PENGENALAN HURUF CETAK DENGAN METODE *HIDDEN MARKOV MODEL*

SKRIPSI

Sebagai salah satu syarat untuk meraih gelar Sarjana Komputer dalam bidang Ilmu Komputer

Oleh:
RINI ALIFAH
0510960052-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2012


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

PENGENALAN HURUF CETAK DENGAN METODE *HIDDEN MARKOV MODEL*

Oleh:
RINI ALIFAH
0510960052 – 96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 7 Agustus 2012
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Dosen Pembimbing I,	**Dosen Pembimbing II,**
Edy Santoso, SSi., M.Kom	**Dian Eka Ratnawati, SSi., M.Kom**
NIP. 197404142003121004	**NIP. 197306192002122001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP. 196709071992031001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	: Rini Alifah
NIM	: 0510960052-96
Jurusan	: Matematika
Program Studi	: Ilmu Komputer
Judul Skripsi	: Pengenalan Huruf Cetak dengan Metode *Hidden Markov Model*

Dengan ini menyatakan bahwa :

1. Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.
2. Apabila dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 7 Agustus 2012
Yang menyatakan,

Rini Alifah
NIM. 0510960052-96

PENGENALAN HURUF CETAK DENGAN METODE *HIDDEN MARKOV MODEL*

ABSTRAK

Optical Character Recognition (OCR) merupakan sistem komputer yang dapat mengenali karakter teks dari sebuah file gambar menjadi file *editable teks* yang dapat dialihkan ke aplikasi program pengolah kata di kompuer. Pada penelitian ini, jenis OCR yang dikembangkan adalah *printed character recognition* atau pengenalan huruf cetak. Pengenalan dilakukan pada file gambar bitmap yang berisi karakter yang berasal dari hasil pemindaian *hardcopy*.

Sistem pengenalan ini dibuat dengan menggunakan metode *Hidden Markov Model* (HMM) sebagai algoritma pembelajaran dan pengenalan. Sebelum dilakukan proses pembelajaran dan pengenalan, akan dilakukan beberapa proses pengolahan gambar terlebih dahulu, antara lain proses binerisasi, *thinning*, ekstraksi ciri, dan segmentasi. Proses pembelajaran dan pelatihan HMM dilakukan dengan menggunakan algoritma *forward backward* dan algoritma *baum welch*. Algoritma *baum welch* digunakan untuk mengestimasi model parameter HMM yang telah dibuat sehingga didapatkan model HMM yang baru. Untuk menghitung nilai *likelihood* dari tiap karakter, digunakan algoritma *forward backward*.

Hasil pengujian menunjukkan bahawa rata-rata akurasi hasil pengenalan karakter untuk jenis font yang telah dilatihkan adalah sebesar 94.9%, sedangkan hasil pengenalan karakter untuk jenis font non-pelatihan adalah sebesar 60%.

PRINTED CHARACTER RECOGNITION USING HIDDEN MARKOV MODEL METHOD

ABSTRACT

Optical character reconition (OCR) is a computer system that recognize text character from an image file and convert it into editable text file. The proposed research, used the printed character recognition. The recognition will be done in bitmap image file containing character from scanlated image.

The system recognition is build based on Hidden Markov Model (HMM) as learning and recognition algorithm. Before the learning and recognition, the system will do the image prepocessing first. The image preprocessing including converting image into binary, thinning, feature extraction and segmentation. The learning and recognition process from hidden markov model will be done with forward backward algorithm and baum welch algorithm. Baum welch algorithm will be used to estimate hidden markov model parameter from the old HMM parameter that produced by forward backward algorithm. To count the likehood value form each character, forward backward algorithm will be used.

The test result shows that the average accurate character recognition value from the learned font is 94.9%, and the average value from unlearned font is 60%.

KATA PENGANTAR

Syukur Alhamdulillah penulis ucapkan kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya sehingga penyusunan skripsi dengan judul "Pengenalan Huruf Cetak dengan metode *Hidden Markov Model*" ini dapat terselesaikan. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi ilmu komputer, jurusan matematika, fakultas MIPA, Universitas Brawijaya.

Dalam penyusunan skripsi ini, penulis telah mendapat dukungan dari berbagai pihak. Oleh karena itu, atas bantuan yang telah diberikan, penulis menyampaikan ucapan terima kasih kepada:

1. Edy Santoso, S.Si., M.Kom., selaku dosen pembimbing utama. Terima kasih atas semua saran, kritik, waktu, bantuan, serta bimbingan yang telah diberikan.
2. Dian Eka Ratnawati, S.Si., M.Kom., selaku dosen pembimbing pendamping. Terima kasih atas semua saran, kritik, waktu, bantuan, serta bimbingan yang telah diberikan.
3. Djoko Pramono, ST. selaku Penasihat Akademik.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis.
5. Kedua orang tua dan saudara penulis yang selalu memberikan dukungan serta doa kepada penulis.
6. Teman-teman dari Program Studi Ilmu Komputer Universitas Brawijaya, terima kasih atas semua bantuan yang telah diberikan.
7. Teman-teman kos yang selalu memberikan semangat dan dukungannya kepada penulis.
8. Pihak-pihak yang tidak dapat kami sebutkan satu persatu, atas bantuan dalam penyusunan skripsi ini..

Penulis menyadari sepenuhnya bahwa dalam penyusunan skripsi ini masih banyak kekurangan, karena keterbatasan pengetahuan dan kemampuan, untuk itu kritik dan saran yang membangun dari pembaca sangat diharapkan.

Demikian kata pengantar ini, semoga dapat bermanfaat, khususnya bagi penulis dan bagi pembaca pada umumnya.

Malang, 2012

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR SOURCECODE

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Digitalisasi informasi adalah suatu proses untuk mengubah informasi dari dokumen cetak (*hard copy*) menjadi dokumen digital. Pembuatan buku digital memiliki beberapa keuntungan, diataranya adalah agar mudah diperoleh melalui *search engine*, memiliki ruang penyimpanan yang lebih kecil, mudah dibawa kemana-mana, serta dapat menghindarkan dokumen dari kerusakan yang biasanya dialami oleh buku cetak seperti halaman yang hilang, sobek, terkena coretan, dan sebagainya.

Salah satu cara untuk melakukan proses digitalisasi informasi adalah dengan cara pemindaian dokumen buku dengan bantuan alat optik, seperti scanner dan kamera digital. Namun, hasil dari pemindaian tersebut biasanya masih dalam format gambar sehingga tidak dapat dilakukan proses pengeditan jika terdapat kesalahan dalam penulisan. Pengguna juga harus mengetik ulang sendiri jika ingin menyalin teks yang ada dan menyimpannya ke dalam komputer. Hal ini tentu akan memerlukan waktu yang lama. Oleh karena itu, diperlukan suatu program yang dapat mengenali karakter teks dari sebuah file gambar yang selanjutnya dapat disimpan menjadi file teks yang dapat diedit. Salah satu perangkat lunak yang dapat mengubah format gambar menjadi format *editable* teks yang dapat dialihkan ke aplikasi program pengolah kata di komputer adalah *Optical Character Recognition* (OCR).

Optical Character Recognition (OCR) adalah sebuah sistem komputer yang mampu mengalihkan karakter gambar ke dalam aplikasi program pengolah kata tanpa melakukan pengetikan ulang dari sumber dokumen yang dimaksud (Kuswara, 2006). Proses OCR merupakan proses yang menerjemahkan gambar karakter (*character image*) menjadi bentuk teks dengan cara mencocokkan pola karakter per baris dengan pola yang telah tersimpan dalam database aplikasi (Nur, 2007). Hasil dari proses OCR adalah berupa teks sesuai yang tampak pada gambar output scanner dimana tingkat keakuratan penerjemahan karakter tergantung dari tingkat kejelasan gambar. OCR membutuhkan sebuah perangkat keras berupa scanner (alat optik untuk duplikasi gambar dari bentuk cetak ke dalam format digital) dan sebuah *software* (perangkat lunak sebagai program).

Dalam penelitian ini akan dibangun suatu sistem pengenalan karakter dengan menggunakan metode *Hidden Markov Model* (HMM). *Hidden Markov Models* (HMM) merupakan sebuah model statistik dari suatu proses *Markov* dengan parameter-parameter tersembunyi (*hidden*) yang akan diperoleh dari parameter-parameter yang diamati (*observable*). Pada HMM *state*-nya tidak dapat diamati secara langsung (*hidden*), dimana setiap *state* memiliki distribusi peluang output yang mungkin muncul sebagai suatu set proses stokastik yang akan membentuk suatu deretan observasi. Pada pengenalan karakter, algoritma HMM yang digunakan adalah *Baum welch* dan *Forward backward*. Algoritma *Baum welch* digunakan untuk mengestimasi parameter HMM awal sehingga didapatkan parameter HMM yang baru, sedangkan algoritma *forward backward* digunakan untuk menghitung nilai *likelihood* dari karakter. Dari hasil penelitiannya, Jianying, 2007 menunjukkan bahwa *Hidden Markov Model* memiliki tingkat keakuratan yang cukup tinggi dalam pengenalan karakter, yaitu dengan hasil pengenalan rata-rata sebesar 90.8% untuk data yang sudah dilatihkan dan 74.6% untuk data yang belum dilatihkan.

Berdasarkan latar belakang yang telah dipaparkan, maka judul yang diambil dalam penelitian ini adalah *"PENGENALAN HURUF CETAK DENGAN METODE HIDDEN MARKOV MODEL"*.

1.2 Rumusan Masalah

1. Bagaimana merancang dan membangun sistem Pengenalan huruf cetak dengan menggunakan metode *Hidden Markov Model*.
2. Bagaimana tingkat keakuratan metode *Hidden Markov Model* dalam mengenali karakter.

1.3 Batasan Masalah

1. Pengujian dilakukan pada file hasil scanner.
2. Pengenalan dilakukan terhadap data offline.
3. File citra hanya berisi dokumen teks saja.
4. Teks merupakan huruf tegak dan tidak bersambung.
5. Citra yang diproses adalah citra dalam format bitmap (.bmp).
6. Karakter yang akan dikenali terdiri dari huruf alfabet, yang terdiri huruf besar, huruf kecil, dan angka.
7. Besar ukuran citra yang digunakan maksimal 400x400 piksel.

8. Pengenalan dilakukan pada huruf cetak yang terdiri dari 6 jenis huruf yang berbeda, yaitu arial, berlin sans FB, comic sans MS, microsoft sans serif, times new roman, dan verdana.

1.4 Tujuan

1. Mempelajari dan menerapkan sistem pengenalan huruf cetak dengan menggunakan metode *Hidden Markov Model*.
2. Menganalisa tingkat keakuratan metode *Hidden Markov Model* dalam mengenali karakter.

1.5 Manfaat

Dari hasil penelitian yang telah dilakukan, diharapkan perangkat lunak pengenalan karakter dengan metode *hidden markov model* yang telah dibangun dapat digunakan dan bermanfaat bagi pengembangan ilmu pengetahuan.

1.6 Sistematika Penulisan

Adapun sistematika penulisan yang digunakan adalah sebagai berikut :

1. BAB I PENDAHULUAN
   Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, serta sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Bab ini berisi tentang dasar-dasar teori yang digunakan dalam perancangan dan pembuatan sistem Pengenalan Huruf Cetak dengan metode *Hidden Markov Model*.
3. BAB III METODOLOGI DAN PERANCANGAN
   Bab ini menjelaskan tentang perancangan sistem Pengenalan Huruf Cetak, langkah-langkah yang dilakukan dalam pembuatan sistem, serta cara kerja dari metode *Hidden Markov Model*.
4. IMPLEMENTASI DAN PEMBAHASAN
   Bab ini menjelaskan tentang sistem pengenalan huruf cetak yang telah dibuat, uji coba, serta analisanya.
5. PENUTUP
   Bab ini berisi kesimpulan yang didapat dari pembahasan serta saran yang diperlukan untuk perbaikan penulisan selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Pengolahan Citra Digital

2.1.1 Pengertian Citra

Citra (*image*) adalah gambar pada bidang dwimatra (dua dimensi). Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi obyek, obyek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, misalnya mata manusia, kamera, pemindai (*scanner*), dan sebagainya, sehingga bayangan obyek yang disebut citra tersebut terekam.

Citra ada dua macam, yaitu citra kontinu dan citra diskrit. Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog, misalnya mata manusia dan kamera analog. Citra diskrit dihasilkan melalui proses digitalisasi terhadap citra kontinu. Beberapa sistem optik dilengkapi dengan fungsi digitalisasi sehingga ia mampu menghasilkan citra diskrit, misalnya kamera digital dan *scanner*. Citra diskrit disebut juga citra digital. (Munir, 2004)

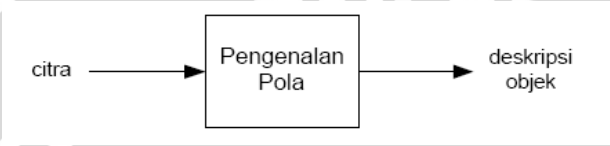
2.1.2 Pengolahan Citra

Pengolahan citra merupakan kegiatan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih baik daripada citra masukan, misalnya citra yang warnanya kurang tajam, kabur (*blurring*), mengandung *noise* (misal bintik-bintik putih), dan lain-lain sehingga perlu ada pemrosesan untuk memperbaiki citra tersebut. Proses pengolahan citra dapat dilihat pada gambar 2.1.

Gambar 2.1 Pengolahan citra

Pengenalan Pola mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini komputer). Tujuan pengelompokan adalah untuk mengenali suatu obyek di dalam

citra. Manusia bisa mengenali obyek yang dilihatnya karena otak manusia telah belajar mengklasifikasi obyek-obyek di alam sehingga mampu membedakan suatu obyek dengan obyek lainnya. Kemampuan sistem visual manusia inilah yang dicoba ditiru oleh mesin. Komputer menerima masukan berupa citra obyek yang akan diidentifikasi, memproses citra tersebut, dan memberikan keluaran berupa deskripsi obyek di dalam citra. Proses pengenalan pola dapat dilihat pada gambar 2.2.



Gambar 2.2 Proses pengenalan pola
(Munir, 2004)

2.2 Operasi Pengolahan Citra Digital

Operasi pengolahan citra yang dilakukan untuk mentransformasikan suatu citra menjadi citra lain dapat dikategorikan berdasarkan tujuan transformasi maupun cakupan operasi yang dilakukan terhadap citra.

Berdasarkan tujuan transformasi, operasi pengolahan citra dikategorikan sebagai berikut :

- Peningkatan kualitas citra (*Image Enhancement*)
  Operasi peningkatan kualitas citra bertujuan untuk meningkatkan fitur tertentu pada citra.
- Pemulihan Citra (*Image Restoration*)
  Operasi pemulihan citra bertujuan untuk mengembalikan kondisi citra pada kondisi yang diketahui sebelumnya akibat adanya pengganggu yang menyebabkan penurunan kualitas citra.

Berdasarkan cakupan operasi yang dilakukan terhadap citra, operasi pengolahan citra dikategorikan sebgai berikut :

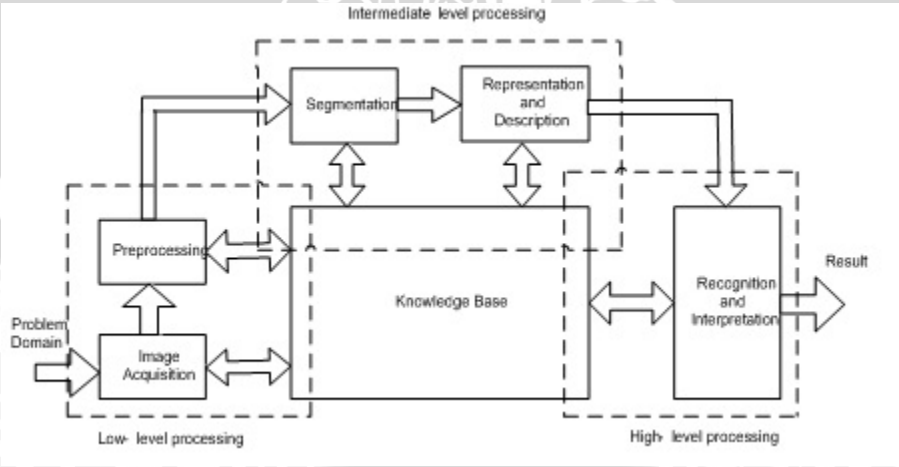
- Operasi titik
  Merupakan operasi yang dilakukan terhadap setiap piksel pada citra yang keluarannya hanya ditentukan oleh nilai piksel itu sendiri.
- Operasi area
  Merupakan operasi yang dilakukan terhadap setiap piksel pada citra yang keluarannya dipengaruhi oleh piksel tersebut dan

piksel lainnya dalam suatu daerah tertentu. Salah satu contoh dari operasi berbasis area adalah operasi ketetanggaan dengan piksel yang sedang diolah.

- Operasi global
  Merupakan operasi yang dilakukan terhadap setiap piksel pada citra yang keluarannya ditentukan oleh keseluruhan piksel yang membentuk citra.

Terdapat beberapa langkah dalam proses pengolahan citra secara umum yang terdiri dari :

- Pembentukan citra (*Data Acquisition*) : Menentukan data yang diperlukan.
- Pengolahan citra tingkat awal (*Image Preprocessing*) : meningkatkan kontras, menghilangkan gangguan geometrik/radiometrik, menentukan bagian citra yang akan diobservasi.
- Segmentasi Citra (*Image Segmentation*) dan deteksi sisi (*Edge Detection*) : Melakukan partisi citra menjadi wilayah-wilayah obyek (*internal properties*) atau menentukan garis batas wilayah obyek (*external shape characteristics*).
- Seleksi dan Ekstraksi Ciri (*Feature Extraction and Selection*) : Seleksi ciri memilih informasi kuantitatif dari ciri yang ada, yang dapat membedakan kelas-kelas obyek secara baik. Ekstraksi Ciri mengukur besaran kuantitatif ciri setiap piksel.



Gambar 2.3 Proses Pengolahan Citra dan Implementasinya

2.2.1 Binerisasi

Citra biner (*binary image*) adalah citra yang hanya mempunyai dua nilai derajat keabuan : hitam dan putih. Piksel-piksel obyek bernilai 1 dan piksel-piksel latar belakang bernilai 0. Citra biner memiliki sejumlah keuntungan sebagai berikut :

1. Kebutuhan memori kecil karena nilai derajat keabuan hanya membutuhkan representasi 1 bit.
2. Waktu pemrosesan lebih cepat dibandingkan dengan citra hitam putih karena banyak operasi pada citra biner yang dilakukan sebagai operasi logika daripada operasi aritmetika bilangan bulat.

Aplikasi yang menggunakan citra biner sebagai masukan untuk pemrosesan pengenalan obyek diantaranya adalah pengenalan karakter secara optik, analisis kromosom, pengenalan sparepart komponen industri, dan sebagainya. (Munir, 2004)

Konversi citra *grayscale* ke citra biner dilakukan dengan *thresholding* (operasi pengambangan). Operasi pengambangan mengelompokkan nilai derajat keabuan setiap piksel ke dalam dua kelas yaitu hitam dan putih. Dua pendekatan yang digunakan dalam versi pengambangan adalah pengambangan secara global dan pengambangan secara lokal adaptif. Pada pengambangan secara global setiap piksel di dalam citra dipetakan ke dua nilai yaitu 1 atau 0 dengan fungsi pengambangan seperti pada persamaan 2.1 :

$$g(x, y) = \begin{cases} 1 \text{ jika } f(x, y) > T \\ 0 \text{ jika } f(x, y) \leq T \end{cases} \qquad (2.1)$$

Dimana:

f(x,y) adalah citra *grayscale*
g(x,y) adalah citra biner
T adalah nilai ambang yang dispesifikasikan

Nilai ambang *T* dipilih sedemikian sehingga galat yang diperoleh sekecil mungkin. Cara yang umum menentukan nilai *T* adalah dengan membuat histogram citra. Nilai *T* dapat dipilih secara manual atau dengan teknik yang otomatis. Teknik yang manual dilakukan dengan cara coba-coba (*trial and error*) dan menggunakan histogram sebagai panduan.

Dengan operasi pengambangan tersebut, obyek dibuat berwarna gelap (1 atau hitam) sedangkan latar belakang berwarna terang (0 atau putih).

Pada pengambangan secara global tidak selalu tepat untuk seluruh macam gambar. Beberapa informasi penting di dalam gambar mungkin hilang karena pengambangan global ini. Lagipula tidak ada harga nilai ambang yang berlaku secara global untuk seluruh daerah citra (misalnya pada citra kedokteran).

Sedangkan pada pengambangan secara lokal dilakukan terhadap daerah di dalam citra. Dalam hal ini, citra dipecah menjadi bagian kecil kemudian proses pengambangan dilakukan secara lokal. Nilai ambang untuk setiap bagian belum tentu sama dengan bagian yang lain. Dengan pengambangan secara lokal adaptif, secara subjektif citra biner yang dihasilkan terlihat lebih baik dan sedikit informasi yang hilang. (Munir, 2004)

Untuk menemukan nilai threshold yang sesuai, maka digunakan *Iterative Selection*. *Iterative selection* adalah proses dimana dibuat sebuah nilai threshold awal sebagai inisialisasi dan terus diperbaiki seiring dengan proses penelusuran gambar. Nilai threshold awal adalah nilai rata-rata *grey-level* (dianggap sebagai Tb), rata-rata dari nilai piksel yang lebih besar atau sama dengan Tb dianggap sebagai To maka nilai perkiraan untuk nilai threshold adalah (Tb + To) / 2. Hal ini terus diulang sampai tidak ada perubahan nilai.

2.2.2 Segmentasi

Segmentasi adalah proses pembagian sebuah citra kedalam sejumlah bagian atau obyek. Segmentasi merupakan suatu bagian yang sangat penting dalam analisis citra secara otomatis, sebab pada prosedur ini obyek yang diinginkan akan disadap untuk proses selanjutnya, misalnya: pada pengenalan pola. Algoritma segmentasi didasarkan pada 2 buah karakteristik nilai derajad kecerahan citra, yaitu: *discontinuity* dan *similarity*. (Gonzalez , 1987)

Segmentasi citra merupakan bagian dari proses pengolahan citra. Proses segmentasi citra ini lebih banyak merupakan suatu proses pra pengolahan pada sistem pengenalan obyek dalam citra. Segmentasi citra (*image segmentation*) mempunyai arti membagi suatu citra menjadi wilayah-wilayah yang homogen berdasarkan kriteria keserupaan yang tertentu antara tingkat keabuan suatu piksel dengan tingkat keabuan piksel – piksel tetangganya, kemudian hasil dari proses segmentasi ini

akan digunakan untuk proses tingkat tinggi lebih lanjut yang dapat dilakukan terhadap suatu citra, misalnya proses klasifikasi citra dan proses identifikasi obyek. Adapun dalam proses segmentasi citra itu sendiri terdapat beberapa algoritma, diantaranya : algoritma Deteksi Titik, Deteksi Garis, dan Deteksi Sisi (berdasarkan Operator Robert dan Operator Sobel).

2.2.3 Thinning

Penipisan citra (*thinning*) merupakan operasi pemrosesan reduksi citra biner yang dalam hal ini obyek (*region*) menjadi rangka (*skeleton*) yang menghampiri garis sumbu obyek. Tujuannya adalah mengurangi bagian yang tidak perlu (*redundant*) sehingga dihasilkan informasi yang esensial saja. Pola penipisan harus tetap mempunyai bentuk yang menyerupai pola asalnya. *Thinning* merupakan metode yang digunakan untuk *skeleton*izing yang salah satu penggunaannya adalah dalam aplikasi pengenalan pola.

Thinning adalah suatu operasi di mana suatu obyek diubah menjadi garis-garis yang kira-kira adalah garis tengah, yang disebut sebagai *skeleton*. Tujuannya adalah mereduksi komponen citra menjadi suatu informasi yang sifatnya mendasar, sehingga dapat memfasilitasi analisis selanjutnya. Walaupun operasi *thinning* dapat dikenakan pada berbagai bentuk obyek, tetapi kegunaan sebenarnya dari proses ini baru terlihat pada bentuk-bentuk memanjang. Operasi ini sangat berguna apabila kita lebih tertarik pada pola relatif obyek ketimbang ukuran obyek. Proses ini kerap diterapkan pada pra-analisis dokumen untuk mengenali garis-garis pada gambar dan stroke karakter pada teks.

Cara umum yang digunakan dalam operasi *thinning* adalah dengan memeriksa setiap piksel di dalam citra dalam kaitannya dengan *neighborhood*. *region* minimal untuk satu kelompok *neighborhood* berukuran 3×3 piksel, dan “ mengupas “ batas daerah citra satu per satu piksel, sampai menjadi baris. Proses ini dilakukan berulang-ulang, pada tiap iterasi, setiap piksel diperiksa dalam kelompok piksel berukuran nxn, dan batas setebal satu piksel yang tidak digunakan untuk mempertahankan koneksi atau tidak pada posisi di ujung garis dihapus. Iterasi akan berakhir bila tidak ada perubahan yang terjadi (tidak ada lagi aksi penghapusan) pada obyek yang sedang dikenakan operasi. Terdapat banyak algoritma untuk *image thinning* dengan tingkat kompleksitas, efisiensi, dan akurasi yang berbeda-beda.

Algoritma *thinning* dapat dibagi menjadi 2 jenis, yaitu iteratif dan *non*-iteratif. Walaupun proses algoritma *thinning* yang *non*-iteratif lebih cepat dibandingkan dengan yang iteratif, tapi tidak selalu menghasilkan hasil yang bagus. Algoritma bersifat iteratif berguna untuk mengikis lapisan piksel terluar sampai tidak ada lapisan lagi yang dapat dihilangkan. Kebanyakan algoritma iteratif *thinning* menggunakan *Template based mark-and-delete*. Metode *thinning* jenis ini menggunakan template untuk dicocokkan dengan citra yang akan di-*thinning*, di mana jika suatu kelompok piksel dari sebuah *image* cocok dengan template, maka piksel tengahnya akan di-*delete*. Metode ini cukup terkenal karena *reliability* dan keefektifannya. Salah satu algoritma *thinning* yang bersifat iteratif ini adalah metode *Stentiford*. (Engkamat, 2005).

Algoritma Zhang-Suen merupakan salah satu algoritma thinning yang cepat dan mudah diimplementasikan. Setiap iterasi dari metode ini terdiri dari dua sub-iterasi yang berurutan yang dilakukan terhadap *contour points* dari wilayah citra. *Contour point* adalah setiap piksel dengan nilai 1 dan memiliki setidaknya satu dari 8-neighbor yang memiliki nilai 0.

Algoritma Zhang Suen dibagi menjadi dua sub-iterasi. Pada iterasi pertama, *contour point* P1 akan dihapus jika semua kondisi ini dipenuhi :

a) $2 < B(P1) < 6$
b) $A(P1) = 1$
c) $P2*P4*P6 = 0$
d) $P4*P6*P8 = 0$

dimana A(P1) adalah jumlah dari transisi 0-1 pada urutan P2,P3,...,P8,P9.

Sedangkan B(P1) adalah jumlah tetangga dari P1 yang tidak 0, yaitu :

$$B(P1) = P2 + P3 + P4 + ... + P8 + P9$$

Pada iterasi kedua, kondisi a) dan b) sama dengan iterasi pertama, sedangkan kondisi c) dan d) diubah menjadi :

(c) $P2*P4*P8 = 0$
(d) $P2*P6*P8 = 0$

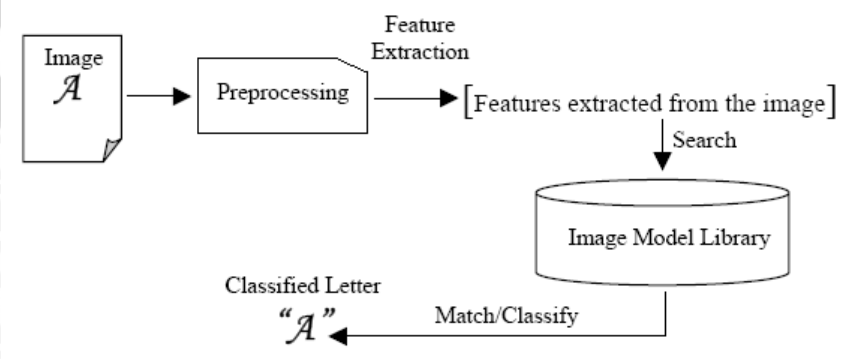
Iterasi pertama dilakukan terhadap semua *border pixel* di citra. Jika salah satu dari keempat kondisi di atas tidak dipenuhi atau dilanggar maka nilai piksel yang bersangkutan tidak diubah. Sebaliknya jika semua kondisi tersebut dipenuhi maka piksel tersebut ditandai untuk

penghapusan. Piksel yang telah ditandai tidak akan dihapus sebelum semua *border points* selesai diproses. Hal ini berguna untuk mencegah perubahan struktur data. Setelah langkah 1 selesai dilakukan untuk semua *border points* maka dilakukan penghapusan untuk titik yang telah ditandai (diubah menjadi 0). Setelah itu dilakukan langkah 2 pada data hasil dari langkah 1 dengan cara yang sama dengan langkah 1. (Engkamat, 2005)

2.3 Optical Character Recognition

OCR kependekan dari *Optical Character Recognition* yaitu suatu teknologi yang mampu mengalihkan karakter gambar ke dalam aplikasi program pengolah kata tanpa melakukan pengetikan ulang dari sumber dokumen yang dimaksud. Biasanya bila men-scan suatu teks atau gambar pada media cetak dengan alat scanner maka akan dihasilkan file dengan format bitmap. Tetapi dengan software OCR, huruf-huruf yang terdapat pada halaman tersebut bisa dimengerti dan dengan mudah untuk menambah atau mengurangi setiap karakter hurufnya. Artinya program ini dapat mengubah format gambar bitmap menjadi teks yang dapat dialihkan ke aplikasi program pengolah kata di komputer. (Kuswara, 2006).

OCR merupakan lingkup penelitian yang penting dalam *pattern recognition*. Sistem OCR digunakan untuk mengenali huruf alfabet, angka, atau karakter yang lain yang ada dalam digital image, tanpa campur tangan manusia. Sistem ini bekerja dengan mencari kemiripan diantara ekstraksi ciri dari karakter image yang telah diberikan dengan model image yang telah ada. Gambar 2.4 menunjukkan *basic process* dari sistem OCR.

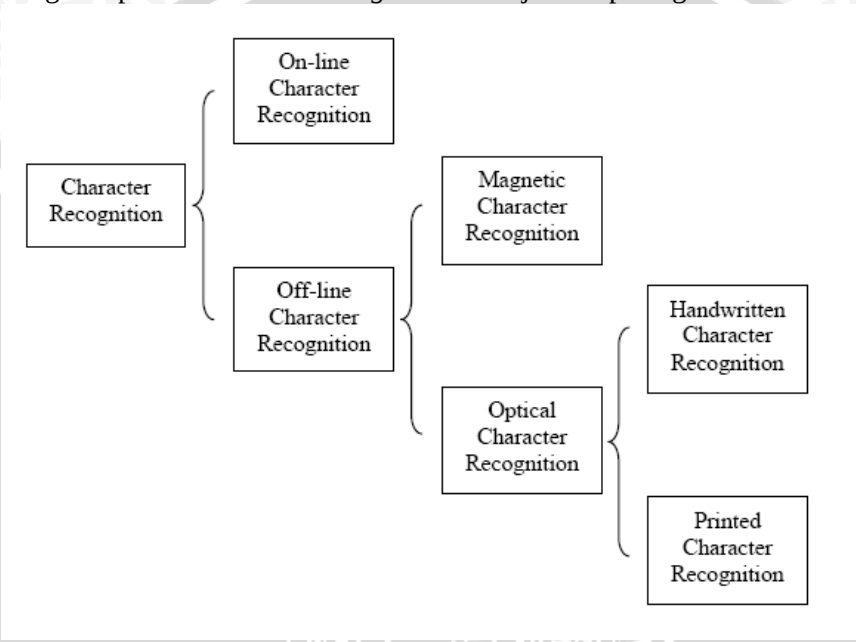


Gambar 2.4 *Basic process* dari sistem OCR

Pengenalan karakter dapat dibedakan menjadi dua bagian, yaitu :

- *On-line character recognition*
- *Off-line character recognition*

Pengelompokan *character recognition* ditunjukkan pada gambar 2.5.



Gambar 2.5 Pengelompokan *character recognition*

Perbedaan mendasar dari on-line dan off-line *character recognition* ini adalah bahwa *on-line character recognition* memiliki waktu pengurutan informasi yang kontekstual sedangkan *off-line character recognition* tidak memilikinya. Hal ini disebabkan karena perbedaan proses arsitektur dan metode.

Off-line character recognition dapat dikelompokkan menjadi dua, yaitu :

- *Magnetik Character Recognition* (MCR)
- *Optical Character Recognition* (OCR)

Pada MCR, karakter dicetak dengan tinta magnetik. MCR banyak digunakan di bank untuk mengecek autentifikasi. OCR dilakukan untuk mengenali karakter yang dihasilkan oleh alat optik, seperti scanner atau kamera. Karakter dibentuk menjadi piksel-piksel dalam image, dapat berupa huruf cetak atau tulisan tangan. (Chen, 2003)

Printed character recognition atau pengenalan huruf cetak adalah sebuah teknik dimana data yang berupa lembaran kertas dapat di-*scan* menggunakan *scanner* dan menghasilkan gambar yang pada komputer akan dikenali sebagai titik-titik. Titik-titik inilah yang kemudian diproses lebih lanjut dengan menggunakan algoritma tertentu menjadi karakter sehingga dapat dikenali dan diolah menjadi informasi.

OCR dapat dipandang sebagai bagian dari pengenal otomatis yang lebih luas yakni pengenal pola otomatis (*automatic pattern recognition*). Dalam pengenal pola otomatis, sistem pengenal pola mencoba mengenali apakah citra masukan yang diterima cocok dengan salah satu citra yang telah ditentukan. Ada banyak pendekatan yang dapat dipakai untuk mengembangkan pembuatan pengenal pola otomatis antara lain memakai pendekatan numerik, statistik, sintaktik, neural dan, aturan produksi (*rule-based*). Secara umum metode-metode tersebut dapat digolongkan menjadi dua kelompok metode yakni metode berbasis statistik dan metode berbasis struktur. Dalam metode yang berbasis statistik, setiap pola ditransformasi ke dalam vektor yang memakai ukuran dan karakteristik tertentu. Karakteristik ini seringkali lebih bersifat statistik misalnya distribusi piksel ataupun jarak piksel. Sedang dalam metode yang berbasis struktur, setiap pola yang diproses dinyatakan sebagai gabungan beberapa struktur elementer. Pengenalan selanjutnya dilakukan dengan mencocokkan komposisi struktur elementer dengan struktur yang sudah disimpan memakai aturan tertentu misalnya memakai pendekatan teori bahasa formal dan automata. (Chen, 2003)

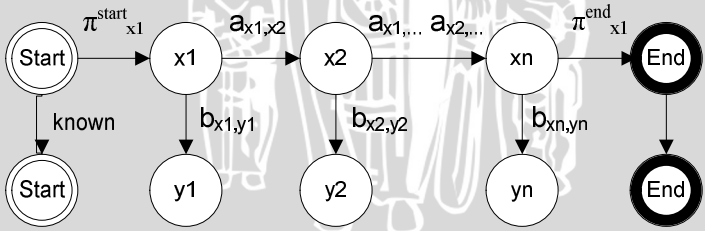
2.4 Metode *Hidden Markov Model*

Menurut Connel, 1996, *Hidden Markov Models* (HMM) merupakan sebuah model statistik dari suatu proses *Markov* dengan parameter-parameter tersembunyi (*hidden*) yang akan diperoleh dari parameter-parameter yang diamati (*observable*). Pada HMM *state*-nya tidak dapat diamati secara langsung (*hidden*), dimana setiap *state* memiliki distribusi peluang output yang mungkin muncul sebagai suatu set proses stokastik yang akan membentuk suatu deretan observasi.

Hidden Markov Model merupakan sebuah metode yang telah sukses digunakan untuk aplikasi pengenalan suara (*speech recognition*), dan sekarang, HMM mulai dikembangkan untuk aplikasi yang lain, diantaranya adalah untuk *Optical Character Recognition* (OCR).

Markov Model merupakan suatu metode yang memodelkan *signal* sebagai rangkaian *observable* output yang dihasilkan dari beberapa proses yang disebut *source*. *Signal* dapat bersifat diskrit maupun kontinu. Nilai *signal* bisa stabil (nilai statistiknya tidak berubah terhadap waktu) maupun nonstabil (nilai statistiknya berubah terhadap waktu). Dengan melakukan pemodelan terhadap *signal* secara benar, dapat dilakukan simulasi terhadap sumber dan pelatihan sebanyak mungkin melalui proses simulasi tersebut, sehingga model dapat diterapkan dalam sistem prediksi, sistem pengenalan, maupun sistem identifikasi. Secara garis besar model *signal* dapat dikategorikan menjadi dua, yaitu : model deterministik dan model statistikal. Model deterministik menggunakan nilai-nilai properti dari sebuah signal seperti : amplitudo, frekuensi, fase dari gelombang sinus. Sedangkan model statistikal menggunakan nilai-nilai statistik dari sebuah signal seperti : proses gaussian, proses poisson, proses Markov, dan proses Hidden Markov. (Connel, 1996)

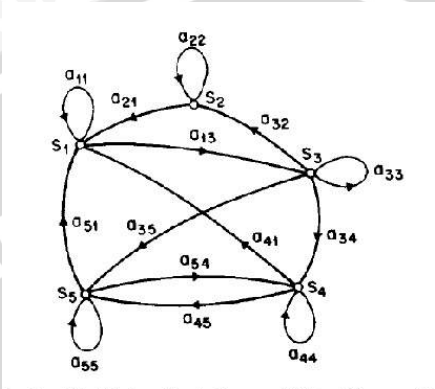
Dalam arsitektur umum tentang HMM, masing-masing bentuk oval menggambarkan sebuah variabel acak (random variable) yang berisikan nilai. Variabel Acak x(t) berisikan nilai sebuah variabel tersembunyi pada saat t. variabel acak y(t) berisikan nilai sebuah variabel yang dapat diamati (tidak tersembunyi) pada saat t. Anak panah menunjukkan ketergantungan kondisional. Dari diagram, jelas kiranya bahwa nilai x(t) hanya bergantung pada nilai x(t-1). Selain itu, nilai y(t) hanya bergantung pada x(t). (Velagapudi, tanpa tahun). Model arsitektur HMM ditunjukkan pada gambar 2.6.



Gambar 2.6 Model arsitektur HMM

Pada *hidden markov model*, sebuah proses stokastik yang berbeda dikombinasikan dengan sebuah markov chain untuk menghasilkan observasi yang ada diantara statistical properties dari berbagai proses stokastik dan statistical properties dari markov chain. *Hidden markov*

model efektif digunakan untuk pengenalan suara dan merupakan dasar untuk menampilkan pengenalan kata. Contoh markov chain ditunjukkan pada gambar 2.7.



Gambar 2.7 Markov Chain dengan 5 state dengan transisi state

Suatu HMM diskret dapat dinyatakan dengan :

- O = {O_1,O_2,...O_T} merupakan rangkaian observasi.
- T = panjang rangkaian observasi.
- Q = {q_1,q_2,...,q_N}, merupakan sejumlah state dalam model.
- N, merupakan jumlah state dalam model
- V = {v1,v2,...,vM} merupakan kemungkinan observasi.
- M, merupakan jumlah simbol observasi yang berbeda per state.

Metode HMM secara umum dapat dinotasikan dengan $\lambda = (A,B,\Pi)$, dimana A dan B adalah matriks, sedangkan Π adalah vektor.

Matriks A didefinisikan sebagai kumpulan dari semua *state transition probability distribution* untuk model seperti pada persamaan 2.2 :

$$a_{ij} = P[q_j \text{ at } t+1 \mid q_i \text{ at } t] \qquad (2.2)$$

dimana q_i adalah state pada model pada waktu ke t, q_j adalah kemungkinan state selanjutnya, dan a_{ij} adalah probabilitas bahwa model akan membuat transisi untuk state selanjutnya pada waktu selanjutnya.

Matriks B didefinisikan sebagai kumpulan dari semua *observation probability distribution function* untuk model seperti pada persamaan 2.3:

$$b_j(k) = P[v_k \text{ at } t \mid q_j \text{ at } t] \qquad (2.3)$$

dimana q_j adalah state pada model pada waktu t, v_k adalah observasi yang mungkin ada pada model, dan $b_j(k)$ adalah probabilitas untuk observasi ketika berada pada state q_j.

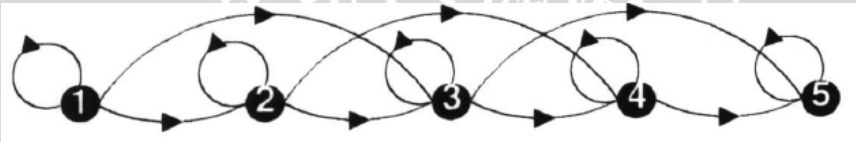
Vektor Π didefinisikan sebagai *initial state distribution* untuk model seperti pada persamaan 2.4:

$$\pi_i = P(q_i \text{ at } t=1) \quad (2.4)$$

dimana q_i adalah state pada model dan π_i adalah probabilitas bahwa model akan memulai pada state q_i saat *observation sequence* dimulai.

2.4.1 Left Right HMM

Model Left-Right HMM digunakan untuk pengenalan karakter karena beberapa alasan, antara lain : urutan feature ditentukan oleh first-order properti, setelah itu ditentukan oleh second-order properties. Metode left-Right parallel telah diaplikasikan pada pengenalan kata, juga menggunakan kedua order tersebut, dengan asumsi bahwa model tersebut *non-stationary*. Model Left-Right HMM ditunjukkan pada gambar 2.8.



Gambar 2.8 Simple Left-Right *Hidden Markov Model*

2.4.2 HMM pada *Optical Character Recognition*

HMM dapat digunakan untuk proses pengenalan karakter dan klasifikasi, dimana model yang ada akan menghasilkan suatu rangkaian simbol. Untuk mengevaluasi model tersebut, maka harus dilakukan perhitungan $P(O|\lambda)$, yang merupakan likelihood dari rangkaian observasi, O, yang dihasilkan dengan memberikan model, λ. Untuk setiap rangkaian observasi yang dihasilkan oleh model, dan diketahui rangkaian statenya, maka probabilitasnya dapat dihitung dengan algoritma *forward backward.*

2.5 Algoritma Forward Backward

Algoritma *forward-backward* ini digunakan untuk mengetahui probabilitas dari rangkaian observasi yang dihasilkan oleh model. Variabel α merupakan variabel yang diberikan untuk forward, sedangkan variabel backward dilambangkan sebagai β.

Probabilitas forward, $\alpha_t(i)$, sebagai probabilitas dari rangkaian observasi dari waktu ke 1 sampai dengan waktu ke-t, dan model yang berada pada state q_i pada waktu t. Hal ini ditunjukkan pada persamaan 2.5, 2.6, dan 2.7.

- Inisialisasi :

$$\alpha_t(i) = \pi i b i(Ot),\ 1 \le i \le N \qquad (2.5)$$

- Untuk t = 1,2,…,t-1 :

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} | b_j(O_{t+1}),\ 1 \le j \le N \qquad (2.6)$$

- Probabilitas, probabilitas dari rangkaian observasi, O, dapat diselesaikan dengan hitungan :

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i) \qquad (2.7)$$

Probabilitas backward, $\beta_t(i)$, dapat didefinisikan sebagai probabilitas dari rangkaian observasi dari waktu t+1 sampai waktu T, dan model berada pada di state q_i pada waktu t. Hal ini ditunjukkan pada persamaan 2.8, 2.9, dan 2.10.

- Inisialisasi :

$$\beta_t(i) = 1,\ 1 \le i \le N \qquad (2.8)$$

- Untuk t = T-1, T-2,…,1 :

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}(j)),\ 1 \le j \le N \qquad (2.9)$$

- Probabilitas, probabilitas dari rangkaian observasi, O, dapat diselesaikan dengan hitungan :

$$P = \sum_{j=1}^{N} \pi i b t(Ot) \beta t(i) \qquad (2.10)$$

Dengan algoritma *forward-backward* ini, dapat diketahui probabilitas dari rangkaian observasi yang dihasilkan oleh model. Untuk dapat memilih satu model terbaik dari yang lain, maka harus dilakukan perbandingan untuk setiap model, manakah yang menghasilkan rangkaian observasi terbaik.

2.6 Algoritma Baum Welch

Algoritma *Baum Welch* digunakan untuk mengestimasi model parameter HMM yang telah dibuat. Probabilitas state qi pada waktu t dapat dilihat seperti pada persamaan 2.11 dan 2.12.

$$\gamma_t(i) = \frac{\alpha t(i)\beta t(i)}{P(O|\lambda)} \quad (2.11)$$

$$\varepsilon t(i,j) = \frac{\alpha t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \quad (2.12)$$

Estimasi model parameter ditunjukkan pada persamaan 2.13, 2.14, 2.15.

Initial probabilities

$$\pi i = \gamma_t(i) \quad (2.13)$$

Transition probabilities

$$aij = \frac{\sum_{t=1}^{T-1} \varepsilon t(i,j)}{\sum_{t=1}^{T-1} \gamma t(i)} \quad (2.14)$$

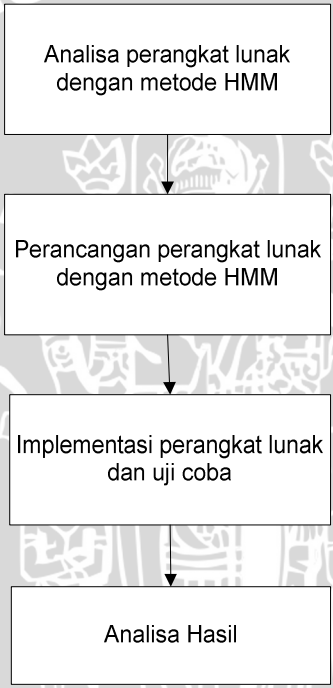
Emission probabilities

$$bj(k) = \frac{\sum_{t}^{T} \gamma t(j) + Ot}{\sum_{t=1}^{T} \gamma t(j)} \quad (2.15)$$

Estimasi model parameter tersebut digunakan untuk mendapatkan nilai baru dari parameter HMM.

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini, akan dibahas tentang penggunaan metode yang digunakan dalam pembuatan perangkat lunak Pengenalan Huruf Cetak dengan metode *Hidden Markov Model*. Langkah-langkah yang dilakukan untuk membuat aplikasi perangkat lunak Pengenalan Huruf Cetak dengan metode *Hidden Markov Model* ditunjukkan pada gambar 3.1.



Gambar 3.1 Langkah-langkah pembuatan perangkat lunak

3.1 Analisa Perangkat Lunak

3.1.1 Deskripsi Umum Perangkat lunak

Dalam penelitian ini, akan dibangun suatu perangkat lunak yang dapat digunakan untuk melakukan pengenalan huruf pada dokumen yang telah discan, yaitu berupa file citra dengan format bitmap, yang kemudian akan diubah menjadi file teks sebagai implementasi dari

sistem pengenalan huruf cetak dengan menggunakan metode *Hidden Markov Model*.

Data yang digunakan untuk pengujian pada penelitian ini adalah citra yang berisi teks dengan format bitmap. Karakter yang akan dikenali terdiri dari 26 jenis huruf besar, 26 jenis huruf kecil, dan 10 jenis angka. Daftar karakter yang digunakan dalam penelitian ini ditunjukkan pada tabel 3.1.

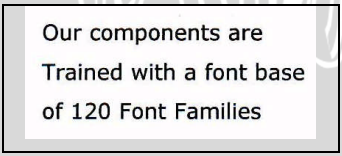
Tabel 3.1 Daftar Karakter

Keterangan	Daftar Karakter	Jumlah
Huruf Kapital	A,B,C,D,E,.....X,Y,Z	26 jenis
Huruf Kecil	a,b,c,d,e,.....x,y.z	26 jenis
Angka	0,1,2,3,4,5,6,7,8,9	10 jenis

Data masukan yang digunakan terdiri dari dua macam, yaitu citra data latih dan citra data uji. Untuk pelatihan, data yang digunakan merupakan citra huruf tunggal. Contoh citra data latih ditunjukkan pada gambar 3.2.

Gambar 3.2 Contoh Citra data latih

Citra data uji adalah citra yang berisi kumpulan karakter hasil dari dokumen yang telah discan. Untuk pengujian, data yang digunakan merupakan citra yang berisi kumpulan huruf yang membentuk kata atau kalimat. Contoh citra data uji ditunjukkan pada gambar 3.3.

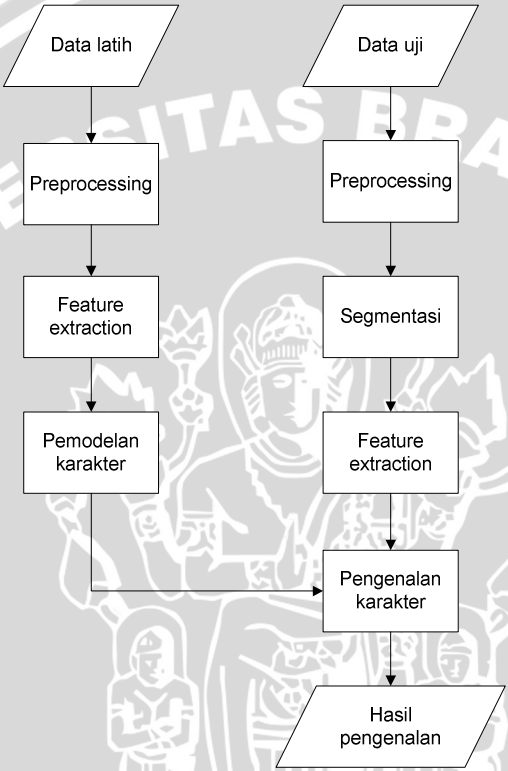


Gambar 3.3 Contoh Citra data uji

3.2 Perancangan Perangkat Lunak

3.2.1 Blok Diagram Sistem

Secara garis besar, sistem pengenalan karakter ini terbagi menjadi dua tahap, yaitu tahap pelatihan dan tahap pengujian. Blok diagram perangkat lunak ditunjukkan pada gambar 3.4.



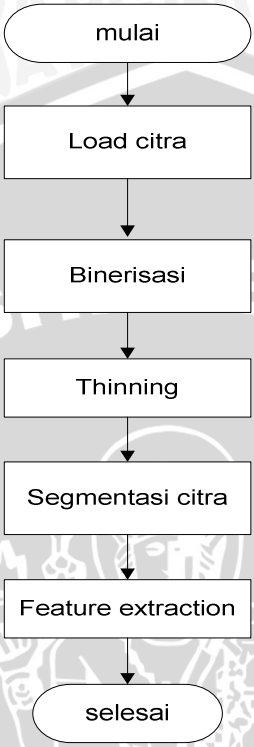
Gambar 3.4 Blok diagram sistem pengenalan karakter

Dari gambar 3.4, dapat diketahui bahwa sistem pengenalan karakter ini terdiri dari dua tahap, yaitu tahap pelatihan dan tahap pengujian atau pengenalan. Pada tahap pelatihan, data latih yang dimasukkan akan melalui *preprocessing*, serta *feature extraction* untuk mengambil ciri dari setiap karakter. Dengan demikian, akan didapatkan model data latih yang nantinya akan digunakan untuk proses pengenalan karakter. Sedangkan pada proses pengujian, data uji yang dimasukkan akan dilakukan *preprocessing*, kemudian dilanjutkan dengan proses segmentasi. Berbeda dengan data latih, pada data uji ini dilakukan

proses segmentasi karena citra teks yang dijadikan sebagai data uji terdiri dari rangkaian karakter yang membetuk kata. Proses selanjutnya adalah *feature extraction* yang mengambil ciri dari setiap karakter. Setelah itu, dilakukan pengenalan terhadap data uji dengan cara membandingkan nilai kemiripan antara karakter data uji dengan karakter data latih. Setelah semua proses selesai dilakukan, maka sistem akan mengeluarkan hasil pengenalan karakter.

3.2.2 Perancangan Proses Pengolahan Citra

Proses pengolahan citra dalam perancangan perangkat lunak ini terdiri dari empat bagian. Proses tersebut antara lain : binerisasi, *thinning*, segmentasi citra, dan *feature extraction*. Proses pertama yang dilakukan adalah memasukkan citra yang akan dikenali ke dalam sistem. Selanjutnya, dilakukan proses *grayscaling* yang berfungsi untuk merubah citra warna menjadi *grayscale*. Setelah itu, citra *grayscale* tersebut diubah menjadi citra biner dengan menggunakan *thresholding*. *Thresholding* berfungsi untuk memisahkan obyek yang akan dikenali dengan latar belakang citra. Proses selanjutnya untuk pengolahan citra adalah melakukan *thinning* pada citra biner untuk mengambil kerangka dari obyek yang akan dikenali. Pemisahan obyek pada citra dilakukan dengan menerapkan segmentasi pada citra. Setelah proses segmentasi, maka langkah terakhir pada pengolahan citra ini adalah *feature extraction*, yaitu proses mengambil ciri-ciri khusus dari setiap karakter sehingga mempermudah proses pengenalan. Untuk lebih jelasnya, proses pengolahan citra yang dilakukan ditunjukkan pada gambar 3.5.



Gambar 3.5 Proses pengolahan citra

3.2.2.1 Binerisasi

Binerisasi meliputi *grayscaling* dan *thresholding.*

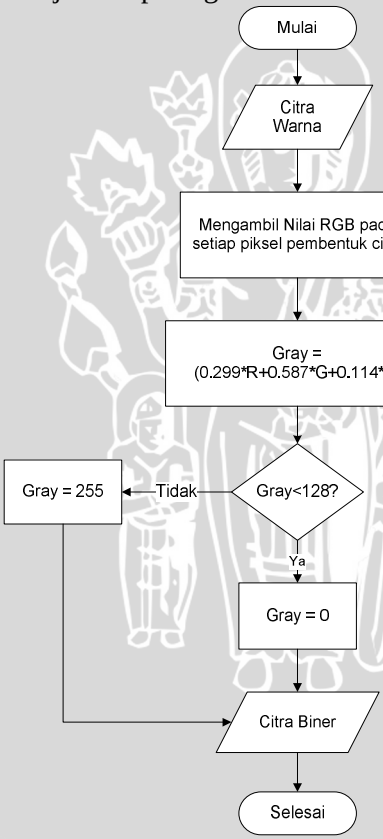
a. ***Grayscaling***

Grayscaling adalah proses perubahan nilai *piksel* dari warna (RGB) menjadi *gray-level* (Gonzalez, 2002). Pada dasarnya proses ini dilakukan dengan meratakan nilai *piksel* dari 3 nilai RGB menjadi 1 nilai. Untuk memperoleh hasil yang lebih baik, nilai *piksel* tidak langsung dibagi menjadi 3 melainkan terdapat persentasi dari masing-masing nilai. Salah satu persentasi yang sering digunakan adalah 29,9% dari warna merah (*Red*), 58,7% dari warna hijau (*Green*), dan 11,4% dari warna biru (*Blue*). Nilai *piksel* didapat dari jumlah persentasi 3 nilai tersebut.

b. ***Thresholding***

Operasi pengambangan (*thresholding*) digunakan untuk mengkonversi dari citra *greyscale* ke citra biner. Operasi ini digunakan untuk memisahkan antara obyek dan latar belakang. Operasi pengambangan ini mengelompokkan nilai derajat keabuan setiap piksel menjadi dua yaitu hitam (1) dan putih (0). Dalam perangkat lunak pengenalan teks ini, operasi pengambangan dapat digunakan untuk memisahkan antara obyek yang dalam hal ini berupa huruf yang akan dikenali dengan latar belakang. Pada gambar 3.7 ini, Gray merupakan nilai threshold yang menjadi tolok ukur pengubahan nilai tiap piksel, apakah menjadi 1 (hitam) atau 0 (putih).

Proses binerisasi ditunjukkan pada gambar 3.6



Gambar 3.6 *Flowchart* Proses Binerisasi

3.2.2.2 Thinning

Penipisan pola (*thinning*) digunakan untuk mengambil kerangka dari suatu obyek. Algoritma *Thinning* yang digunakan dalam penelitian ini adalah algoritma Zhang Suen. Algoritma ini menggunakan metode parallel yang berarti bahwa nilai baru untuk sebuah piksel dari iterasi sebelumnya. Proses thinning dapat dijelaskan sebagai berikut :

- Sub-iterasi 1 :

a. Memasukkan citra biner C(i,j).
b. Menginisialisasi nilai piksel pada citra. Bila piksel merupakan bagian dari obyek, maka piksel bernilai 1. Sebaliknya, apabila piksel merupakan bagian dari background, maka piksel bernilai 0. Nilai piksel dari citra disimpan pada sebuah array 2 dimensi dengan tipe data integer.
c. Mencari 8-neighbor (tetangga), yaitu $p_2,p_3,\ldots,p_9$ dari setiap p_1. Nilai dari piksel p_1-p_9 disimpan dalam sebuah array.
d. Memeriksa apakah tiap piksel p_1 bernilai 1.
e. Memeriksa apakah tetangga (p_2-p_9) dari p1 yang bernilai 0 lebih dari sama dengan 1.
f. Menghitung $B(p_1)$, dimana $B(p_1)$ adalah jumlah tetangga-tetangga dari p_1 yang bukan 0, sehingga $B(p_1) = (p_2+p_3+p_4+p_5+p_6+p_7+p_8+p_9)$.
g. Menghitung $A(p_1)$, dimana $A(p_1)$ adalah jumlah transisi 0 ke 1 dalam urutan $p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9$.
h. Memeriksa apakah $B(p_1)$ lebih dari sama dengan 2 dan kurang dari sama dengan 6, $A(p_1)=1$, $p_2.p_4.p_6=0$, dan $p_4,p_6,p_8=0$.
i. Menandai setiap piksel p1 yang memenuhi syarat pada nomer 8.
j. Menghapus setiap piksel p_1 yang ditandai dengan merubah warnanya menjadi putih.

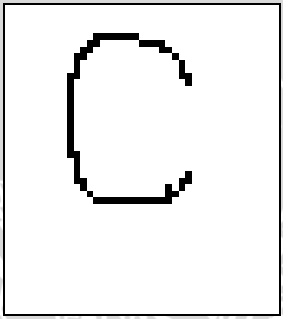
- Sub-iterasi 2 :

a. Memasukkan citra hasil thinning pada sub-iterasi 1.
b. Melakukan langkah (b)-(g) pada sub-iterasi 1.
c. Memeriksa apakah $B(p_1)$ lebih dari sama dengan 2 dan kurang dari sama dengan 6, $A(p_1)=1$, $p_2.p_4.p_8=0$, dan $p_2.p_6.p_8=0$.
d. Menandai setiap piksel p_1 yang memenuhi syarat pada nomer 8.
e. Menghapus setiap piksel p_1 yang ditandai dengan merubah warnanya menjadi putih.

p3 i-1,j+1	p2 i,j+1	p9 i+1,j+1
p4 i-1,j	p1 i,j	p8 i+1,j
p5 i-1,j-1	p6 i,j-1	p7 i+1,j-1

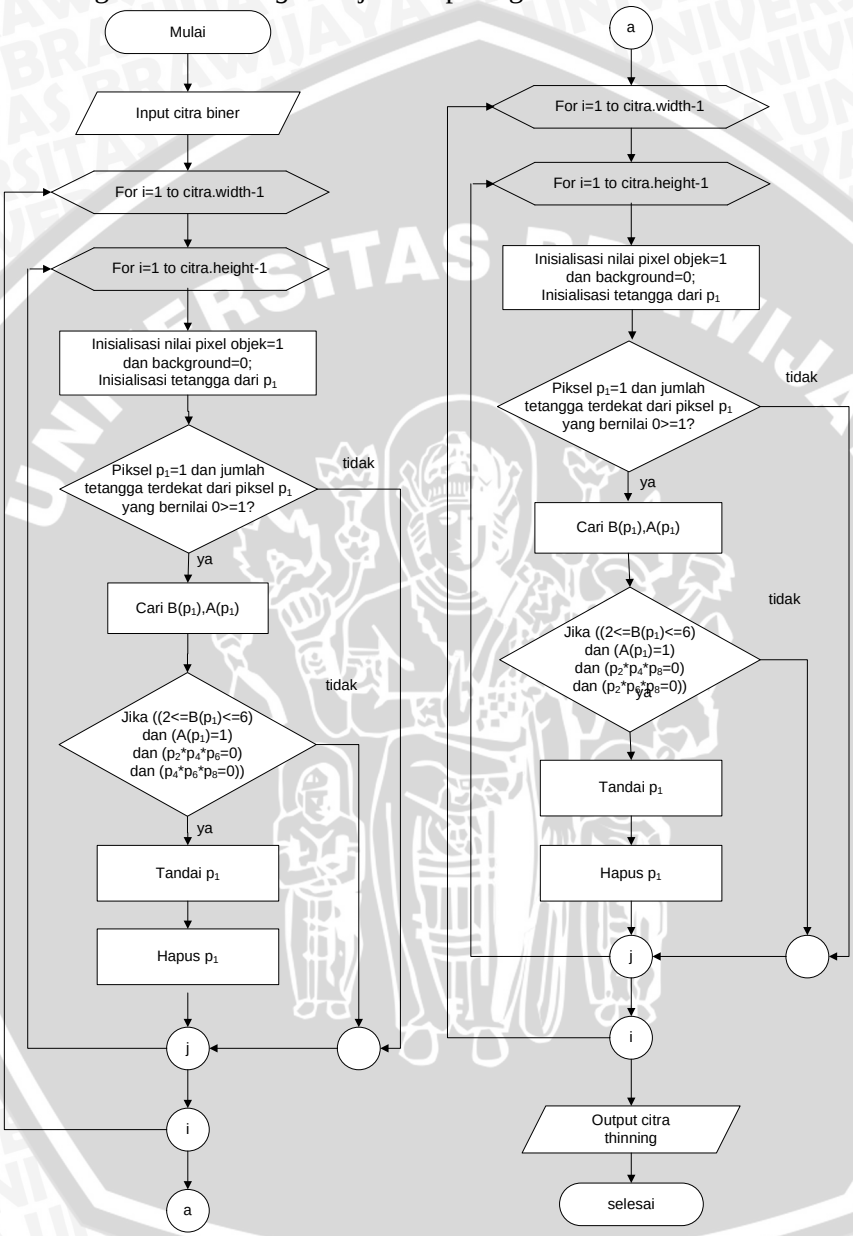
Gambar 3.7 8-neighbor connectivity

Contoh citra hasil *thinning* dapat dilihat pada gambar 3.8.



Gambar 3.8 Citra hasil *thinning*

Algoritma *thinning* ditunjukkan pada gambar 3.9.



Gambar 3.9 *Flowchart* Proses *thinning*

3.2.2.3 Segmentasi Citra

Segmentasi digunakan untuk memisahkan suatu obyek dengan obyek lainnya atau untuk memisahkan obyek dari latar belakang sehingga dapat digunakan untuk pemrosesan citra selanjutnya. Dalam penelitian ini, akan dilakukan segmentasi baris dan segmentasi karakter.

a. Segmentasi baris

Segmentasi baris digunakan untuk membagi citra sesuai dengan banyaknya baris teks yang akan dikenali. Segmentasi baris ini dilakukan dengan penelusuran setiap piksel dari citra untuk menghitung jumlah baris dan menyimpan nilai koordinat atas dan koordinat bawah pada setiap baris.

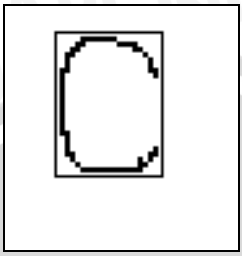
b. Segmentasi karakter

Setelah dilakukan proses segmentasi kata, selanjutnya dilakukan segmentasi karakter untuk memisahkan kata menjadi karakter-karakter. Proses segmentasi ini dilakukan untuk menghitung jumlah karakter dan menyimpan koordinat baris_bawah=y1, baris_atas=y2, batas_kiri=x1, dan batas_kanan=x2 untuk setiap karakter. Pada proses segmentasi karakter ini, akan dilakukan dengan dua cara, yaitu : segmentasi secara lurus vertikal dan melalui proses deteksi tepi (*edge detection*).

Segmentasi secara lurus vertikal dilakukan dengan menelusuri secara vertikal pada setiap baris teks. Pada saat dilakukan penelusuran, piksel hitam yang pertama kali ditemukan pada akan menjadi batas kiri pada baris tersebut. Selanjutnya dilakukan penelusuran lagi sampai tidak ditemukan piksel hitam pada suatu kolom, yang berarti bahwa kolom tersebut merupakan batas kanan dari karakter yang telah ditemukan. Penelusuran tersebut dilakukan terus sampai ditemukan semua karakter yang terdapat dalam satu baris.

Deteksi tepi dilakukan dengan menelusuri piksel hitam dari setiap karakter yang telah didapatkan dari proses segmentasi secara lurus vertikal. Piksel-piksel hitam hasil dari penelusuran ini disimpan dan selanjutnya akan digunakan unuk proses ekstraksi ciri.

Hasil dari seluruh proses segmentasi tersebut adalah posisi koordinat, lebar dan tinggi dari masing-masing karakter. Setiap karakter hasil dari segmentasi ini akan dinormalisasikan menjadi gambar dengan ukuran 50x50 piksel. Contoh citra hasil normalisasi ditunjukkan pada gambar 3.10

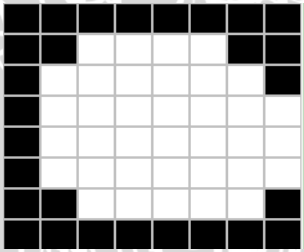


Gambar 3.10 Citra hasil normalisasi

3.2.2.4 Feature Extraction

Setiap karakter memiliki ciri-ciri khusus yang membedakan antara satu karakter dengan karakter yang lainnya. Proses *feature extraction* digunakan untuk mengetahui ciri-ciri khusus yang ada pada karakter-karakter tersebut.

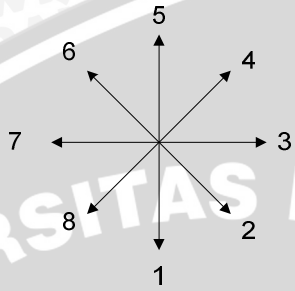
Setelah dilakukan tahap *preprocessing* pada citra masukan, maka telah dihasilkan sebuah citra biner. Terdapat dua jenis data yang didapat dari citra biner tersebut, yaitu nilai 1 dan 0. Nilai 1 digunakan untuk merepresentasikan warna hitam, sedangkan nilai 0 untuk merepresentasikan warna putih.



Gambar 3.11 Citra hasil pemetaan

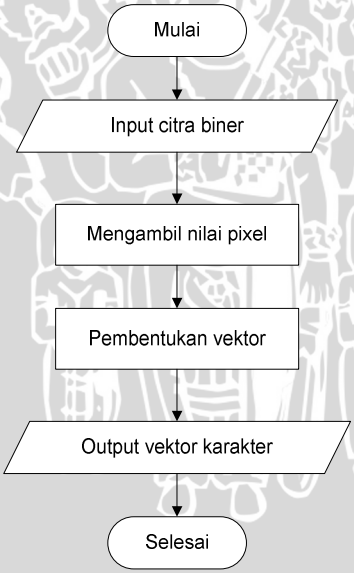
Feature extraction yang digunakan adalah *direction features*. Jadi pada metode ini, digunakan 8 arah untuk mengkodekan setiap pixel dari

karakter. Kode-kode yang digunakan yaitu berupa angka 1 sampai dengan 8, seperti ditunjukkan pada gambar 3.12



Gambar 3.12 *Direction Features*

Flowchart untuk *feature extraction* ditunjukkan pada gambar 3.13.



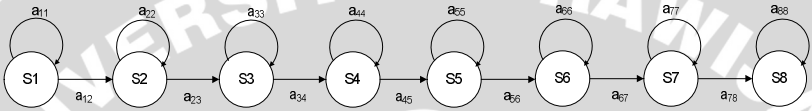
Gambar 3.13 *Flowchart* Proses *feature extraction*

3.2.3 Pengenalan dengan metode *Hidden Markov Model*

Suatu *hidden markov model* diskret dari N state dan M simbol observasi, dapat digambarkan dengan state matriks peluang transisi,

A=[a_{ij}], matriks peluang observasi, B=[$b_j(k)$], dan probabilitas state awal, $\pi = [\pi_1, \pi_2, \ldots, \pi_N]$. Model HMM yang digunakan untuk melakukan pengenalan karakter ini adalah model *left-to-right* HMM. Pada model *left-to-right* ini, terdapat aturan $a_{ij}=0$ untuk j<i dan j>i+1. Dengan state 1 merupakan state awal untuk model *left-to-right*, maka $\pi = [1,0,\ldots,0]$.

Pada penelitian ini, struktur HMM yang dipilih adalah struktur *left-to-right* dengan 8 state. Struktur *Left-to-right* dengan 8 state disajikan dalam gambar 3.14.



Gambar 3.14 Model *left-to-right* HMM

Pada struktur *left-to-right* HMM, matriks peluang transisi, A, dapat diperoleh dengan aturan bahwa nilai $a_{ij}=0$ untuk j<i dan j>i+1 serta aturan $\sum_{j=1}^{N} a_{ij} = 1$, 1≤i≤N. Nilai ini telah memenuhi standar *stochastik / probabilistic*. Dengan demikian, didapatkan matriks peluang transisi seperti pada gambar 3.16.

$$A=\{a_{ij}\}=\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{66} & a_{67} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{66} & a_{66} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Gambar 3.15 Model Matriks peluang transisi

$$A=\{a_{ij}\}=\begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Gambar 3.16 Model Awal Matriks peluang transisi

Untuk vektor distribusi state awal ditetapkan bahwa $\pi_1=1$ dan $\pi_i=0$ untuk $i\neq1$. Hal ini berarti bahwa state 1 adalah state awal untuk model *left-to-right*. Dengan demikian, didapatkan vektor distribusi awal state seperti pada gambar 3.17.

$$\Pi = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Gambar 3.17 vektor distribusi

Matriks peluang observasi, B, merupakan matriks dari model HMM yang mengobservasi simbol V_k yang ada dalam state j. Untuk menentukan model awal Matriks peluang observasi, digunkanan rumus $bj(k) = \frac{1}{M} \quad \forall j, k$ dimana $\sum_{k=1}^{M} bj(k) = 1$.

Dengan demikian, didapatkan matriks peluang observasi seperti pada gambar 3.18.

$$
B = \begin{bmatrix}
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125
\end{bmatrix}
$$

Gambar 3.18 Model Awal Matriks peluang observasi

Dengan struktur tersebut, selanjutnya adalah melatih model data training.

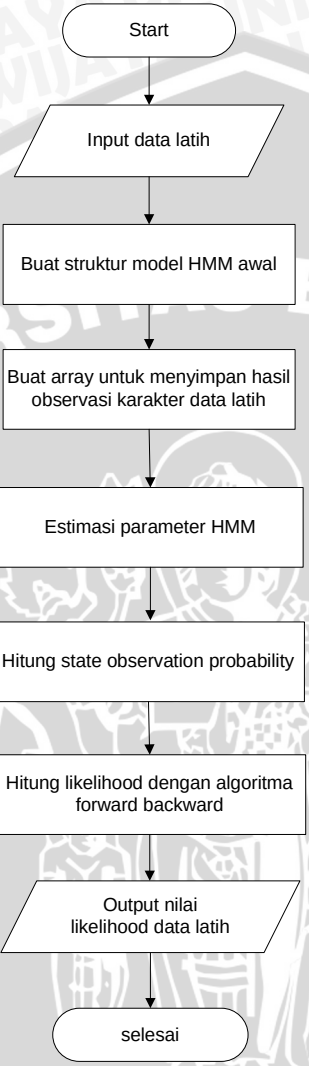
a. Pelatihan

Citra yang digunakan untuk pelatihan adalah citra data tunggal dari huruf kapital A-Z, huruf kecil a-z, serta angka 0-9. Data latih yang digunakan diambil dari 3 jenis huruf yang berbeda. Sehingga total terdapat 186 karakter.

Dalam pelatihan karakter ini, terlebih dahulu dibuat model awal HMM, seperti ditunjukkan pada gambar 3.16, 3.17, dan 3.18. Model awal HMM tersebut dihitung dengan menggunakan algoritma *Baum-Welch* sehingga didapatkan model HMM yang baru.

Untuk mengevaluasi model HMM yang telah dibuat, terlebih dahulu dihitung nilai $P(O|\lambda)$, yang merupakan *likelihood* dari rangkaian observasi,O, yang dihasilkan oleh model, λ. Probabilitas dari rangkaian observasi ini dapat dihitung dengan persamaan (2.5). *Likelihood* ini dapat dihitung dengan menggunakan algoritma *Forward-Backward*.

Dengan algoritma *forward-backward,* dapat diketahui probabilitas dari rangkaian observasi yang dihasilkan oleh model. Untuk dapat memilih satu model terbaik dari yang lain, maka harus dilakukan perbandingan untuk setiap model, manakah yang menghasilkan rangkaian observasi terbaik. Setelah semua perhitungan di atas dilakukan, maka akan didapatkan nilai *likelihood* dari data pelatihan. Proses pelatihan ditunjukkan pada gambar 3.19.



Gambar 3.19 *Flowchart* proses pelatihan

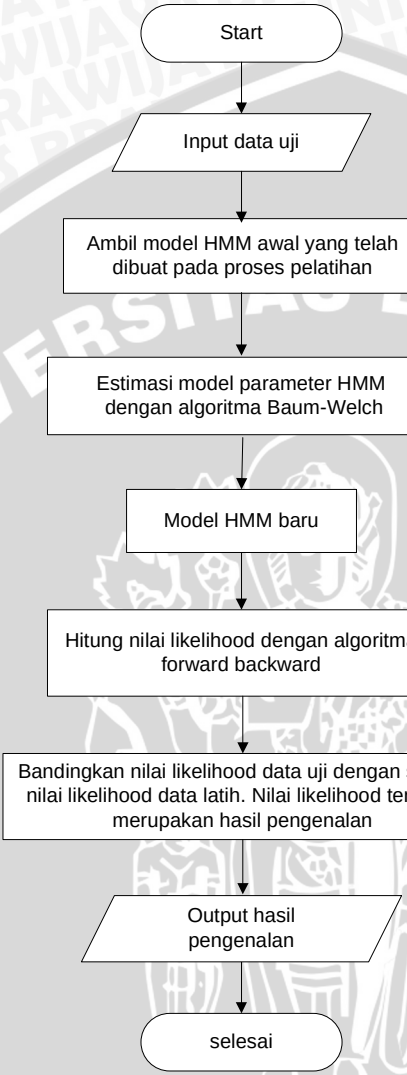
b. Pengenalan

Pada proses pengenalan, akan diberikan data uji berupa citra yang berisi kumpulan huruf yang membentuk kata atau kalimat. Pada citra data uji tersebut,akan dilakukan proses segmentasi terlebih dahulu. Segmentasi yang dilakukan dibagi menjadi dua bagian, yaitu segmentasi baris dan segmentasi karakter. Setelah dilakukan proses segmentasi dan didapatkan karakter yang akan dikenali, selanjutnya dilakukan perhitungan HMM untuk mencari nilai *likelihood* terbaik untuk setiap karakter.

Proses perhitungan model HMM pada tahap pengenalan ini hampir sama dengan proses pelatihan, yaitu setelah mengambil model HMM awal yang telah dibuat, selanjutnya dihitung dengan algoritma *Baum-Welch* sehingga didapatkan model HMM yang baru.

Dari model HMM baru tersebut, didapatkan nilai matriks peluang observasi karakter yang akan diuji. Proses selanjutnya adalah menghitung nilai *likelihood* dengan algoritma *forward backward*.

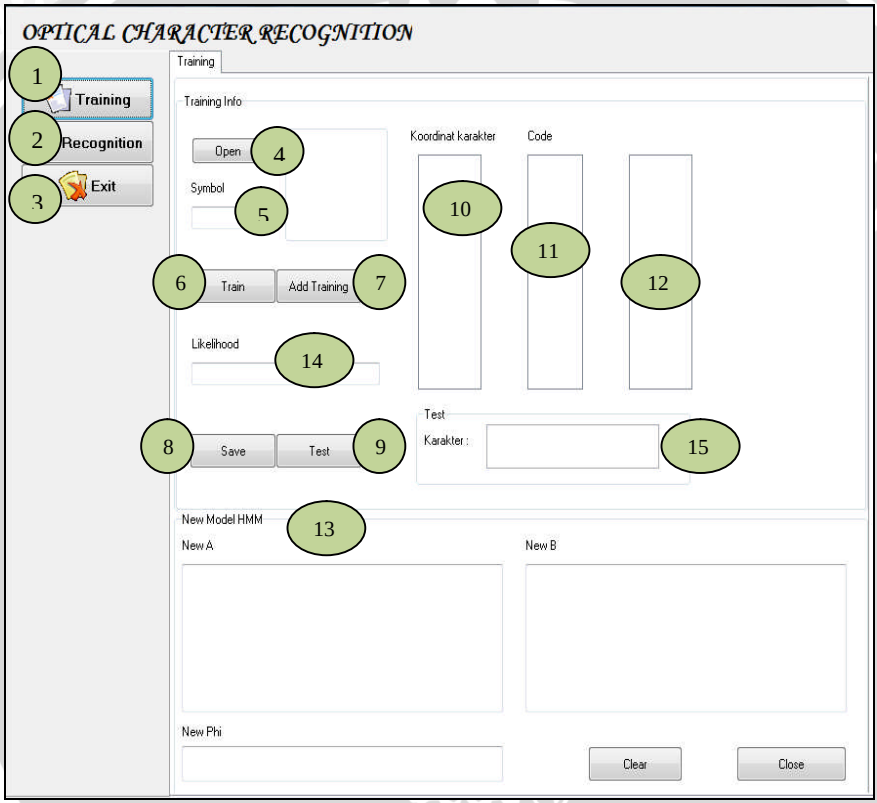
Setelah dilakukan perhitungan dengan algoritma *forward backward* dan didapatkan nilai likelihood, langkah selanjutnya adalah membandingkan nilai likelihood karakter yang diuji dengan nilai likelihood karakter yang telah dilatihkan. Karakter yang memiliki tingkat kemiripan terbesar yang akan dipilih sebagai karakter hasil pengenalan. Langkah-langkah dari proses pengenalan karakter ini ditunjukkan pada gambar 3.20.



Gambar 3.20 *Flowchart* proses pengenalan

3.3 Perancangan *User Interface*

Sistem ini dibagi menjadi dua bagian, yaitu form Pengenalan dan form pelatihan. Form pengenalan menampilkan citra yang akan digunakan dalam uji coba dan menampilkan hasil pengenalannya, sedangkan form pelatihan digunakan untuk menyimpan data karakter yang dilatihkan. Rancangan *user interface* ditunjukkan pada gambar 3.21 dan gambar 3.22

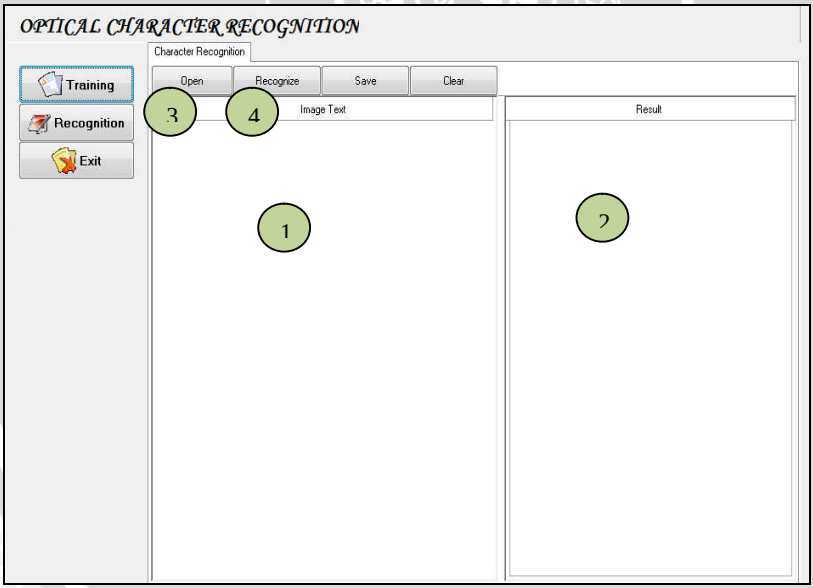


Gambar 3.21 *User Interface* Form Pelatihan

Keterangan :

1. Tombol Training : untuk membuka tab Pelatihan.
2. Tombol Recognition : untuk membuka tab pengenalan.
3. Tombol Exit : untuk keluar dari program.
4. Tombol Open : untuk membuka citra data latih.

5. Symbol : memasukkan symbol karakter yang dilatihkan.
6. Tombol Train : untuk menjalankan preprocessing citra, menghitung parameter HMM, dan menghitung nilai likelihood karakter data latih.
7. Tombol AddTraining : untuk memasukkan ciri-ciri karakter ke dalam array.
8. Tombol save : untuk menyimpan nilai likelihood dari karakter yang dilatihkan.
9. Tombol Test : untuk menguji karakter yang telah disimpan.
10. ListBox untuk menampilkan koordinat tepi dari karakter.
11. ListBox untuk menampilkan kode hasil dari ekstraksi ciri.
12. ListBox untuk menampilkan simbol karakter yang telah disimpan.
13. GroupBox New Model HMM unuk menampilkan nilai parameter HMM yang baru.
14. Likelihood : menampilkan nilai likelihood karakter latih.
15. Test : menampilkan hasil pengenalan karakter latih.



Gambar 3.22 *User Interface* Form Pengenalan

Keterangan :

1. Untuk menampilkan citra data uji.

2. Untuk menampilkan hasil pengenalan.
3. Tombol Open : untuk membuka citra data uji.
4. Tombol Recognize : untuk melakukan proses pengenalan.

3.4 Rancangan Uji Coba

Setelah proses pengembangan sistem, tahap selanjutnya yang dilakukan adalah membuat rancangan uji coba. Proses ini dilakukan dengan melakukan uji coba terhadap sistem yang telah dibangun, yang meliputi proses pelatihan dan pengenalan karakter pada citra dokumen.

Pada penelitian ini, jumlah karakter yang akan dikenali adalah 62 karakter yang meliputi :

a. 26 karakter untuk huruf kapital
b. 26 karakter untuk huruf kecil
c. 10 karakter untuk angka

Pada saat pelatihan, akan digunakan karakter yang berasal dari tiga jenis huruf yang berbeda, yaitu Times New Roman, Arial, dan Comic Sans MS. Masing-masing jenis huruf terdiri dari 62 karakter, sehingga total karakter yang digunakan dalam pelatihan adalah sejumlah 186 karakter.

Tingkat keberhasilan dalam pengenalan karakter dapat dihitung dengan cara :

$$\text{Keberhasilan} = \frac{\text{Jumlah karakter yang dikenali benar}}{\text{Jumlah karakter yang diuji}} \times 100\%$$

Proses pengujian dilakukan menjadi dua tahap, yaitu pengujian terhadap data yang telah dilatihkan (data pelatihan) dan pengujian terhadap data yang yang belum dilatihkan (data non-pelatihan). Data pelatihan adalah citra yang berisi karakter dengan dengan jenis huruf Arial, Comic Sans MS, dan Times New Roman. Data non-pelatihan adalah citra yang berisi karakter dengan jenis huruf Berlin Sans FB, Microsoft Sans Serif, dan Verdana.

Uji coba pertama dilakukan terhadap citra yang berisi karakter-karakter yang telah dilatihkan, yaitu citra karakter dengan jenis huruf yang sama dengan jenis huruf yang telah dilatihkan. Setiap karakter data

latih akan diberikan 3 contoh sampel, sehingga pada uji coba pertama ini setiap jenis huruf cetak akan memiliki 186 sampel yang terdiri dari :

a. 3 sampel karakter untuk huruf kapital = 3 x 26 = 78
b. 3 sampel karakter untuk huruf kecil = 3 x 26 = 78
c. 3 sampel karakter untuk angka = 3 x 10 = 30

Pengenalan ini akan dipisahkan berdasarkan jenis huruf masing-masing. Rancangan uji coba pertama ini ditunjukkan pada tabel 3.2.

Tabel 3.2 Rancangan tabel pengenalan karakter jenis huruf pelatihan

Jenis Huruf	Jumlah karakter	Jumlah benar	Jumlah salah	Keberhasilan (%)
Rata-rata				

Uji coba kedua dilakukan terhadap citra teks dengan jenis huruf yang tidak sama dengan data pelatihan. Jenis huruf yang digunakan pada pengujian ini adalah Berlin Sans FB, Microsoft Sans Serif, dan Verdana. Pengenalan akan dilakukan pada citra teks yang telah discan. Rancangan uji coba kedua ini ditunjukkan pada tabel 3.3.

Tabel 3.3 Rancangan tabel pengenalan karakter jenis huruf non-pelatihan

Jenis Huruf	Jumlah karakter	Jumlah benar	Jumlah salah	Keberhasilan (%)
Rata-rata				

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Sistem

Lingkungan sistem yang akan dijelaskan dalam subbab ini adalah lingkungan perangkat keras dan perangkat lunak yang digunakan untuk pengembangan sistem.

4.1.1 Lingkungan Perangakat Keras

Perangkat keras yang digunakan dalam pengembangan sistem pengenalan karakter ini adalah sebagai berikut :

1. Processor Intel® Dual Core CPU T4200
2. Memori 2.00 GB
3. Harddisk dengan kapasitas 200 GB
4. Monitor 12,1”

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem pengenalan karakter ini adalah sebagai berikut :

1. Sistem Operasi windows XP SP 3
2. Delphi 7

4.2 Implementasi Program

Pada implementasi program ini, terdapat beberapa proses yang dilakukan dalam pembuatan aplikasi pengenalan karakter dengan menggunakan metode *Hidden Markov Model*. Berdasarkan perancangan proses yang telah dijelaskan pada bab 3, maka pada subbab ini akan dijelaskan implementasi dari proses-proses tersebut.

4.2.1 Struktur Data

Struktur data yang digunakan dalam implementasi program pengenalan karakter dengan metode HMM ini ditunjukkan sebagai berikut :

```
1   type
2     koordinat = record
3     x : integer;
4     y : integer;
5   end;
```

***Source code* 4.1** Struktur data *record* koordinat

Pada struktur data di atas, *record* koordinat digunakan untuk menyimpan nilai x dan y dari piksel citra.

```
1  Type
2    ListObj = ^List;
3    List = record
4       x : integer;
5       y : integer;
6       next : ListObj;
7  end;
```

***Source code* 4.2** Struktur data *record* List

Record List digunakan menyimpan piksel-piksel tepi citra. *Record* ini menyimpan nilai x dan y dari tiap piksel citra yang telah di-*thinning*, serta nilai next yang menunjukkan tepi piksel selanjutnya. Nilai-nilai tersebut akan digunakan untuk proses ekstraksi ciri.

```
1  type
2  TMatrik = array[0..8,0..8] of real;
3     TSampel = array of record
4               huruf: string;
5               Matrik : TMatrik;
6               NilaiFB : string;
7  end;
```

***Source code* 4.3** Struktur data pelatihan

Struktur data di atas merupakan struktur data untuk proses pelatihan citra karakter. TMatrik merupakan array dua dimensi yang digunakan untuk menyimpan matrik citra karakter yang telah dilatihkan. *Record* TSampel menyimpan data dari citra karakter yang terdiri dari huruf, matrik, dan nilai *likelihood* yang kemudian disimpan untuk proses pengenalan.

4.2.2 Load Citra

Pemrosesan awal yang dilakukan adalah load citra. Proses ini digunakan untuk membuka citra dari suatu file. Prosedur untuk membuka citra ditunjukkan oleh *source code* 4.4

```
1  procedure TForm1.Button1Click(Sender: TObject);
2   begin
3     if OpenPictureDialog1.Execute then
4  Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
5  end;
```

***Source code* 4.4** *Source code Load citra*

4.2.3 Binerisasi

Proses binerisasi digunakan untuk mendapatkan nilai biner dari suatu citra. Dalam proses binerisasi ini meliputi dua proses, yaitu *grayscaling* dan *thresholding*. Proses *grayscaling* mengubah nilai piksel dari warna (RGB) menjadi citra *grayscale* (keabuan). Sedangkan proses *thresholding* akan mengkonversi citra *grayscale* menjadi citra biner. Pada citra biner, objek dibuat berwarna gelap (hitam) dan latar belakang berwarna terang (putih). Prosedur *binerisasi* ditunjukkan oleh *source code* 4.5

```
1   procedure TForm1.Binerisasi(img:TImage);
2   var
3   i,j:integer;
4   Warna:TColor;
5   Gray,R,G,B:byte;
6   begin
7     for i:=0 to img.Picture.Graphic.Width-1 do
8     begin
9       for j:=0 to img.Picture.Graphic.Height-1 do
10      begin
11        Warna := img.Canvas.Pixels[i,j];
12        R := GetRValue(Warna);
13        G := GetGValue(Warna);
14        B := GetBValue(Warna);
15        Gray := Round(0.299*R+0.587*G+0.114*B);
16        if (Gray<128) then
17           Gray := 0
18        else
19           Gray := 255;
20        img.Canvas.Pixels[i,j] := RGB (Gray,Gray,Gray);
21      end;
22    end
23  end;
```

***Source code* 4.5** *Source code* Prosedur Binerisasi

Pada *Source code* 4.5 di atas, proses *Grayscaling* ditunjukkan pada baris 15. Proses ini mengambil nilai R,G,B dari gambar dengan persentase yang telah ditentukan, yaitu 29,9% dari warna merah (*Red*), 58,7% dari warna hijau (*Green*), dan 11,4% dari warna biru (*Blue*). Proses *Thresholding* yang digunakan untuk mengkonversi dari citra *greyscale* ke citra biner dapat dilihat pada baris 16-20. Jika nilai *grayscale* kurang dari 128, maka piksel berwarna putih, dan jika nilai *grayscale* lebih dari 128, maka piksel berwarna hitam.

4.2.4 Thinning

Thinning (penipisan pola) digunakan untuk mengambil kerangka dari suatu objek. Pada proses *thinning*, terlebih dahulu dilakukan proses inisialisasi nilai piksel pada citra. Bila piksel merupakan bagian dari obyek, maka piksel bernilai 1. Sebaliknya, apabila piksel merupakan bagian dari background, maka piksel bernilai 0. Pada iterasi pertama, jika kondisi 2<B(P1)<=6, A(P1)=1, P2*P4*P6=0, dan P4*P6*P8=0 terpenuhi semua, maka piksel akan ditandai kemudian dilakukan penghapusan. Selanjutnya, dilakukan iterasi kedua pada data hasil dari iterasi pertama. Pada iterasi kedua ini, jika memenuhi kondisi 2<B(P1)<=6, A(P1)=1, P2*P4*P8=0, dan P2*P6*P8=0, maka piksel ditandai kemudian dihapus. Prosedur dari proses *thinning* ditunjukkan pada *source code* 4.6

```
1   procedure TForm1.Thinning(img:TImage);
2   var i,j : integer;
3       sum,x,N,Sx : integer;
4   begin
5    for i:=0 to Img.Picture.Width-1 do
6     begin
7      for j:=0 to img.Picture.Height-1 do
8       begin
9       if img.Canvas.Pixels[i,j] = 255 then
10      Temp1[i,j] := 0;
11      if img.Canvas.Pixels[i,j] =0 then
12      Temp1[i,j] := 1;
13      end;
14    end;
15
16  for i:=0 to Img.Picture.Width-1 do
17   for j:=0 to Img.Picture.Height-1 do
18    Kondisi1[i,j] := 1;
19
20  for i:=0 to Img.Picture.Width-2 do
21   begin
22    for j:=0 to Img.Picture.Height-2 do
23     begin
24      if Temp1[i,j]=1 then
25       begin
26       p[1] := Temp1[i,j];
27       p[2] := Temp1[i,j+1];
28       p[3] := Temp1[i-1,j+1];
29       p[4] := Temp1[i-1,j];
30       p[5] := Temp1[i-1,j-1];
31       p[6] := Temp1[i,j-1];
32       p[7] := Temp1[i+1,j-1];
33       p[8] := Temp1[i+1,j];
34       p[9] := Temp1[i+1,j+1];
35       p[10] := p[2];
```

```
36
37      sum:=0;
38      for x:=2 to 9 do
39      if p[x]=0 then
40      inc(Sum);
41
42      if sum >= 1 then
43      begin
44      N:=0;
45      for x:=2 to 9 do
46      N := N + p[x];
47
48      Sx:=0;
49      for x:=2 to 9 do
50      if (p[x]=0) and (p[x+1]=1) then
51      inc(Sx);
52
53       if    (N>=2)    and    (N<=6)    and    (Sx=1)    and
         (p[2]*p[4]*p[6]=0) and (p[4]*p[6]*p[8]=0) then
54       Kondisi1[i,j]:=0;
55      end;
56     end;
57    end;
58   end;
59  delete(image1);
60
61  {iterasi kedua}
62
63       if    (N>=2)    and    (N<=6)    and    (Sx=1)    and
64       (p[2]*p[4]*p[8]=0) and (p[2]*p[6]*p[8]=0) then
65       Kondisi1[i,j]:=0;
66      end;
       end;
67    end;
68   end;
69  delete(image1);
70  end;
```

***Source code* 4.6** *Source code* prosedur *Thinning*

Proses *thinning* yang digunakan adalah algoritma Zhang Suen. Algoritma Zhang Suen dibagi menjadi dua sub-iterasi. Pada *source code* 4.6, iterasi pertama ditunjukkan pada baris 53. Jika kondisi pada baris tersebut terpenuhi, maka piksel akan dihapus. Iterasi kedua ditunjukkan pada baris 66.

4.2.5 Feature Extraction

Setiap karakter memiliki ciri-ciri khusus yang membedakan antara satu karakter dengan karakter yang lainnya. Proses *feature extraction* digunakan untuk mengetahui ciri-ciri khusus yang ada pada karakter-karakter tersebut. Sebelumnya, karakter yang telah di-*thinning* akan dicari tepi pikselnya. Proses pencarian piksel tepi karakter ditunjukkan oleh *source code* 4.7

```
1   function Tepi(var x, y: integer; col: TColor; image:
2   TImage) : koord;
3   var
4     w1, w2 : TColor;
5     a,b : integer;
6     cekLL : boolean;
7   begin
8     w1 := image.Canvas.Pixels[x+1,y-1];
9     w2 := image.Canvas.Pixels[x+1,y];
10    a := x+1;
11    b := y;
12    cekLL := inEdgeList(a,b);
13    if ((w1<>col) and (w2=col) and (cekLL=false)) then
14    begin
15      Tepi.x := x+1;
16      Tepi.y := y;
17    End
18
19    if ((w1<>col) and (w2=col) and (cekLL=false)) then
20    begin
21      Tepi.x := x+1;
22      Tepi.y := y-1;
23    end
24    else
25    begin
26       Tepi.x := 0;
27       Tepi.y := 0;
28    end;
29
30  end;
```

***Source code* 4.7** *Source code* fungsi deteksi tepi karakter

Setelah dilakukan proses deteksi tepi dan ditemukan piksel penyusun karakter, maka koordinat-koordinat piksel tersebut kemudian disimpan. Sesuai dengan metode *Direction Feature* yang digunakan, maka koordinat-koordinat penyusun karakter tersebut diberi kode sesuai dengan arah koordinat piksel. Prosedur untuk proses *feature extraction* ditunjukkan oleh *source code* 4.8

```
1   procedure TForm1.EkstraksiCiri(img:TImage);
2   var
3    imWidth, imHeight : integer;
4    curClr : TColor;
5    bg : TColor;
6    x, y, x0, y0 : integer;
7    i : integer;
8    pos : koordinat;
9   begin
10
11  while (y<imHeight) and (curClr=bg) do
12   begin
13    while (x<imWidth) and (curClr=bg) do
14    begin
15     curClr := img.Canvas.Pixels[x,y];
16     if (curClr=bg) then
17      x:=x+1;
18    end;
19    if (curClr=bg) then
20    begin
21     y := y+1;
22     x := 0;
23    end;
24   end;
25   x0 := x;
26   y0 := y;
27   add2LL(x0,y0);
28
29   i:=1;
30   pos := Tepi(x0,y0,img.Canvas.Pixels[0,0],img);
31   x := pos.x;
32   y := pos.y;
33   i:=i+1;
34   add2LL(x,y);
35
36   while ((x<>0) or (y<>0)) do
37   begin
38    pos := Tepi(pos.x,pos.y,img.Canvas.Pixels[0,0],img);
39    x:=pos.x;
40    y:=pos.y;
41
42    if not((x=0) and (y=0)) then
43     add2LL(x,y);
44
45    if i=2 then
46    i := i+1;
47   nextEdge(x,y,img.Canvas.Pixels[0,0],img,ListBox1);
48   end;
49
50  end;
51     end;
```

***Source code* 4.8** *Source code* prosedur *Feature Extraction*

Pada *Source code* 4.8, proses Tepi pada baris 45 merupakan proses untuk memperoleh batas-batas pinggir dari karakter. Sedangkan proses nextEdge pada baris 54 merupakan proses untuk mendapatkan ciri dari karakter.

4.3 Implementasi *Hidden Markov Model*

4.3.1 Pelatihan

Pada proses pelatihan ini, terlebih dahulu ditentukan nilai awal parameter HMM (A,B,π) seperti yang telah dijelaskan pada bab 3. Selanjutnya, digunakan algoritma *Baum-Welch* untuk mendapatkan nilai baru parameter HMM (NewA, NewB, Newπ). Setelah didapatkan nilai parameter HMM untuk karakter yang dilatihkan, selanjutnya dilakukan perhitungan nilai *likelihood* karakter dengan algoritma *forward backward*. Prosedur untuk proses pelatihan ditunjukkan oleh *source code* 4.9

```
1    procedure TrainHmm();
2    var i,k,l : integer;
3       NewA: IFArr2D;
4       NewB: IFArr2D;
5       NewP0: IFArr1D;
6    begin
7    N := 8;
8    M := 8;
9    K := ListBox1.Count;
10
11   P0 := TFArr1D.Create(N);
12
13   A := TFArr2D.Create(N,N);
14
15   B := TFArr2D.Create(N,M);
16
17   Idxs := TIArr1D.Create(K);
18   for i:=0 to ListBox1.Count-1 do
19   Idxs[i+1] := StrToInt(ListBox1.Items[i]);
20
21     hmm := Thmm.Create(A, B, P0);
22      hmm.BaumWelchIteration(Idxs, NewA, NewB, NewP0);
23      ShowMatrix(NewA, Memo1);
24      ShowMatrix(NewB, Memo2);
25      ShowPi(NewP0, Memo3);
26      HMM.SetTransitions(NewA);
27      HMM.SetEmissions(NewB);
28      HMM.SetProbabilities(NewP0);
29
30      A := NewA;
31      B := NewB;
32      P0 := NewP0;
33
```

```
34  Edit3.Text := FloatToStr(hmm.ForwardProbability(Idxs));
35  hmm.Free;
36  end;
```

***Source code* 4.9** *Source code* Prosedur Pelatihan HMM

Pada *source code* 4.9 di atas, A adalah matriks peluang transisi dan B adalah matriks peluang observasi. Baris 14 pada *source code* di atas merupakan implementasi dari model matriks peluang transisi yang dapat dilihat pada gambar 3.15. Sedangkan implementasi model awal matriks peluang observasi pada gambar 3.18 ditunjukkan pada baris 17.

Algoritma *BaumWelch*

Pada prosedur TrainHMM, telah ditentukan model awal HMM dari karakter yang dilatihkan. Algoritma *BaumWelch* digunakan untuk mengestimasi model awal parameter HMM tersebut sehingga didapatkan model HMM yang baru, yaitu NewA, NewB, dan NewP0. Prosedur algoritma *BaumWelch* ditunjukkan oleh *source code* 4.10

```
1
2   for i := alpha.Lo2 to alpha.Hi2 do
3       sum := sum + alpha[t,i]*beta[t,i];
4
5   for t := alpha.Lo1 to alpha.Hi1 do
6     begin
7       for i := alpha.Lo2 to alpha.Hi2 do
8         result[t,i] := alpha[t,i]*beta[t,i]/Norm;
9
10    t := gamma.Lo1;
11    for i := gamma.Lo2 to gamma.Hi2 do
12      result[i] := gamma[t,i];
13
14  xi :=
15  alpha[t,i]*A[i,j]*B[j,ObsIdxs[t+1]]*beta[t+1,j]/Norm;
16          num := num + xi;
17
18  for t := ObsIdxs.Lo1 to ObsIdxs.Hi1 do
19        begin
20          k := ObsIdxs[t];
21          result[i,k] := result[i,k] + gamma[t,i]/den;
22        end;
```

***Source code* 4.10** Potongan fungsi *BaumWelch*

Source code 4.10 di atas merupakan implementasi dari algoritma *BaumWelch*. Baris 10 pada *source code* 4.14 merupakan implementasi dari persamaan (2.11), sedangkan baris 19 merupakan implementasi dari

persamaan (2.12). Persamaan tersebut digunakan untuk menghitung probabilitas state qi pada waktu t.

```
1   procedure Thmm.BaumWelchIteration(const ObsIdxs: IIArr1D;
2   var NewA, NewB: IFArr2D; var NewP0: IFArr1D);
3
4   Begin
5     alpha := GetAlpha(ObsIdxs, A, B, P0);
6     beta  := GetBeta(ObsIdxs, A, B);
7     Norm  := GetNorm(alpha, beta);
8     gamma := GetGamma(alpha, beta, Norm);
9     // estimasi
10    NewP0 := GetP0Estimate(gamma);
11    NewA  := GetAEstimate(ObsIdxs, A, B, alpha, beta, gamma,
12  Norm);
13    NewB  := GetBEstimate(ObsIdxs, gamma);
14  end;
15
16
```

***Source code* 4.11** *Source code* Prosedur *BaumWelch*

Source code 4.11 berisi fungsi-fungsi dari prosedur *BaumWelch* yang digunakan untuk mengestimasi model awal dari parameer HMM yang telah dibuat. Baris 6 sampai 7 merupakan fungsi untuk mendapatkan nilai alpha, beta, dan gamma yang akan digunakan untuk proses estimasi parameter HMM. NewP0 pada baris 11 merupakan nilai *Initial probabilities* baru dari HMM, sedangkan NewA dan NewB masing-masing merupakan *Transition probabilities* dan *Emission probabilities* baru dari parameter HMM yang telah diestimasi.

Algoritma *Forward Backward*

Algoritma *Forward Backward* digunakan untuk menghitung nilai *likelihood* dari karakter. Prosedur algoritma *forwardBackward* ditunjukkan oleh *source code* 4.12

```
1   function Thmm.GetAlpha(const ObsIdxs: IIArr1D; const A:
2   IFArr2D; const B: IFArr2D; const P0: IFArr1D): IFArr2D;
3
4   begin
5
6     // Initialization
7     t := ObsIdxs.Lo1;
8     for i := A.Lo1 to A.Hi1 do
9       result[t,i] := P0[i]*B[i,ObsIdxs[t]];
10
11    // Recursion
12    for t := ObsIdxs.Lo1 to ObsIdxs.Hi1-1 do
```

```
13    begin
14      for j := A.Lo1 to A.Hi1 do
15      begin
16        sum := 0.0;
17        for i := A.Lo1 to A.Hi1 do
18          sum := sum + result[t,i]*A[i,j];
19
20        result[t+1,j] := sum*B[j,ObsIdxs[t+1]];
21      end;
22    end;
23  end;
24
```

***Source code* 4.12** *Source code* fungsi *GetAlpha*

Source code 4.12 di atas merupakan proses inisialisasi dan rekursi dari probabilitas *forward*. Proses inisialisasi ditunjukkan pada baris 10 yang merupakan implementasi dari persamaan (2.5). Sedangkan hasil dari proses rekursi ditunjukkan pada baris 21 yang merupakan implementasi dari persamaan (2.6).

```
1   function Thmm.ForwardProbability(const ObsIdxs: IIArr1D):
2   TFloat;
3   var
4     t, i: TInt;
5     sum: TFloat;
6     alpha: IFArr2D;
7   begin
8     // Initialization, Recursion
9     alpha := GetAlpha(ObsIdxs, A, B, P0);
10
11    // Termination
12    t := ObsIdxs.Hi1;
13    sum := 0.0;
14    for i := A.Lo1 to A.Hi1 do
15      sum := sum + alpha[t,i];
16
17    result := sum;
18  end;
```

***Source code* 4.13** *Source code* Prosedur *Forward Backward*

Langkah selanjutnya dari algoritma *forward* ini adalah menghitung nilai probabilitas dari rangkaian observasi, yang didapatkan melalui persamaan (2.7). Implementasi dari persamaan (2.7) ini dapat dilihat pada *source code* 4.13 pada baris 15.

```
1   function Thmm.GetBeta(const ObsIdxs: IIArr1D; const A:
2   IFArr2D; const B: IFArr2D): IFArr2D;
3
4   begin
```

```
5
6       // Initialization
7       t := ObsIdxs.Hi1;
8       for i := A.Lo1 to A.Hi1 do
9         result[t,i] := 1.0;
10
11      // Recursion
12      for t := ObsIdxs.Hi1-1 downto ObsIdxs.Lo1 do
13      begin
14        for i := A.Lo1 to A.Hi1 do
15        begin
16          sum := 0.0;
17          for j := A.Lo1 to A.Hi1 do
18            sum := sum +
19                A[i,j]*B[j,ObsIdxs[t+1]]*result[t+1,j];
20
21        result[t,i] := sum;
22      end;
23    end;
24  end;
```

***Source code* 4.14** *Source code* Fungsi *GetBeta*

Source code 4.14 berisi fungsi *GetBeta* yang digunakan untuk menghitung probabilitas *backward.* Fungsi ini merupakan implementasi dari persamaan (2.8) dan persamaan (2.9). Implementasi persamaan (2.8) yang merupakan proses inisialisasi dari probabilitas *backward* dapat dilihat pada source code 4.14 baris 10, sedangkan Implementasi persamaan (2.9) yang merupakan proses rekursi dari probabilitas *backward* dapat dilihat pada source code 4.14 baris 20.

```
1   function Thmm.BackwardProbability(const ObsIdxs: IIArr1D):
2   TFloat;
3   var
4     t, i: TInt;
5     sum: TFloat;
6     beta: IFArr2D;
7   begin
8     // Initialization, Recursion
9     beta := GetBeta(ObsIdxs, A, B);
10    // Termination
11    t := ObsIdxs.Lo1;
12    sum := 0.0;
13    for i := A.Lo1 to A.Hi1 do
14      sum := sum + P0[i]*B[i,ObsIdxs[t]]*beta[t,i];
15    result := sum;
16  end;
```

***Source code* 4.15** *Source code* Prosedur *Backward*

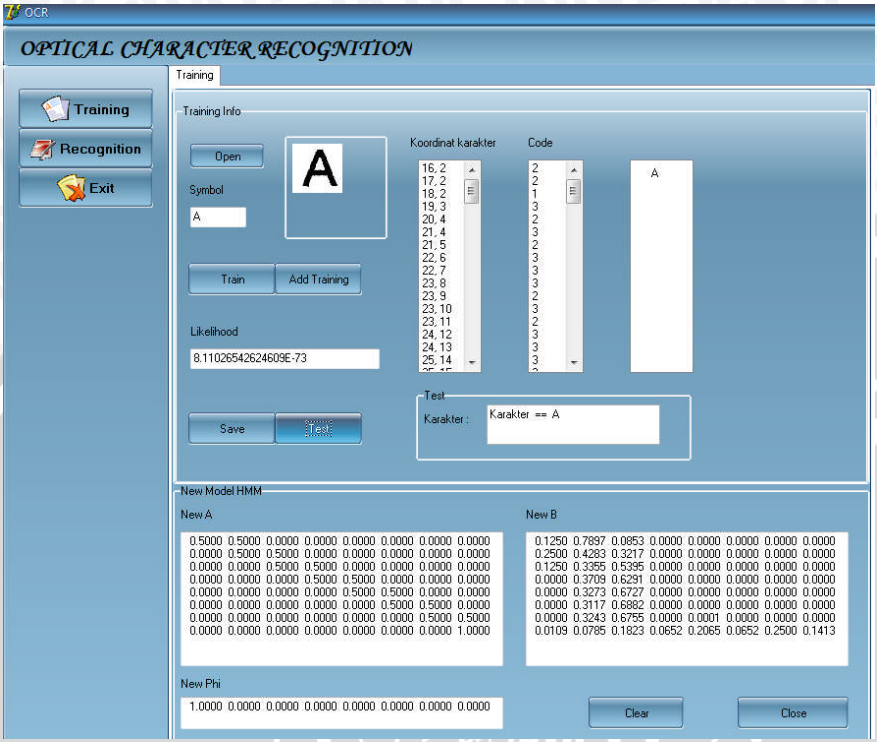
S*ource code* 4.15 berisi prosedur yang digunakan untuk menghitung nilai probabilitas dari rangkaian observasi, yang didapatkan melalui persamaan (2.10). Pada implementasi programnya, prosedur *forward* dan *backward* ini menghasilkan nilai *likelihood* yang sama.

4.4 Implementasi Antarmuka

Antarmuka pada sistem ini terdiri dari dua bagian, yaitu antarmuka *Training* dan antarmuka proses pengenalan. Antarmuka training menunjukkan proses pembentukan data pelatihan karakter, sedangkan antarmuka proses pengenalan menunjukkan proses pengenalan karaker yang diujikan.

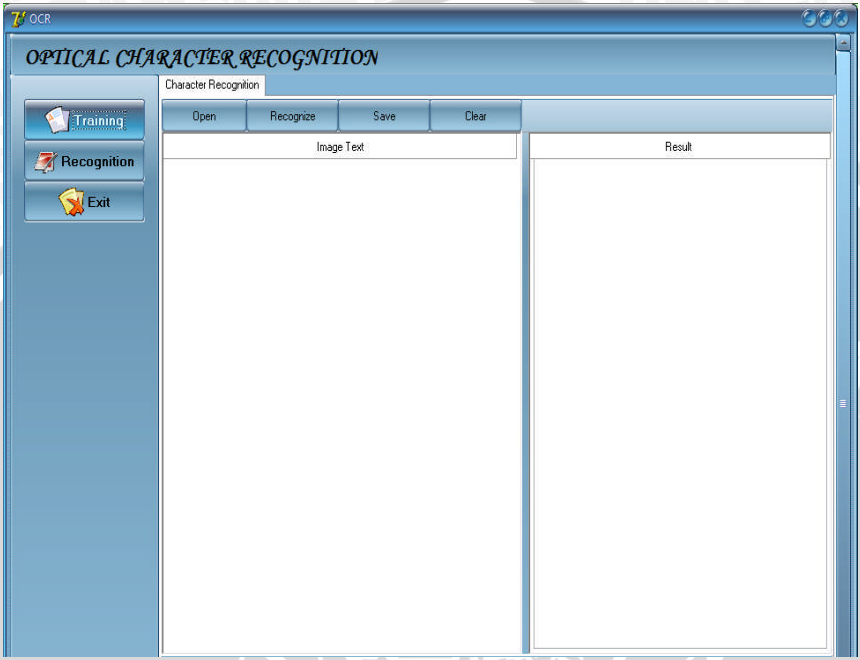
Pada antarmuka *Training*, tombol Train digunakan untuk melatih citra karakter, yang menampilkan beberapa informasi, antara lain : data koordinat karakter yang merupakan koordinat tepi piksel dari karakter yang dilatihkan. Koordinat-koordinat karakter tersebut selanjutnya akan dikodekan sesuai dengan arah koordinat piksel. Kode ini merupakan ciri dari karakter yang akan digunakan untuk menentukan matrik peluang observasi yang baru (NewB) dan juga untuk menghitung nilai *likelihood* dari karakter yang dilatihkan. Nilai newB dan nilai *likelihood* ini selanjutnya akan disimpan dalam sebuah file teks yang digunakan untuk proses pengenalan. Antarmuka *Training* dapat dilihat pada gambar 4.1.

Pada antarmuka *Character Recognition* atau proses pengenalan, terdapat dua bagian, yaitu *Image Text* dan *Result*. *Image Text* digunakan untuk menampilkan karakter yang akan dikenali, dan *Result* digunakan untuk menampilkan hasil pengenalan. Antarmuka proses pengenalan dapat dilihat pada gambar 4.2.



Gambar 4.1 Antarmuka *Training*

Antarmuka Proses Pengenalan dapat dilihat pada gambar 4.2.



Gambar 4.2 Antarmuka Pengenalan

4.5 Implementasi Uji Coba dan Analisa Hasil

Aplikasi perangkat lunak yang telah dibuat akan dilakukan beberapa pengujian dan analisa.

4.5.1 Hubungan keakuratan pembacaan teks dengan jenis huruf

Pengujian ini dilakukan untuk mengetahui keberhasilan pembacaan dari tiap huruf data uji dengan variasi jenis huruf yang berbeda. Ukuran huruf yang digunakan adalah 12. Data uji yang digunakan dapat dilihat pada lampiran 1.

4.5.2 Pengujian Pertama

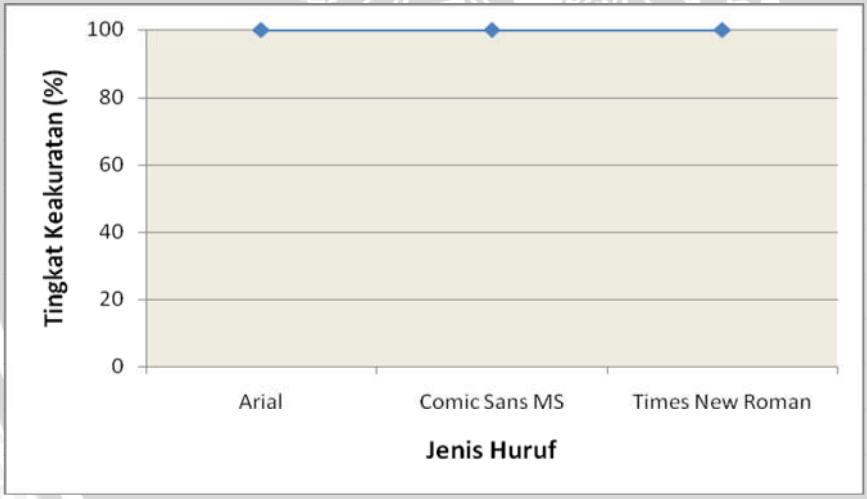
Pengujian pertama dilakukan terhadap karakter dengan jenis huruf yang sama dengan jenis huruf pelatihan, yaitu Arial, Comic Sans MS, dan Times New Roman. Hasil pengujian ini dapat dilihat pada tabel 4.1

Tabel 4.1 Pengujian karakter jenis huruf pelatihan

Jenis Huruf	Jumlah karakter	Jumlah benar	Jumlah salah	Keberhasilan (%)
Arial	62	62	0	100
Comic Sans MS	62	62	0	100
Times New Roman	62	62	0	100
Rata-rata				100

Dari tabel 4.1, dapat dilihat bahwa tingkat keberhasilan pengujian rata-rata adalah 100%. Karakter yang diujikan dari tiap-tiap jenis huruf adalah karakter A..Z, karakter a..z, dan karakter 0..9. Jumlah karakter yang digunakan pada pengujian ini sebanyak 186 karakter. Data pengujian dan hasil pengujian dari karakter jenis huruf pelatihan ini dapat dilihat pada lampiran 2.

Berdasarkan tabel hasil pengukuran tingkat keakuratan pada pengujian pertama, maka tabel 4.1 dapat disajikan dalam bentuk grafik seperti pada gambar 4.3



Gambar 4.3 Grafik Pengujian Pertama

Grafik pada gambar 4.3 menunjukkan persentase jumlah huruf yang dikenali serta tingkat keakuratannya. Pada pengujian pertama ini, ketiga jenis huruf yang diujikan memiliki tingkat keakuratan yang tinggi, yaitu 100%. Hal ini karena jenis huruf dari karakter yang diujikan sama dengan jenis huruf dari karakter yang telah dilatihkan.

Nilai *likelihood* yang dihasilkan karakter data uji memiliki tingkat kemiripan yang tinggi dengan nilai data latih sehingga proses pengenalan pun dapat dilakukan dengan akurat.

4.5.3 Pengujian Kedua

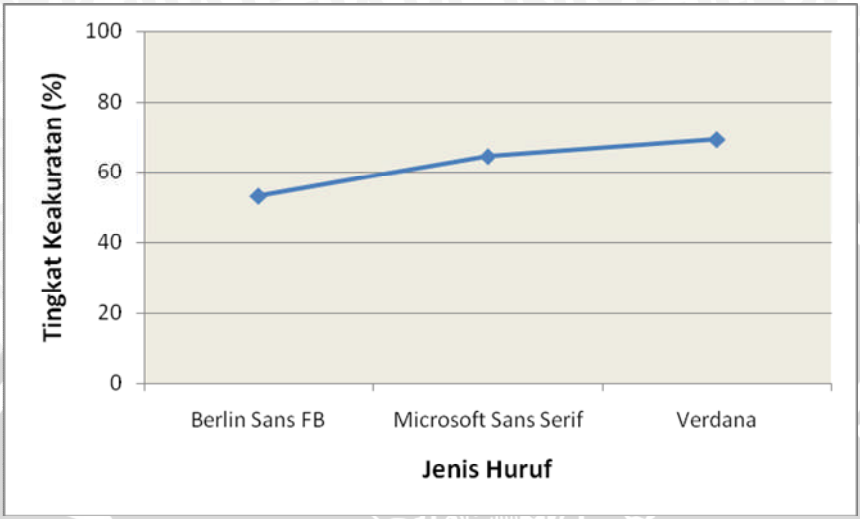
Pengujian kedua dilakukan terhadap karakter dengan jenis huruf non-pelatihan, yaitu Berlin Sans FB, Microsoft Sans Serif, dan Verdana. Hasil pengujian ini dapat dilihat pada tabel 4.2

Tabel 4.2 Pengujian karakter bukan jenis huruf pelatihan

Jenis Huruf	Jumlah karakter	Jumlah benar	Jumlah salah	Keberhasilan (%)
Berlin Sans FB	62	33	29	53.2
Microsoft Sans Serif	62	40	22	64.5
Verdana	62	43	19	69.3
Rata-rata				62.3

Dari Tabel 4.2, dapat dilihat bahwa tingkat keberhasilan pengujian rata-rata adalah 62.3%. Seperti pada pengujian pertama, karakter dari tiap-tiap jenis huruf yang digunakan pada pengujian kedua ini adalah karakter A..Z, karakter a..z, dan karakter 0..9. Jumlah karakter yang digunakan pada pengujian ini sebanyak 186 karakter. Data pengujian dan hasil pengujian dari karakter jenis huruf pelatihan ini dapat dilihat pada lampiran 2.

Berdasarkan tabel hasil pengukuran tingkat keakuratan pada pengujian kedua, maka tabel 4.2 dapat disajikan dalam bentuk grafik seperti pada gambar 4.4



Gambar 4.4 Grafik Pengujian Kedua

Grafik pada gambar 4.4 menunjukkan persentase jumlah huruf yang dikenali serta tingkat keakuratannya. Pada pengujian kedua ini, dapat dilihat bahwa tingkat keberhasilannya rata-rata 62.36%. Tingkat keakuratan tertinggi adalah pengujian pada jenis huruf Verdana, yaitu dengan keberhasilan sebesar 69.35%.

Pengujian kedua yang dilakukan pada karakter jenis huruf non-pelatihan ini memiliki tingkat keakuratan yang kurang bagus, yaitu dengan tingkat keberhasilan pengujian rata-rata sebesar 62.36%. Hal ini disebabkan karena model karakter yang diujikan berbeda dengan model karakter yang telah dilatihkan. Sedikit saja perbedaan model karakter, akan menyebabkan perbedaan pada proses ekstraksi ciri yang juga akan berpengaruh pada nilai *likelihood* yang dihasilkan. Jika model karakter yang diujikan mirip dengan salah satu model karakter yang telah dilatihkan, maka rata-rata hasil pengenalan tersebut akan tinggi, begitu juga sebaliknya.

4.5.4 Pengujian Ketiga

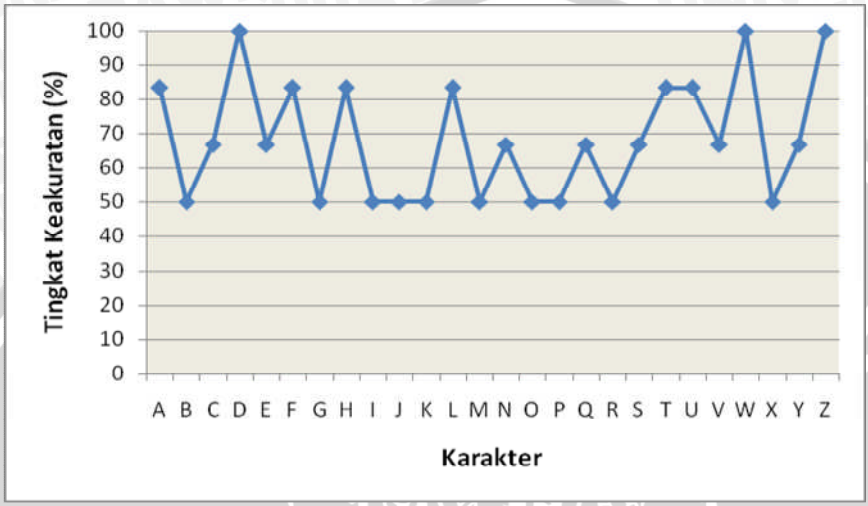
Pengujian ketiga ini dilakukan terhadap karakter dengan jenis huruf yang telah digunakan pada pengujian pertama dan kedua. Pengujian ini bertujuan untuk melihat tingkat keakuratan dari masing-masing karakter dari berbagai jenis huruf yang telah diujikan. Pada pengujian ini, jika karakter benar dikenali akan diberikan nilai 1, sedangkan jika karakter

salah dikenali akan diberikan nilai 0. Hasil pengujian ini ditunjukkan pada tabel 4.3

Tabel 4.3 Pengujian karakter huruf kapital

Karakter	Jenis Huruf						Jumlah benar	Persentase (%)
	Arial	Comic Sans MS	Times New Roman	Berlin Sans FB	Microsoft Sans Serif	Verdana		
A	1	1	1	1	0	1	5	83.3
B	1	1	1	0	0	0	3	50
C	1	1	1	1	0	0	4	66.7
D	1	1	1	1	1	1	6	100
E	1	1	1	0	0	1	4	66.7
F	1	1	1	0	1	1	5	83.3
G	1	1	1	0	0	0	3	50
H	1	1	1	1	0	1	5	83.3
I	1	1	1	0	0	0	3	50
J	1	1	1	0	0	0	3	50
K	1	1	1	0	0	0	3	50
L	1	1	1	0	1	1	5	83.3
M	1	1	1	0	0	0	3	50
N	1	1	1	0	0	1	4	66.7
O	1	1	1	0	0	0	3	50
P	1	1	1	0	0	0	3	50
Q	1	1	1	0	1	1	4	66.7
R	1	1	1	0	0	0	3	50
S	1	1	1	0	0	1	4	66.7
T	1	1	1	1	0	1	5	83.3
U	1	1	1	0	1	1	5	83.3
V	1	1	1	0	1	0	4	66.7
W	1	1	1	1	1	1	6	100
X	1	1	1	0	0	0	3	50
Y	1	1	1	0	1	0	4	66.7
Z	1	1	1	1	1	1	6	100

Tabel 4.3 dapat disajikan dalam bentuk grafik seperti pada gambar 4.5



Gambar 4.5 Grafik Pengujian karakter huruf kapital

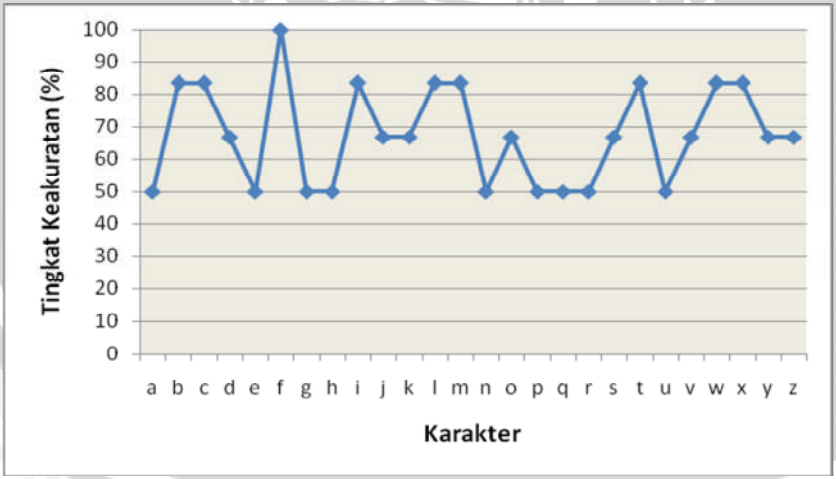
Berdasarkan tabel 4.3, dapat diketahui bahwa karakter huruf yang memiliki tingkat keakuratan terbesar adalah karakter D, W, dan Z, dengan persentase keberhasilan sebesar 100%. Hal ini menunjukkan bahwa pada pengujian yang telah dilakukan, tiga karakter tersebut dapat dikenali secara benar pada semua jenis huruf. Persentase terkecil dalam pengujian ini adalah sebesar 50%.

Tabel 4.4 Pengujian karakter huruf kecil

Karakter	Jenis Huruf						Jumlah benar	Persentase
	Arial	Comic Sans MS	Times New Roman	Berlin Sans FB	Microsoft Sans Serif	Verdana		
a	1	1	1	0	0	0	3	50
b	1	1	1	1	1	0	5	83.3
c	1	1	1	0	1	1	5	83.3
d	1	1	1	0	0	1	4	66.7
e	1	1	1	0	0	0	3	50
f	1	1	1	1	1	1	6	100
g	1	1	1	0	0	0	3	50

h	1	1	1	0	0	0	3	50
i	1	1	1	0	1	1	5	83.3
j	1	1	1	1	0	0	4	66.7
k	1	1	1	0	0	1	4	66.7
l	1	1	1	0	1	1	5	83.3
m	1	1	1	1	0	1	5	83.3
n	1	1	1	0	0	0	3	50
o	1	1	1	1	0	0	4	66.7
p	1	1	1	0	0	0	3	50
q	1	1	1	0	0	0	3	50
r	1	1	1	0	0	0	3	50
s	1	1	1	0	0	1	4	66.7
t	1	1	1	1	0	1	5	83.3
u	1	1	1	0	0	0	3	50
v	1	1	1	0	0	1	4	66.7
w	1	1	1	1	0	1	5	83.3
x	1	1	1	1	1	0	5	83.3
y	1	1	1	0	0	1	4	66.7
z	1	1	1	0	0	1	4	66.7

Tabel 4.4 dapat disajikan dalam bentuk grafik seperti pada gambar 4.6



Gambar 4.6 Grafik Pengujian karakter huruf kecil

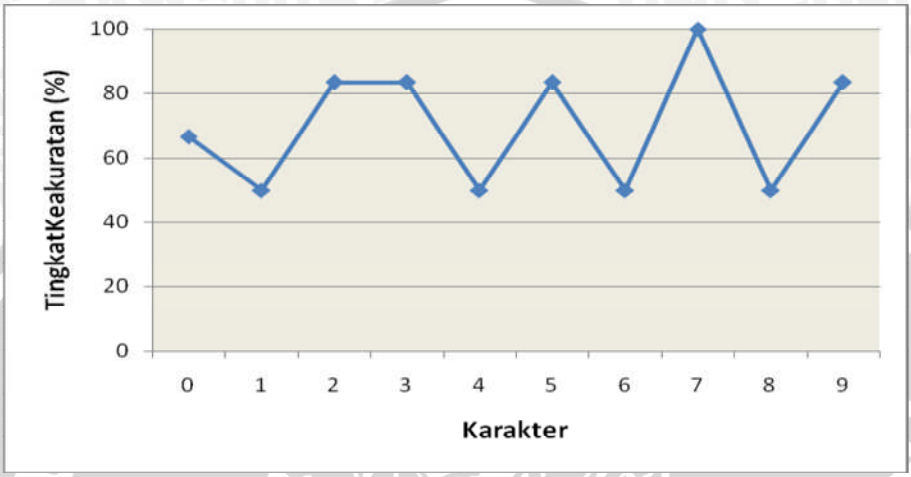
Berdasarkan tabel 4.4, dapat diketahui bahwa hanya ada satu karakter huruf yang memiliki tingkat keakuratan terbesar, yaitu karakter

f, dengan persentase keberhasilan sebesar 100%. Hal ini menunjukkan bahwa pada pengujian yang telah dilakukan, karakter f tersebut dapat dikenali secara benar pada semua jenis huruf. Persentase terkecil dalam pengujian ini adalah sebesar 50%.

Tabel 4.5 Pengujian karakter angka

Karakter	Jenis Huruf						Jumlah benar	Persentase
	Arial	Comic Sans MS	Times New Roman	Berlin Sans FB	Microsoft Sans Serif	Verdana		
0	1	1	1	0	0	1	4	66.7
1	1	1	1	0	0	0	3	50
2	1	1	1	0	1	1	5	83.3
3	1	1	1	1	1	0	5	83.3
4	1	1	1	0	0	0	3	50
5	1	1	1	1	1	0	5	83.3
6	1	1	1	0	0	0	3	50
7	1	1	1	1	1	1	6	100
8	1	1	1	0	0	0	3	50
9	1	1	1	0	1	1	5	83.3

Tabel 4.5 dapat disajikan dalam bentuk grafik seperti pada gambar 4.7



Gambar 4.7 Grafik Pengujian karakter angka

Berdasarkan tabel 4.5, dapat diketahui bahwa karakter angka yang memiliki tingkat keakuratan terbesar adalah angka 7, dengan persentase keberhasilan sebesar 100%. Hal ini menunjukkan bahwa pada pengujian yang telah dilakukan, angka 7 tersebut dapat dikenali secara benar pada semua jenis huruf. Persentase terkecil dalam pengujian ini adalah sebesar 50%.

4.5.5 Pengujian Keempat

Pengujian keempat ini dilakukan dengan 3 jenis data uji yang berbeda. Masing-masing data uji terdiri dari 6 jenis huruf yang akan diuji, yaitu jenis huruf yang telah digunakan pada pengujian pertama dan kedua. Pengujian ini bertujuan untuk melihat keakuratan pengenalan karakter untuk berbagai jenis huruf. Hasil pengujian ini ditunjukkan pada tabel 4.6 dan tabel 4.7

Tabel 4.6 Pengujian dokumen jenis huruf pelatihan

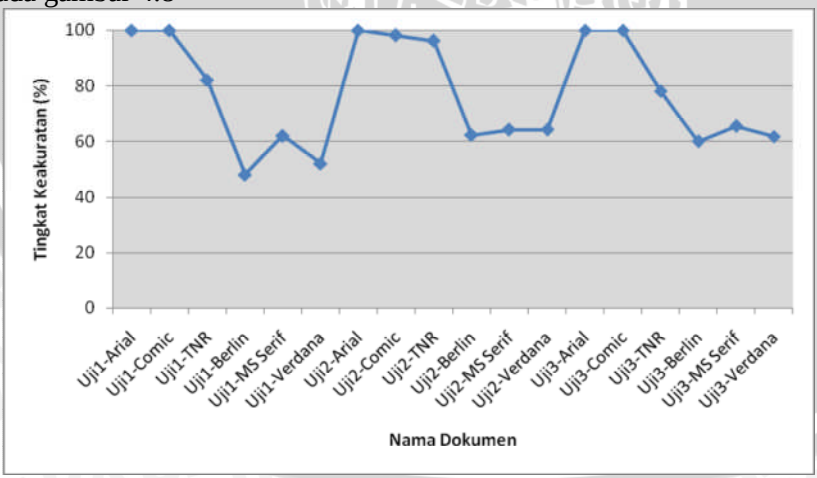
Nama dokumen	Jumlah karakter	Jumlah benar	Jumlah salah	Keberhasilan (%)
Uji1-Arial	50	50	0	100
Uji 1-Comic	50	50	0	100
Uji 1-TNR	50	41	9	82

Uji 2-Arial	53	53	0	100
Uji 2-Comic	53	52	1	98.1
Uji 2-TNR	53	51	2	96.2
Uji 3-Arial	55	55	0	100
Uji 3-Comic	55	55	0	100
Uji 3-TNR	55	43	12	78.1
Rata-rata				94.9

Tabel 4.7 Pengujian dokumen jenis huruf non-pelatihan

Nama dokumen	Jumlah karakter	Jumlah benar	Jumlah salah	Keberhasilan (%)
Uji 1-Berlin	50	24	26	48
Uji 1-MS Serif	50	31	19	62
Uji 1-Verdana	50	26	24	52
Uji 2-Berlin	53	33	20	62.3
Uji 2-MS Serif	53	34	19	64.2
Uji 2-Verdana	53	34	19	64.2
Uji 3-Berlin	55	33	22	60
Uji 3-MS Serif	55	36	19	65.5
Uji 3-Verdana	55	34	21	61.8
Rata-rata				60

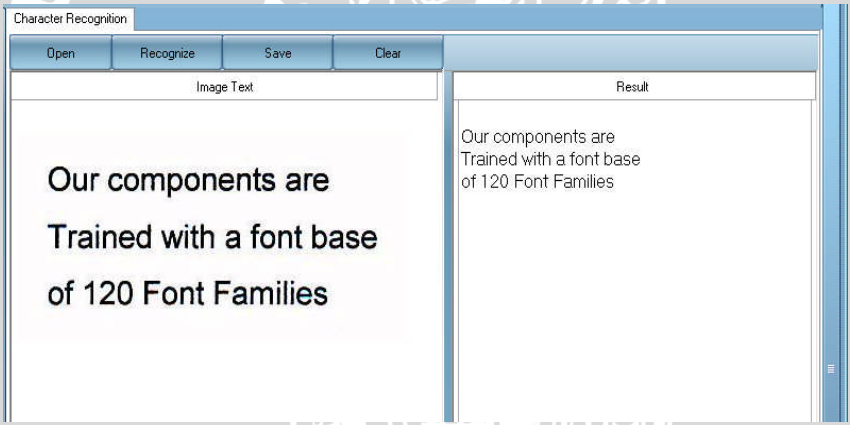
Tabel 4.6 dan Tabel 4.7 dapat disajikan dalam bentuk grafik seperti pada gambar 4.8



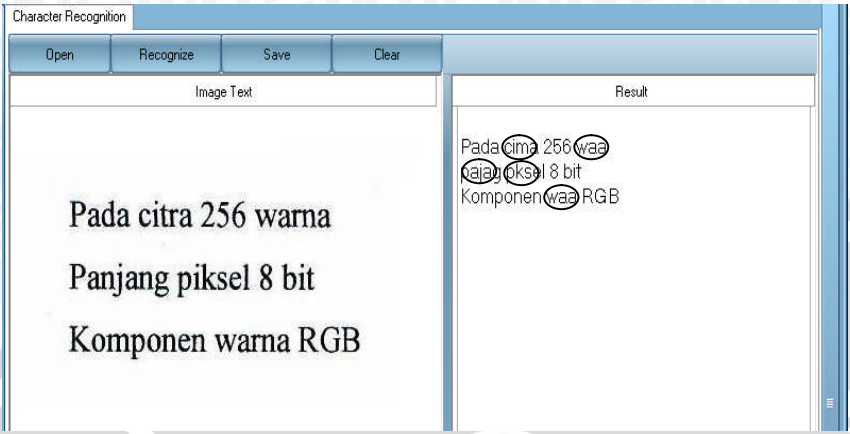
Gambar 4.8 Grafik Pengujian dokumen

Gambar 4.8 merupakan grafik pengujian keempat secara keseluruhan. Berdasarkan grafik tersebut, dapat diketahui bahwa data uji arial, comic, dan times new roman memiliki tingkat keakuratan yang tinggi pada setiap pengujian, sedangkan data uji berlin, MS serif, dan verdana memiliki tingkat keakuratan yang rendah pada setiap pengujian. Pada pengujian keempat ini, tingkat keakuratan tertinggi adalah 100% untuk data uji jenis huruf pelatihan dan 65.5% untuk jenis huruf non-pelatihan. Tingkat keakuratan terendah adalah 78.1% untuk jenis huruf pelatihan yang didapatkan pada pengujian Uji3-TNR dan 48% untuk jenis huruf non-pelatihan yang didapatkan pada pengujian Uji1-Berlin.

Pada pengujian keempat ini, dari ketiga jenis huruf pelatihan, jenis huruf Times New Roman memiliki tingkat keakuratan paling rendah. Hal ini dapat dilihat pada tabel 4.6, pengujian Uji1-TNR hanya memiliki tingkat keakuratan sebesar 82%. Contoh pengujian ditunjukkan pada gambar 4.9, 4.10, dan 4.11



Gambar 4.9 Contoh Pengujian Uji-2 Arial



Contoh kesalahan dalam proses pengenalan

Citra setelah di-*thinning*

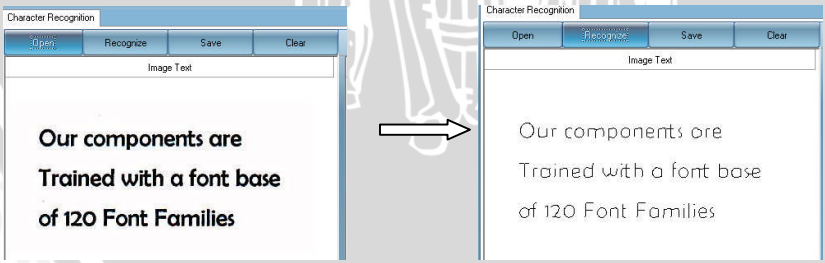
Gambar 4.10 Contoh Pengujian Uji-1 TNR

Pada gambar 4.10 dapat dilihat bahwa kesalahan pengenalan disebabkan karena adanya beberapa huruf yang terhubung, antara lain huruf t dan r pada kata citra, huruf a,r, dan n pada kata warna, serta huruf i dan k pada kata piksel. Terhubungnya beberapa huruf ini akan berpengaruh pada proses segmentasi, karena beberapa karakter yang terhubung tersebut akan dikenali sebagai satu karakter, sehingga mengakibatkan kesalahan pada proses pengenalan. Proses segmentasi dengan metode deteksi tepi tidak efektif untuk mensegmentasi karakter yang terhubung, namun metode ini efektif digunakan untuk mensegmentasi karakter yang bertumpangan, seperti pada contoh gambar 4.11

Gambar 4.11 Contoh citra teks

Pada gambar 4.11, dapat dilihat bahwa karakter f dan o saling bertumpangan, namun dengan segmentasi deteksi tepi, kedua karakter tersebut dapat dipisahkan dengan benar.

Pada pengujian keempat dengan jenis huruf non-pelatihan, dari ketiga jenis huruf yang dilatihkan, jenis huruf Berlin Sans FB memiliki tingkat keakuratan paling rendah. Hal ini dapat dilihat pada tabel 4.7 dimana pengujian Uji1-Berlin hanya memiliki tingkat keakuratan sebesar 48%. Contoh pengujian dokumen Berlin Sans FB ditunjukkan pada gambar 4.12



Citra awal Citra setelah di-*thinning*

Gambar 4.12 Contoh Pengujian dokumen Berlin Sans FB

Pada gambar 4.12 dapat dilihat bahwa dokumen Berlin Sans FB memiliki kerapatan yang tinggi antara satu karakter dengan karakter lainnya dalam satu kata. Setelah dilakukan proses *thinning*, dapat dilihat bahwa ada beberapa huruf yang saling terhubung. Hal ini menyebabkan kesalahan pengambilan huruf dalam proses segmentasi, yang akan menyebabkan kesalahan dalam proses pengenalan.

Dari seluruh pengujian yang telah dilakukan, dapat diketahui bahwa tiap data uji yang diujikan memiliki tingkat keakuratan yang berbeda-beda. Hal ini dapat dipengaruhi oleh banyaknya jumlah huruf yang akan dikenali serta jenis huruf yang digunakan dalam pengujian. Untuk pengujian karakter jenis huruf pelatihan, tingkat keakuratan yang dihasilkan lebih tinggi. Hal ini dapat dilihat pada pengujian pertama dan keempat, dimana jenis huruf Arial, Comic Sans MS, dan Times New Roman memiliki tingkat keakuratan yang tinggi, dengan rata-rata di atas 90%, sedangkan pada pengujian karakter jenis huruf non-pelatihan, tingkat keakuratan yang dihasilkan rendah, yaitu dengan rata-rata 60%. Dari hasil tersebut, dapat diketahui bahwa hasil pengenalan karakter bergantung pada jenis karakter tersebut, apabila model karakter yang akan dikenali mirip dengan salah satu model karakter yang telah dilatihkan, maka hasil pengenalan karakter tersebut akan tinggi.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang diperoleh dari hasil penulisan skripsi ini adalah :

1. Sistem pengenalan karakter yang dibangun terdiri dari dua bagian, yaitu tahap pelatihan dan tahap pengenalan. Pada tahap pelatihan, proses yang dilakukan adalah input data latih, *preprocessing*, *feature extraction*, dan pemodelan karakter. Sedangkan pada tahap pengenalan, proses yang dilakukan adalah input data uji, *preprocessing*, segmentasi, *feature extraction*, dan pengenalan karakter.
2. Proses pengenalan karakter untuk setiap jenis huruf yang telah dilatihkan memberikan hasil pengenalan yang baik, yaitu dengan persentase kebenaran rata-rata sebesar 94.9%, sedangkan proses pengenalan karakter dengan jenis huruf non-pelatihan memberikan hasil yang kurang baik, yaitu dengan persentase kebenaran rata-rata sebesar 60%.

5.2 Saran

Berdasarkan hasil dan kesimpulan yang telah diperoleh, terdapat beberapa saran untuk pengembangan penelitian selanjutnya.

1. Metode segmentasi karakter yang digunakan dapat diperbaiki sehingga dapat dilakukan segmentasi karakter dengan benar.
2. Citra data latih dapat ditambahkan dengan jenis huruf yang lebih bervariasi sehingga sistem dapat melakukan pengenalan karakter dengan lebih baik.

DAFTAR PUSTAKA

Amelia, Sari. 2007. Aplikasi Jaringan Saraf Tiruan Dalam Rekognisi Karakter Numerik berbasis Citra**. ITB Central Library. Bandung.**

Anigbogu, J.C. dan A. Belaid. 1991. **Hidden Markov Model in Text Recognition.** CRIN/SRNS/Inria Lorraine. Cedex.

Chen, Qing. 2003. **Evaluation of OCR Algorithm for Images with Different Spatial Resolutions and Noise.** School Of Information Technology and Engineering Faculty of Engineering University of Ottawa. Ottawa.

Connell, Scott. 1996. **A Comparison of Hidden Markov Model Features for the Recognition of Cursive Handwriting.** Michigan State University. Michigan.

Engkamat, Adeline Anak. 2005. **Enhancement Of Parallel Thinning Algorithm For Handwritten Character Using Neural Network.** Thesis. Faculty of Computer Science & Information Systems University Technology Malaysia.

Gonzalez, Rafael C. 1997. **Digital Image Processing**. Addison-Wesley Publishing.

Gregorius Budhi ST.,MT., dkk. Metode Jaringan Saraf Tiruan Backpropagation untuk Pengenalan Huruf Cetak pada Citra Digital**. Universitas Kristen Petra. Surabaya.**

Irfani, Angela, dkk. 2006. **Algoritma Viterbi dalam Metode Hidden Markov Models pada Teknologi Speech Recognition**.Institut Teknologi Bandung.

Jianying, Hu, dkk. 2007. **Handwriting Recognition With Hidden Markov Models And Gramatical Constraints**. AT&T Bell Laboratories. New Jersey.

Kuswara, Revi. 2006. **Proses OCR Dengan OmniPage Pro 10**. Elexmedia Komputindo. Jakarta.

Lim, Resmana, dkk. 2004. **Aplikasi OCR Dengan Metode Template Matching Untuk Pendukung Bermain Gitar**. Universitas Kristen Petra. Surabaya.

Munir, Rinaldi. 2004. **Pengolahan Citra Digital Dengan Pendekatan Algoritmik**. Penerbit Informatika Bandung.

Nur, Hassan. **Optical character recognition (OCR).** http://librarycorner.org. Diakses pada tanggal 28 Februari 2007.

Priyatma, J. Eka dan St. Eko Hari Parmadi**.** 2004. **Perluasan Algoritma Pengenal huruf Metode Tupel-N Memakai Logika Kabur**. Jurusan Ilmu Komputer Fakultas MIPA Universitas Sanata Dharma, Yogyakarta. Integral, 9 (3):108.

Suriaatmadja, Stefanus. 2007. **Pembuatan aplikasi perangkat lunak pengenalan huruf cetak pada file text hasil scanning dengan menggunakan metode kohonen neural network.** Skipsi Sarjana Jurusan Teknik Informatika Universitas Petra. Surabaya.

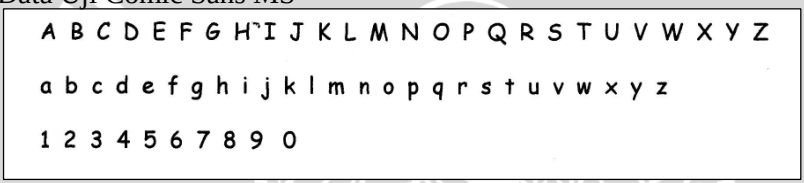
Velagapudi, Prasanna. **Using HMMs to Boost Accuracy in Optical Character Recognition**. Robotics Institute Carnegie Mellon University. Pittsburgh.

LAMPIRAN 1

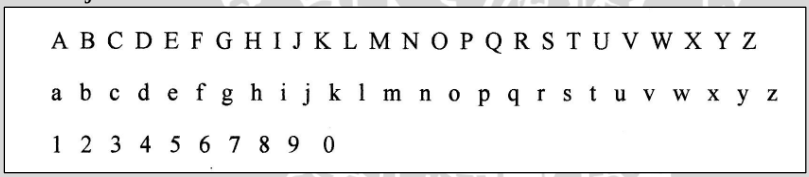
Data latih dan data uji yang digunakan

Data Uji Arial

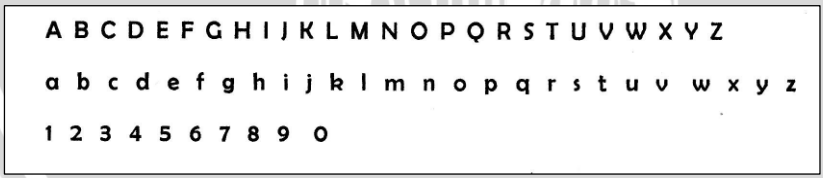
Data Uji Comic Sans MS



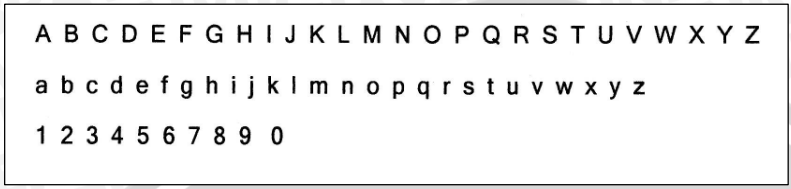
Data Uji Times New Roman



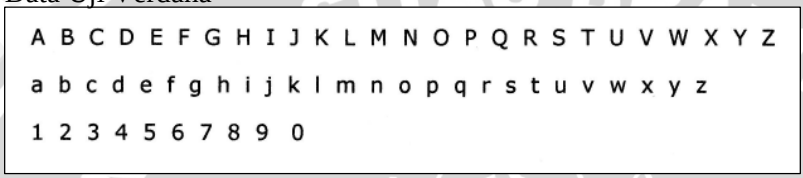
Data Uji Berlin Sans FB



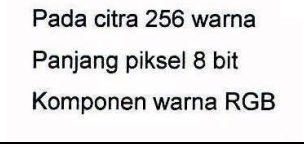
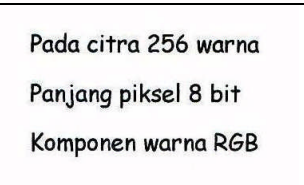
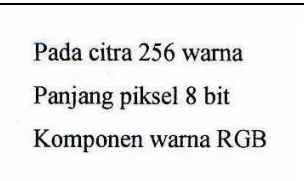
Data Uji Microsoft Sans Serif



Data Uji Verdana



Data Uji Dokumen

Uji1-Arial	
Uji1-Comic	
Uji1-TNR	
Uji1-Berlin	

Uji1-MS Serif	Pada citra 256 warna Panjang piksel 8 bit Komponen warna RGB
Uji1-Verdana	Pada citra 256 warna Panjang piksel 8 bit Komponen warna RGB
Uji2-Arial	Our components are Trained with a font base of 120 Font Families
Uji2-Comic	Our components are Trained with a font base of 120 Font Families
Uji2-TNR	Our components are Trained with a font base of 120 Font Families
Uji2-Berlin	Our components are Trained with a font base of 120 Font Families
Uji2-MS Serif	Our components are Trained with a font base of 120 Font Families
Uji2-Verdana	Our components are Trained with a font base of 120 Font Families

Uji3-Arial	Citra format Bitmap Berukuran 512x512 piksel Memori sebesar 32 KB
Uji3-Comic	Citra format Bitmap Berukuran 512x512 piksel Memori sebesar 32 KB
Uji3-TNR	Citra format Bitmap Berukuran 512x512 piksel Memori sebesar 32 KB
Uji3-Berlin	Citra format Bitmap Berukuran 512x512 piksel Memori sebesar 32 KB
Uji3-MS Serif	Citra format Bitmap Berukuran 512x512 piksel Memori sebesar 32 KB
Uji 3-Verdana	Citra format Bitmap Berukuran 512x512 piksel Memori sebesar 32 KB

LAMPIRAN 2

Hasil Uji Coba

a. Pengujian Jenis Huruf Arial

Karakter	Dikenali sebagai	Benar	Salah
0	0	✓	
1	1	✓	
2	2	✓	
3	3	✓	
4	4	✓	
5	5	✓	
6	6	✓	
7	7	✓	
8	8	✓	
9	9	✓	
A	A	✓	
B	B	✓	
C	C	✓	
D	D	✓	
E	E	✓	
F	F	✓	
G	G	✓	
H	H	✓	
I	I	✓	
J	J	✓	
K	K	✓	
L	L	✓	
M	M	✓	
N	N	✓	
O	O	✓	
P	P	✓	
Q	Q	✓	
R	R	✓	
S	S	✓	
T	T	✓	

U	U	✓	
V	V	✓	
W	W	✓	
X	X	✓	
Y	Y	✓	
Z	Z	✓	
a	a	✓	
b	b	✓	
c	c	✓	
d	d	✓	
e	e	✓	
f	f	✓	
g	g	✓	
h	h	✓	
i	i	✓	
j	j	✓	
k	k	✓	
l	l	✓	
m	m	✓	
n	n	✓	
o	o	✓	
p	p	✓	
q	q	✓	
r	r	✓	
s	s	✓	
t	t	✓	
u	u	✓	
v	v	✓	
w	w	✓	
x	x	✓	
y	y	✓	
z	z	✓	
Jumlah benar		62	
Jumlah salah			0
Keberhasilan		62/62*100% = 100%	

b. Pengujian Jenis Huruf Comic Sans MS

Karakter	Dikenali sebagai	Benar	Salah
0	0	✓	
1	1	✓	
2	2	✓	
3	3	✓	
4	4	✓	
5	5	✓	
6	6	✓	
7	7	✓	
8	8	✓	
9	9	✓	
A	A	✓	
B	B	✓	
C	C	✓	
D	D	✓	
E	E	✓	
F	F	✓	
G	G	✓	
H	H	✓	
I	I	✓	
J	J	✓	
K	K	✓	
L	L	✓	
M	M	✓	
N	N	✓	
O	O	✓	
P	P	✓	
Q	Q	✓	
R	R	✓	
S	S	✓	
T	T	✓	
U	U	✓	
V	V	✓	
W	W	✓	
X	X	✓	

Y	Y	✓	
Z	Z	✓	
a	a	✓	
b	b	✓	
c	c	✓	
d	d	✓	
e	e	✓	
f	f	✓	
g	g	✓	
h	h	✓	
i	i	✓	
j	j	✓	
k	k	✓	
l	l	✓	
m	m	✓	
n	n	✓	
o	o	✓	
p	p	✓	
q	q	✓	
r	r	✓	
s	s	✓	
t	t	✓	
u	u	✓	
v	v	✓	
w	w	✓	
x	x	✓	
y	y	✓	
z	z	✓	
Jumlah benar		62	
Jumlah salah			0
Keberhasilan		62/62*100% = 100%	

c. Pengujian Jenis Huruf Times New Roman

Karakter	Dikenali sebagai	Benar	Salah
0	0	✓	
1	1	✓	

2	2	✓	
3	3	✓	
4	4	✓	
5	5	✓	
6	6	✓	
7	7	✓	
8	8	✓	
9	9	✓	
A	A	✓	
B	B	✓	
C	C	✓	
D	D	✓	
E	E	✓	
F	F	✓	
G	G	✓	
H	H	✓	
I	I	✓	
J	J	✓	
K	K	✓	
L	L	✓	
M	M	✓	
N	N	✓	
O	O	✓	
P	P	✓	
Q	Q	✓	
R	R	✓	
S	S	✓	
T	T	✓	
U	U	✓	
V	V	✓	
W	W	✓	
X	X	✓	
Y	Y	✓	
Z	Z	✓	
a	a	✓	
b	b	✓	
c	c	✓	
d	d	✓	

e	e	✓	
f	f	✓	
g	g	✓	
h	h	✓	
i	i	✓	
j	j	✓	
k	k	✓	
l	l	✓	
m	m	✓	
n	n	✓	
o	o	✓	
p	p	✓	
q	q	✓	
r	r	✓	
s	s	✓	
t	t	✓	
u	u	✓	
v	v	✓	
w	w	✓	
x	x	✓	
y	y	✓	
z	z	✓	
Jumlah benar		62	
Jumlah salah			0
Keberhasilan		62/62*100% = 100%	

d. Pengujian Jenis Huruf Berlin Sans FB

Karakter	Dikenali sebagai	Benar	Salah
0	B		✓
1	j		✓
2	2	✓	
3	3	✓	
4	1		✓
5	5	✓	
6	f		✓

7	7	✓	
8	3		✓
9	0		✓
A	A	✓	
B	D		✓
C	C	✓	
D	D	✓	
E	z		✓
F	F	✓	
G	G	✓	
H	H	✓	
I	d		✓
J	J	✓	
K	V		✓
L	L	✓	
M	H		✓
N	4		✓
O	N		✓
P	P	✓	
Q	C		✓
R	D		✓
S	S	✓	
T	T	✓	
U	l		✓
V	W		✓
W	W	✓	
X	Y		✓
Y	W		✓
Z	Z	✓	
a	z		✓
b	b	✓	
c	c	✓	
d	4		✓

e	e	✓	
f	f	✓	
g	z		✓
h	h	✓	
i	j		✓
j	j	✓	
k	b		✓
l	l	✓	
m	m	✓	
n	p		✓
o	o	✓	
p	p	✓	
q	G		✓
r	F		✓
s	s	✓	
t	t	✓	
u	u	✓	
v	A		✓
w	w	✓	
x	x	✓	
y	w		✓
z	z	z	
Jumlah benar		33	
Jumlah salah			29
Keberhasilan		33/62*100% = 53.2%	

e. Pengujian Jenis Microsoft Sans Serif

Karakter	Dikenali sebagai	Benar	Salah
0	0	✓	
1	1	✓	
2	2	✓	
3	3	✓	

4	u		✓
5	5	✓	
6	c		✓
7	7	✓	
8	p		✓
9	9	✓	
A	A	✓	
B	F		✓
C	C	✓	
D	D	✓	
E	Z		✓
F	F	✓	
G	G	✓	
H	H	✓	
I	j		✓
J	J	✓	
K	X		✓
L	L	✓	
M	M	✓	
N	X		✓
O	R		✓
P	P	✓	
Q	Q	✓	
R	R	✓	
S	S	✓	
T	T	✓	
U	U	✓	
V	V	✓	
W	W	✓	
X	Y		✓
Y	Y	✓	
Z	Z	✓	
a	g		✓

b	b	✓	
c	c	✓	
d	k		✓
e	e	✓	
f	f	✓	
g	z		✓
h	h	✓	
i	i	✓	
j	l		✓
k	u		✓
l	l	✓	
m	R		✓
n	n	✓	
o	o	✓	
p	0		✓
q	G		✓
r	f		✓
s	s	✓	
t	t	✓	
u	k		✓
v	w		✓
w	w	✓	
x	x	✓	
y	v		✓
z	z	✓	
Jumlah benar		40	
Jumlah salah			22
Keberhasilan		40/62*100% = 64.5%	

f. Pengujian Jenis Huruf Verdana

Karakter	Dikenali sebagai	Benar	Salah
0	0	✓	

1	1	✓	
2	2	✓	
3	2		✓
4	4	✓	
5	E		✓
6	6	✓	
7	7	✓	
8	B		✓
9	9	✓	
A	A	✓	
B	O		✓
C	C	✓	
D	D	✓	
E	E	✓	
F	F	✓	
G	g		✓
H	H	✓	
I	I	✓	
J	q		✓
K	6		✓
L	L	✓	
M	M	✓	
N	N	✓	
O	Q		✓
P	P	✓	
Q	R		✓
R	O		✓
S	S	✓	
T	T	✓	
U	U	✓	
V	y		✓
W	W	✓	
X	K		✓

Y	Y	✓	
Z	Z	✓	
a	3		✓
b	b	✓	
c	c	✓	
d	d	✓	
e	e	✓	
f	f	✓	
g	z		✓
h	h	✓	
i	i	✓	
j	l		✓
k	k	✓	
l	l	✓	
m	m	✓	
n	a		✓
o	o	✓	
p	p	✓	
q	s		✓
r	r	✓	
s	s	✓	
t	t	✓	
u	k		✓
v	v	✓	
w	w	✓	
x	Y		✓
y	y	✓	
z	z	✓	
Jumlah benar		29	
Jumlah salah			33
Keberhasilan		43/62*100% = 69.3%	