current_coord];
      end
   end
end


% --------------------------------------------------------------------------------
```

```
% ------------------------------------------------------------------------------------------
input_skeleton_image = double(input_skeleton_image) ./
double(max(input_skeleton_image(:)));
% ------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------
kernel = [1 1 1; 1 1 1; 1 1 1];
conv_img = conv2(input_skeleton_image,kernel,'same');
conv_img = conv_img .* input_skeleton_image;
[Y,X] = find(conv_img > 3);



% ------------------------------------------------------------------------------------------
% If there are intersecting points, select only 1 pixel from each island of
% intersection points
% ------------------------------------------------------------------------------------------
intersecting_pts = [];
if ~isempty(X)
          classes = sortclasses([X,Y],1,8);
          for n = 1:1:length(classes)
       X = mean(classes{n}(:,1));
       Y = mean(classes{n}(:,2));
       temp = classes{n};
       temp(:,1) = X;
       temp(:,2) = Y;
       distance = euclidean_distance(temp,classes{n});
       temp = sortrows([distance,classes{n}]);
       intersecting_pts = [intersecting_pts;temp(1,[2,3])];
          end
else
end

% ------------------------------------------------------------------------------------------
% If testing then display the skeleton image with the detected end points
% ------------------------------------------------------------------------------------------
% if strcmp(option,'not testing')
%    disp('Display the vertices');
%    disp('-----------------------------------------------');

 %   disp('Coordinat Intersecting');
 %   intersecting_pts(:,:),'r*';

 %   disp('-----------------------------------------------');
 %   disp('Coordinat end');
 %   end_pts(:,:),'r*';
%       imshow(input_skeleton_image);
%        hold on;
```

```
%    if ~isempty(intersecting_pts)
%        plot(intersecting_pts(:,1),intersecting_pts(:,2),'r*');
%     end
%        plot(end_pts(:,1),end_pts(:,2),'r*');
%     title('Red points indicated detected vertices points');
%     xlabel('Test mode for "find_vertices.m" function');
% end
%


ends=[end_pts];
intsc=[intersecting_pts];
% ------------------------------------------------------------------------------------------
--
%y=[end_pts
%intersecting_pts];
%matriks = [];
%y_aks = [];
%siz = size(y)
%baris = siz(1,2)
%for h=1 : baris
%    matriks(h,1) = y(h,1) + (0.001*(y(h,2)));
%end
%matriks = sort(matriks);
%ds = sort(y(:,1));
%for h=1 : baris
%    y_aks(h,1) = ds(h,1);
%    y_aks(h,2) = (matriks(h) - ds(h,1)) * 1000;
%end
%w = y_aks*1;

Feature_extraction :
function Bit = Fuzzy_extraction(teta)


% ------------------------------------------------------------------------
% teta=90;
a=(0+teta)/45;
b=(180-teta)/45;
c=(360-teta)/45;
if a<0
   a=a*-1;
end
FH = 1-min(min(min(c,min(a,b)),1));
member=[FH];
% ------------------------------------------------------------------------
d=(90-teta)/45;
e=(270-teta)/45;
```

```
if d<0
   d=d*-1;
end
FV = 1-min(min(min(d,e),1));
member=[member;FV];
% ------------------------------------------------------------------------
f=(45-teta)/45;
g=(225-teta)/45;
if f<0
   f=f*-1;
end
FR = 1-min(min(min(f,g),1));
member=[member;FR];
% ------------------------------------------------------------------------
h=(135-teta)/45;
i=(315-teta)/45;
if h<0
   h=h*-1;
end
FL = 1-min(min(min(h,i),1));
member=[member;FL];
% ------------------------------------------------------------------------

value=max(FH,max(FV,max(FR,FL)));
for i=1:4
   if value==member(i)
      x=i;
      break
   end
end

if x==1
   H=1;V=0;R=0;L=0;
   Bit=[0 1 0 0 0];
end
if x==2
   H=0;V=1;R=0;L=0;
   Bit=[0 0 1 0 0];
end
if x==3
   H=0;V=0;R=1;L=0;
   Bit=[0 0 0 1 0];
end
if x==4
   H=0;V=0;R=0;L=1;
   Bit=[0 0 0 0 1];
end
% ------------------------------------------------------------------------
```

Get_koordinat :

```
function MT=Get_koor_ttg(position,input_skeleton_image)


A=input_skeleton_image;
x=position(1);
y=position(2);
x1=x-1;
x2=x+1;
y1=y-1;
y2=y+1;
MT=[];
for i=x1:x2,
   for j=y1:y2,
      if A(j,i)==1,
         MT=[MT;[i,j]];
      end
   end
end
```

Input_pattern :

```
function varargout = InputPattern(varargin)
% INPUTPATTERN M-file for InputPattern.fig
%      INPUTPATTERN, by itself, creates a new INPUTPATTERN or raises the
existing
%      singleton*.
%
%      H = INPUTPATTERN returns the handle to a new INPUTPATTERN or the
handle to
%      the existing singleton*.
%
%      INPUTPATTERN('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in INPUTPATTERN.M with the given input
arguments.
%
%      INPUTPATTERN('Property','Value',...) creates a new INPUTPATTERN or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before InputPattern_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to InputPattern_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
```

```
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help InputPattern

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @InputPattern_OpeningFcn, ...
                   'gui_OutputFcn',  @InputPattern_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before InputPattern is made visible.
function InputPattern_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to InputPattern (see VARARGIN)

% Choose default command line output for InputPattern
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
initialize_gui(hObject, handles, false);

% UIWAIT makes InputPattern wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = InputPattern_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
global namafile panjang lebar filename pathname;
[filename, pathname] = uigetfile({'*.bmp';'*.jpg';'*.gif';'*.*'}, 'Pick an Image
File');
 if filename~=0
   S = imread([pathname,filename]);
%     axes(handles.axes1);
   imshow(S);
   handles.S = S;
  % guidata(hObject, handles);


dimensi=size(S);
panjang = dimensi(1,2);
lebar = dimensi (1,1);
namafile=filename ;
 end
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
warning off all
global namafile lebar panjang intsc ends filename pathname;
if filename==0
   open('Exception1.fig');
else
       I = imread([pathname,filename]);
       dim = size(namafile);        % size of input image
       leng = length(dim);          % Change input image to gray scale

       if leng == 3
          I = rgb2gray(I);
       end
       imgBW = im2bw(I, 0.90455);      %% Binary Image
```

```
FilterImg = medfilt2(imgBW,[3 1]);
for i = 1 : lebar
      for j = 1 : panjang
         if FilterImg(i,j) == 1
           FilterImg(i,j) = 0;
         else
           FilterImg(i,j) = 1;
         end
      end
end
FilterImg2=flipud(FilterImg);
input_skeleton_image = bwmorph(FilterImg2,'thin',inf);
find_vertices(input_skeleton_image);
Value1=[];
Value2=[];
Value3=[];
% ------------------------------------------------------------------------
if size(intsc,1)==0 && size(ends,1)~=0
   MSE = Segmen_from_ends(ends,intsc,input_skeleton_image);
   Value = Konfersi_bit(MSE);
   mi=zeros(10,10);
   mi(1,2)=1;mi(2,1)=1;
   MI=mi;
end
% ------------------------------------------------------------------------
if size(intsc,1)>=1 && size(ends,1)~=0
   MSE = Segmen_from_ends(ends,intsc,input_skeleton_image);
   MSI=Segmen_from_intsc(intsc,ends,input_skeleton_image);
   Value1 = Konfersi_bit(MSE);
   Value2 = Konfersi_bit(MSI);
   Value= [Value1;Value2];
   MI = Matrik_interrelationship(MSE,MSI);
end

% ------------------------------------------------------------------------
if size(ends,1)==0,size(intsc,1)==0
   loop=1;
   Value = [1 0 0 0 0 0 0 0 ];
   MI = zeros(10,10);
end

JSTin = JST_input(Value,MI);
huruf = char(get(handles.huruf,'String'));
Nilai = Nilai_huruf(huruf);

load JSTInput;

%      JST_Target=[];
```

```
%        JST_Input=[];
       JST_Target = JSTTarget(JST_Target,Nilai);
       JST_Input  = JSTInput(JST_Input,JSTin);
       save JSTInput JST_Input  JST_Target
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close;



function huruf_Callback(hObject, eventdata, handles)
% hObject    handle to huruf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of huruf as text
%        str2double(get(hObject,'String')) returns contents of huruf as a double


% --- Executes during object creation, after setting all properties.
function huruf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to huruf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
initialize_gui(gcbf, handles, true);
```

```
function initialize_gui(fig_handle, handles, isreset)
% If the metricdata field is present and the reset flag is false, it means
% we are we are just re-initializing a GUI by calling it from the cmd line
% while it is up. So, bail out as we dont want to reset the data.
global filename;
if isfield(handles, 'metricdata') && ~isreset
   return;
end

handles.metricdata.edit3 = '';
set(handles.huruf, 'String', handles.metricdata.edit3);
cla;
filename=0;

JST_input :

function JSTin = JST_input(Value,MI)

k=size(Value,1);
h=size(MI,1);
JSTin=[];

a=zeros(10,8);
for i=1:k
   a(i,:)=Value(i,:);
end

b=size(a,1);
for i=1:b
   JSTin=[JSTin a(i,:)];
end

for j=1:h
   JSTin=[JSTin MI(j,:)];
end

Konversi_bit :

function Value = Konfersi_bit(MS)

a=size(MS,2);
Value=[];
   for i=1:a
      MS{i};
      str_pts=MS{i}(1,:);
      end_points=MS{i}(size(MS{i},1),:);
```

```
        [m,C]=Base_line(MS{i},end_points);
        teta = Sudut(m);
        loop = Loop_status(str_pts,end_points,MS{i});
        if loop == 1
           Bit=[1 0 0 0 0 0 0 0];
        else

           Bit = Fuzzy_extraction(teta);
           [D,Threshold]=Deviasi(str_pts,m,MS{i});
           if D > Threshold
              [IR,IL]= Status_left_right(teta,C,MS{i});
                    if IR == 1 && IL == 0
                       CR = [0 1 0];
                       Bit = [Bit CR] ;
                    end

                    if IL == 1 && IR == 0
                       CL = [0 0 1];
                       Bit = [Bit CL];
                    end

                    if IL == 0 && IR == 0
                       CL = [0 0 1];
                       Bit = [Bit CL];
                    end

                    if IL == 1 && IR == 1
                       CRL = [0 1 1];
                       Bit = [Bit CRL];
                    end
           else
              line = [1 0 0];
              Bit = [Bit line];

           end

        end
        Value=[Value;Bit];
     end
```

Loop_status :

```
function loop = Loop_status(str_pts,end_points,MS)


k = size(MS,1);
a = str_pts(1,1);
b = str_pts(1,2);
```

```
e = end_points(1,1);
f = end_points(1,2);
alpha = 45;
d = sqrt((e-a)^2+(f-b)^2 );
g = (k + d)* (alpha/360);


if d <= g
   loop = true;
 else
   loop = false;
end
```

Matrik_interrelationship :

```
function MI = Matrik_interrelationship(MSE,MSI)

a=size(MSE,2);
b=size(MSI,2);
c=zeros(10,10);
% for i=1:(a+b)
%     for j=1:(a+b)
%         if c(i,j)==1
%             c(i,j)=0;
%         end
%     end
% end

% ------------------------------------------------------------------------
for i=1:a
   Segmen{i}=MSE{i};
end

for j=1:b
   Segmen{a+j}=MSI{j};
end
% ------------------------------------------------------------------------
k=size(Segmen,2);
for i=1:k
   sgm=Segmen{i};
   m=size(sgm,1);
   str_pts{i}=sgm(1,:);
   end_points{i}=sgm(m,:);
end
% ------------------------------------------------------------------------
for i=1:(k-1)
   for j=i+1:k
```

```
      if sum(str_pts{i})==sum(end_points{j}) ||
sum(end_points{i})==sum(str_pts{j}) ||
sum(end_points{i})==sum(end_points{j})|| sum(str_pts{i})==sum(str_pts{j})
          c(i,j)=1;
          c(j,i)=1;
       end
   end
end

MI=c;
```

Nilai_huruf :

```
function Nilai = Nilai_huruf(huruf)

switch huruf
      case 'A'
         Nilai = 0.018;
      case 'a'
         Nilai = 0.036;
      case 'B'
         Nilai = 0.054;
      case 'b'
         Nilai = 0.072;
      case 'C'
         Nilai = 0.090;
      case 'c'
         Nilai = 0.108;
      case 'D'
         Nilai = 0.126;
      case 'd'
         Nilai = 0.144;
      case 'E'
         Nilai = 0.162;
      case 'e'
         Nilai = 0.180;
      case 'F'
         Nilai = 0.198;
      case 'f'
         Nilai = 0.216;
      case 'G'
         Nilai = 0.234;
      case 'g'
         Nilai = 0.252;
      case 'H'
         Nilai = 0.270;
      case 'h'
         Nilai = 0.288;
```

```
case 'I'
   Nilai = 0.306;
case 'i'
   Nilai = 0.324;
case 'J'
   Nilai = 0.342;
case 'j'
   Nilai = 0.360;
case 'K'
   Nilai = 0.378;
case 'k'
   Nilai = 0.396;
case 'L'
   Nilai = 0.414;
case 'l'
   Nilai = 0.432;
case 'M'
   Nilai = 0.450;
case 'm'
   Nilai = 0.468;
case 'N'
   Nilai = 0.486;
case 'n'
   Nilai = 0.504;
case 'O'
   Nilai = 0.522;
case 'o'
   Nilai = 0.540;
case 'P'
   Nilai = 0.558;
case 'p'
   Nilai = 0.576;
case 'Q'
   Nilai = 0.594;
case 'q'
   Nilai = 0.612;
case 'R'
   Nilai = 0.630;
case 'r'
   Nilai = 0.648;
case 'S'
   Nilai = 0.666;
case 's'
   Nilai = 0.684;
case 'T'
   Nilai = 0.702;
case 't'
   Nilai = 0.720;
```

```
    case 'U'
       Nilai = 0.738;
    case 'u'
       Nilai = 0.756;
    case 'V'
       Nilai = 0.774;
    case 'v'
       Nilai = 0.792;
    case 'W'
       Nilai = 0.810;
    case 'w'
       Nilai = 0.828;
    case 'X'
       Nilai = 0.846;
    case 'x'
       Nilai = 0.864;
    case 'Y'
       Nilai = 0.882;
    case 'y'
       Nilai = 0.900;
    case 'Z'
       Nilai = 0.918;
    case 'z'
       Nilai = 0.936;
  otherwise
      open('Exception1.fig');
  end
```

Radial_basis_function :

```
function NN_radialbasisfunction(JSTInput,)

RBF=newrb(minmax(JSTInput),
[180,180,1],{'radbas','gaussmf','purelin'},'spread','goal','trainr');
net.train
```

Output :

```
function Character = Output(y)


% a=0;
% c=[];
% max=1;
% for i=1:52
%     a=a+0.018;
%     b=[a y];
%     jarak=diff(b);
```

```
%     if jarak < 0
%         jarak=jarak*-1;
%     end
%     c=[c;jarak];
% end
% urut=0;
% d=min(c);
% for i=1:52
%     if d==c(i)
%         urut=i;
%         break
%     end
%
% end
a=max(y);
urut=0;

for i=1:52
   if y(i)== a
      urut=i;
      break
   end

end
switch urut
      case 1
         Character = 'A';
      case 2
         Character = 'a';
      case 3
         Character = 'B';
      case 4
         Character = 'b';
      case 5
         Character = 'C';
      case 6
         Character = 'c';
      case 7
         Character = 'D';
      case 8
         Character = 'd';
      case 9
         Character = 'E';
      case 10
         Character = 'e';
      case 11
         Character = 'F';
      case 12
```

```
   Character = 'f';
case 13
   Character = 'G';
case 14
   Character = 'g';
case 15
   Character = 'H';
case 16
   Character = 'h';
case 17
   Character = 'I';
case 18
   Character = 'i';
case 19
   Character = 'J';
case 20
   Character = 'j';
case 21
   Character = 'K';
case 22
   Character = 'k';
case 23
   Character = 'L';
case 24
   Character = 'l';
case 25
   Character = 'M';
case 26
   Character = 'm';
case 27
   Character = 'N';
case 28
   Character = 'n';
case 29
   Character = 'O';
case 30
   Character = 'o';
case 31
   Character = 'P';
case 32
   Character = 'p';
case 33
   Character = 'Q';
case 34
   Character = 'q';
case 35
   Character = 'R';
case 36
```

```
        Character = 'r';
    case 37
        Character = 'S';
    case 38
        Character = 's';
    case 39
        Character = 'T';
    case 40
        Character = 't';
    case 41
        Character = 'U';
    case 42
        Character = 'u';
    case 43
        Character = 'V';
    case 44
        Character = 'v';
    case 45
        Character = 'W';
    case 46
        Character = 'w';
    case 47
        Character = 'X';
    case 48
        Character = 'x';
    case 49
        Character = 'Y';
    case 50
        Character = 'y';
    case 51
        Character = 'Z';
    case 52
        Character = 'z';
  otherwise
      Character = '?';
 end
```

Training :

```
function varargout = Train(varargin)
% TRAIN M-file for Train.fig
%      TRAIN, by itself, creates a new TRAIN or raises the existing
%      singleton*.
%
%      H = TRAIN returns the handle to a new TRAIN or the handle to
%      the existing singleton*.
%
%      TRAIN('CALLBACK',hObject,eventData,handles,...) calls the local
```

```
%      function named CALLBACK in TRAIN.M with the given input arguments.
%
%      TRAIN('Property','Value',...) creates a new TRAIN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Train_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Train_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Train

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Train_OpeningFcn, ...
                   'gui_OutputFcn',  @Train_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Train is made visible.
function Train_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Train (see VARARGIN)

% Choose default command line output for Train
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes Train wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Train_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
load JSTInput;
% ------------------------------------------------------------------------
a=size(JST_Target,1);
g=zeros(a,52);
for i=1:a
   c=JST_Target(i,:);
   d=round(c/0.018);
   g(i,d)=1;
end
JST_Target=g';
% ------------------------------------------------------------------------
input=JST_Input';
target=JST_Target;
mse = str2double(get(handles.edit3,'String'));
epochs = str2double(get(handles.edit4,'String'));
   NetBP =newff(minmax(input),[300,240,52],{'logsig','logsig','logsig'},'trainrp');
   NetBP.trainParam.epochs=epochs;
   NetBP.trainParam.goal=mse;
%     NetBP.performFcn = 'sse';
%     NetBP.trainParam.mc = 0.95;
%     NetBP.trainParam.lr=0.2;
   NetBP=train(NetBP,input,target)
save NN_Backpropagation NetBP;

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close;



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```