PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENGENAL KARAKTER OPTIK BERBASIS LINUX

TUGAS AKHIR

Disusun Oleh :

ANDI IWAN NUR HIDAYAT
5197 100 017

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2003

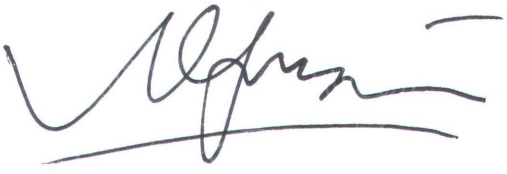
PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENGENAL KARAKTER OPTIK BERBASIS LINUX

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui,

Dosen Pembimbing



Ir. Much. Husni, M.Kom
NIP. 131411100

Surabaya
Agustus 2002

DAFTAR PUSTAKA

1. Gonzales, Rafael C., Paul Wirtz. Digital Image Processing. Addison-Wesley Publishing Company, 1977.
2. Gonzales, R. C., K. S. Fu, C. S. G. Lee. Robotics : Control. Sensing, Vision and Intellegence. Singapore: McGraw-Hill Book Company, 1987.
3. Pal, Sankar P., Dwijesh K. Dutta Majumder. Fuzzy: Pendekatan Matematik untuk Pengenalan Pola (terjemahan). Penerbit Universitas Indonesia, 1989.
4. Pratt, William K. Digital Image Processing. John Wiley & Sons, Inc., 1978.
5. Young, Tsay Y., King-Sun Fu. Handbook of Pattern Recognition and Image Processing. Academic Press, Inc., 1986.
6. Schildt, Herbert. Using Turbo C. McGrawHill, Inc., 1988.

KATA PENGANTAR

Alhamdulillah, puji syukur ke hadirat Allah SWT yang telah memberikan rahmat, dan hidayahnya sehingga penulis dapat menyelesaikan Tugas Akhir ini yang berjudul :

"PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENGENAL KARAKTER OPTIK BERBASIS LINUX".

Tugas Akhir ini disusun guna memenuhi salah satu sarat akademis bagi mahasiswa strata 1 (S1) untuk menyelesaikan studi di jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember di Surabaya.

Dengan terselesaikannya tugas akhir ini, penulis meyampaikan dengan segala kerendahan hati, penghargaan dan ungkapan terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah membantu baik moril maupun material dengan langsung maupun tidak langsung kepada :

1. Pak Husni sebagai pembimbing dan kasi Lab AJK yang telah memberikan izin untuk mengerjakan tugas akhir di Lab AJK.
2. Bu Ester sebagai dosen wali yang banyak memberikan kemudahan-kemudahan kepada penulis dalam menjalani birokrasi akademis yang ruwet.
3. Pak Karmono, Pak Pri, Yatno, Pak Atmoko,Pak Sugeng, Mas Yudhi dan kru satpam lainnya yang telah menjaga sepeda motor penulis dari jamahan tangan – tangan jahil selama berada di lingkungan TC.
4. Segenap Dosen dan karyawan yang telah banyak membantu penulis selama menjalani masa kuliah.

5. Kawan Putu(Sableng), Andik(Molen), Guruh, Arif(Mbah), Zola, Idi(Dingklek), Edi(Paimo), Gershom(Shomy), Roni Vijay, Victoria(Vicka), Hera, Daning, Eric dan segenap anggota C0D lainnya yang telah banyak sekali memberi dukungan moril dan materiil kepada penulis selama masa kuliah.
6. Seluruh anggota Lab AJK yang sekarang (Dwi, Alifah, Delis, Krebo, Surya, Roy(Shincan), Kamas(Sokam), Harry(AryGlewo), Ujik(Orang Lurus), Purna, dll) maupun yang telah punah tugas (Bornok, Doel (Paisexxx), Didit(Pacman), Heri(PekTonk), Tito(Gimball), Anank, dll), **Vivat AJK !!!!!!!!!.**
7. Kawan – kawan angkatan 96 keatas dan 98 ke bawah yang tidak bisa penulis sebutkan satu persatu disini.
8. Cak Sis yang telah merelakan sedikit banyak hartanya baik makanan ataupun minuman untuk diutang oleh penulis.
9. Saudara Imam Kuswardayan yang telah menjadikan hidup penulis lebih hidup.
10. Mbak Davi makasih banyak atas saran-saran yang telah diberikan manakala penulis dirundung masalah.
11. Block N .....hahahahaha.....hahahahahaha.....Entah.....Lah.....:), makasih banyak yah.....
12. Kakak-kakakku, yang kini telah jauh di mata namun selalu dekat dalam bayanganku, Adikku yang tercinta.
13. Kedua orang tuaku, yang telah membiayai dan mendoakan keselamatan dan kebahagiaanku dalam menjalani kehidupan ini.

ABSTRAK

Dalam beberapa tahun ini banyak dijumpai aplikasi yang berbasis Digital Image Processing. Pada dasarnya aplikasi-aplikasi tersebut melakukan dua hal pokok, yaitu perbaikan kualitas citra dan pemrosesan data citra secara otomatis dengan memanfaatkan peralatan komputer.

Dalam Tugas Akhir akan dibahas suatu topik yang berkaitan erat dengan cabang dari Digital Image Processing, yakni Pengenalan Pola, yang lebih khusus lagi memiliki lingkup aplikasi Pengenalan Karakter. Sistem pengenalan karakter yang dirancang dapat mengenali bentuk karakter alfabet yang baku maupun yang tidak baku, tetapi tetap memiliki batasan-batasan tertentu. Ada tahapan-tahapan yang harus dialalui dalam sistem pengenalan karakter alfabet ini, yaitu restorasi, representasi dan deskripsi, serta tahap pengenalan karakter.

DAFTAR ISI

DAFTAR GAMBAR

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dunia informasi semakin berkembang pesat. Orang selalu berusaha untuk mendapatkan atau mengolah informasi dengan cepat dan tepat. Hal tersebut dapat berkembang dengan efesien apabila dutunjang dengan perkembangan komputer baik dari segi sortware maupun dari segi hardware.

Scanner adalah salah satu produk hardware yang berfungsi untuk mendapatkan informasi. Sebuah gambar ataupun naskah dapat dibaca dan diolah oleh komputer melalui scanner ini. Dengan bantuan scanner maka orang tidak perlu lagi mengetik ulang suatu naskah atau dokumen. Tetapi karena hasil pembacaan dari scanner ini berbentuk gambar (dalam mode grafik), bukan dalam bentuk teks seperti naskah (dokumen) yang dibuat dengan menggunakan word processor pada umumnya, maka naskah hasil scan tersebut susah untuk diedit. Untuk itu dibutuhkan suatu tool, dalam hal ini berbentuk software aplikasi (program) yang berfungsi untuk mengenali karakter–karakter hasil dari scanner untuk selanjutnya diubah menjadi karakter–karakter dalam mode teks, sehingga naskah tersebut bisa dikenali oleh komputer dan mudah untuk diedit kembali.

1.2 Perumusan masalah

Suatu citra (gambar) yang diperoleh dari scanner adalah merupakan kumpulan dari titik-titik elemen gambar (piksel) dengan warna-warna tertentu. Begitu juga apabila suatu naskah (dokumen) di-scan, maka hasilnya adalah kumpulan pikse-piksel yang membentuk gambar dari karakter-karakter yang sama dengan naskah aslinya.

Dalam Tugas Akhir ini akan dibuat suatu perangkat lunak (software) yang berfungsi untuk mengenali karakter-karakter yang terbentuk dari elemen-elemen gambar untuk diubah menjadi karakter-karakter dalam mode teks, yang berjalan dalam sistem operasi Linux Redhat 7.2.

1.3 Pembatasan Masalah

Untuk menunjang dalam penyelesaian tugas akhir ini perlu dilakukan adanya pembatasan masalah-masalah agar tercapai tujuan. Adapun batasan masalah tersebut adalah sebagai berikut :

1. Sistem operasi yang digunakan adalah Linux *RedHat 7.2.*
2. Karakter-karakter atau simbol-simbol yang dapat dikenali adalah karakter-karakter standar, yang merupakan hasil cetakan dari printer yang telah di-scan, dimana karakter atau simbol tersebut telah disimpan dalam suatu library.
3. Tiap karakter atau simbol yang akan dikenali, harus terhubung, tidak boleh terputus-putus.

4. Satu karakter atau simbol dengan karakter atau simbol lainnya harus terpisah-pisah, tidak boleh terhubung antara satu dengan lainnya seperti tulisan latin.

1.4 Tujuan

Dalam Tugas Akhir ini akan dirancang suatu sistem (paket program) yang diharapkan dapat mengenali karakter-karakter yang dibentuk oleh pikse-piksel (grafik) untuk diubah menjadi karakter-karakter dalam mode teks sehingga dapat dikenali oleh word processor yang ada dalam sistem operasi Linux Redhat 7.2, misalnya Emac, Star Office, KOffice dan lain-lain. Adapun metode yang digunakan yaitu menggunakan metode skeleton fitting. Setiap karakter yang akan dikenali dideskripkan lebih dahulu dan dianalisa untuk mendapatkan sifat-sifat atau ciri-ciri yang dimiliki karakter yang bersangkutan. Berdasarkan ciri-ciri yang dimiliki oleh karakter atau simbol tersebut, diharapkan karakter tersebut dapat dikenali.

1.5 Metodologi Tugas Akhir

Metodologi yang digunakan untuk menyelesaikan tugas akhir ini adalah :

1. **Studi literatur dan Pemahaman Sistem**

   Pada tahap ini hal yang dilakukan adalah mencari, mengumpulkan dan mempelajari segala macam informasi yang berhubungan dengan sistem operasi Linux, konsep image processing, bahasa pemrograman C untuk sistem operasi Unix/Linux , pemrograman X Window pada Unix dengan

menggunakan *APIs :libarary X11*. Data penunjang ini didapat dari buku dan internet.

2. **Perancangan Perangkat Lunak**

Pada tahap ini dilakukan perancangan sistem perangkat lunak yang akan dibuat dan analisis data yang telah dikumpulkan. Diharapkan dari perancangan ini didapatkan desain yang dapat memenuhi kebutuhan sistem.

3. **Implementasi**

Pada tahap ini dilakukan implementasi dari tahap perancangan sistem dan algoritma yang digunakan pada bahasa pemrograman yang dipilih.

4. **Evaluasi dan Revisi**

Pada tahap ini dilakukan uji coba dan revisi jika terdapat kesalahan pada perangkat lunak yang telah diimplementasikan dan pada akhirnya diharapkan sesuai dengan tujuan dan manfaat dari tugas akhir ini.

5. **Penulisan Laporan Tugas Akhir**

Tahap akhir dari proses tugas akhir ini adalah pembuatan laporan atau dokumentasi secara lengkap dan menyeluruh dari semua kegiatan yang telah dilakukan.

1.6 Sistematika Pembahasan

Adapun sistematika yang dipakai dalam pembahasan laporan tugas akhir ini adalah :

BAB I Merupakan pendahuluan yang meliputi latar belakang, permasalahan, tujuan, pembatasan masalah dan sistematika laporan.

BAB II Membahas tentang dasar teori yang digunakan pada tugas akhir ini, mengenai email, dan pemrograman jaringan pada Unix/Linux . Dasar teori yang disajikan adalah sesuai dengan ruang lingkup maslaah yang diteliti, yaitu dalam lingkup sistem operasi Linux dan masalah aplikasi clustering pada multiple mail server dengan menggunakan metode direct routing.

BAB III Membahas tentang perancangan dan implementasi perangkat lunak.

BAB IV Membahas tentang Uji Coba dan Evaluasi perangkat lunak yang dihasilkan, baik ditinjau dari segi pengoperasiannya maupun unjuk kerja/hasil yang diberikan.

BAB V Membahas tentang penutup dari tugas akhir ini. Bab ini memberikan kesimpulan terhadap penelitian yang telah dilakukan serta saran-saran untuk pengembangan di kemudian hari

BAB II

DASAR TEORI

BAB II

DASAR TEORI

Bab ini berisi dasar teori dan pengetahuan dasar umum yang penting dan perlu diketahui untuk memahami proses perancangan dan penyusunan perangkat lunak aplikasi pengenal karakter optik dengan menggunakan metode skeloton. Beberapa bagian hanya akan dituliskan seperluanya, sehingga mungkin diperlukan untuk membaca refrensi yang dipergunakan. Seluruh refrensi yang dipergunakan tercantum pada bagian akhir dari tugas akhir ini.

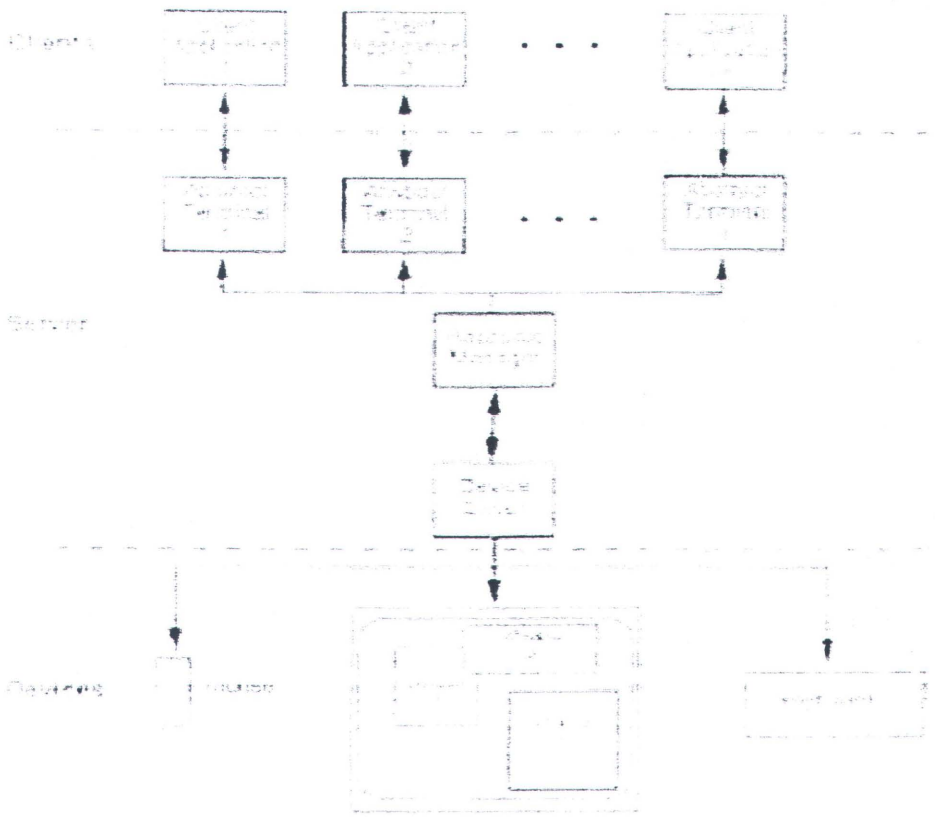
2.1 Sistem X Window

Sistem X Windows pertama kali ditemukan pada tahun 1984 di Massachusetts Institu of Technology (MIT). X Window yang dikembangkan oleh MIT dan dipakai secara luas di UNIX, dapat dipakai di Linux. Dimana XFree86.x (nama program X di Linux) merupakan free software juga. Versi ter- baru dari XFree86 ialah XFree86 3.1.1 dimana versi ini dapat disamakan dengan versi X11R6 pada Sun OS dll.

X WIndow system sendiri merupakan suatu standar industri untuk Graphical User Interface (Dix et al, 1995). Yang membedakan X Window dengan sistem GUI lainnya sehingga menjadikannya sebagai standard industri adalah X berdasarkan protokol jaringan yang secara jelas mendefinsikan komunikasi client-server. Sehingga membuat X lebih bersifat device independent. Sehingga client dan server tidak harus berada di satu mesin yang sama untuk dapat berinteraksi.

Client atau server pada X Window diasosiasikan sebagai terminal abstrak atau window, seperti yang dijelas pada gambar 2.1.

Dasar design dari X adalah tak bergantung dari sistem operasi. Fleksibilitas X dan ketidak terkaitannya dengan satu perangkat keras ataupun satu jenis User interface berkaitan dengan sejarah perkembangan X tersebut. Dimulai di tahun 1985 di MIT sebagai Project Athena yang didukung oleh DEC dan IBM. X ini direncanakan agar dapat mengakses dan memanfaatkan beragam sumber komputasi yang ada. Versi pertama X berjalan pada mesin VAXstation II/GPX dengan sistem operasi Ultrix. Ini adalah versi X10R3. Kemudian vendor-vendor lain juga menerapkan X ini dan hingga akhirnya menjadi standard industri. Pada tahun 1986 disain X menjadi X11. Jadi sistem X ini termasuk sudah matang dan sudah mengalami studi secara ilmiah yang cukup lama dan telah digunakan secara luas sebelumnya. Keuntungan penggunaan X dibanding GUI biasa adalah :



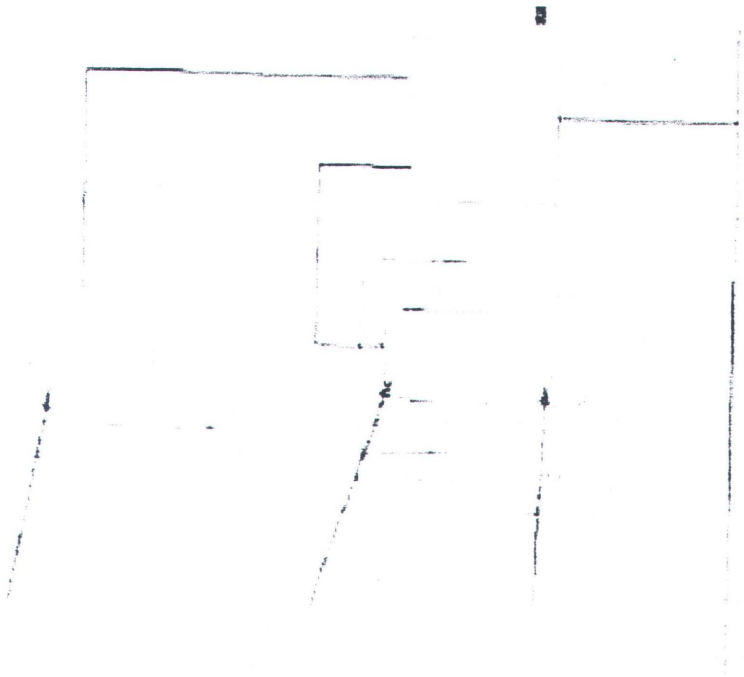
Gambar 2.1

Abstraksi pada X Window (Dix at al, 1994)

- **Tranparansi jaringan**. Sistem X Window memungkinkan menampilkan aplikasi di manapun pada workstation manapun. Sehingga sistem bisa menjalankan program (dieksekusi bukan hanya program tersebut disimpan di sistem remote tersebut seperti pada konsep file server) dan ditampilkan di workstation yang lain. Ini bisa mengatasi keterbatasan sumber daya processor.

Suatu workstation yang tidak memiliki processor tertentu dapat menjalankan program tersebut di mesin yang berprosesor lain dan menambilkan hasilnya di workstationnya sendiri. Ini bukan saja pada jaringan lokal juga pada jaringan Wide Area.

- **Pemisahan komputasi dan grafik**. Pada X Window dilakukan pemisahan yang jelas antara grafik dan komputasi. Sehingga memungkinkan aplikasi dijalankan di komputer Cray, dan hasilnya dilihat di komputer PC486.
- **Berbagai sistem yang tersedia di X**. Dengan konsep client-server di X ini menjadikan tersedia beragam aplikasi oleh vendor yang berbeda untuk X. Bahkan ada X Window di sistem MS Window ataupun lainnya.
- **X menentukan mekanisme bukan kebijakan**. Sehingga protokol X hanya menentukan task utama yang dilakukan oleh X server. Sedangkan menu, button, label tidak dinyatakan pada protokol X. Applikasi akan menyusun user interfacenya sendiri. Sehingga memungkinkan membuat look and feel yang seperti apapun, disesuaikan dengan kebutuhan dan latar belakang pengguna.
- **Terbukanya kesempatan untuk menambahkan kemampuan**. Misal dengan mudah ditambahkan window manager, desktop environment, dan lainnya tanpa perlu terikat dengan X server yang digunakan.



Gambar 2.2

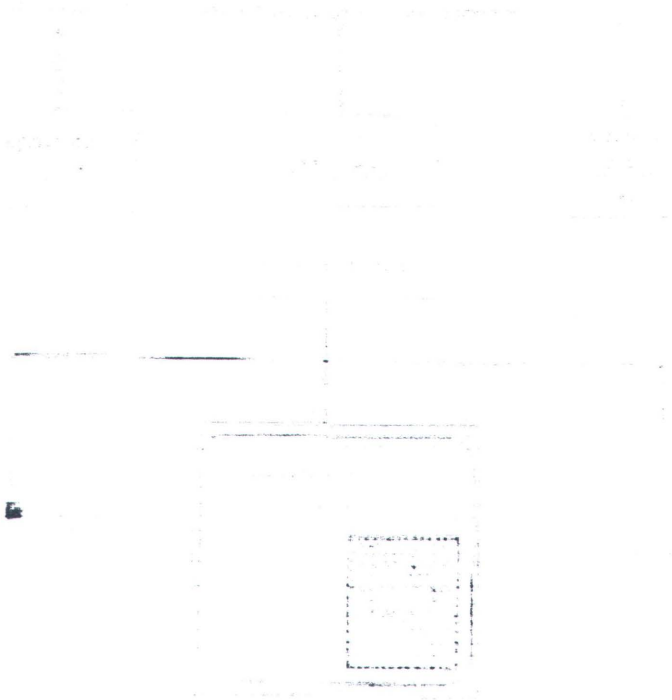
Penamaan yang transparan pada X Window

2.1.1 Sistem Client/Server

X Window adalah kombinasi antara beberapa hal sebagai berikut :

- X protocol
- X display server (X server)
- X client. Yang merupakan suatu aplikasi yang memanfaatkan display pada workstation
- Xlib routine

Adapun komponen-komponen yang membangun sistem X Window dapat dijelaskan pada gambar 2.2



Gambar 2.3

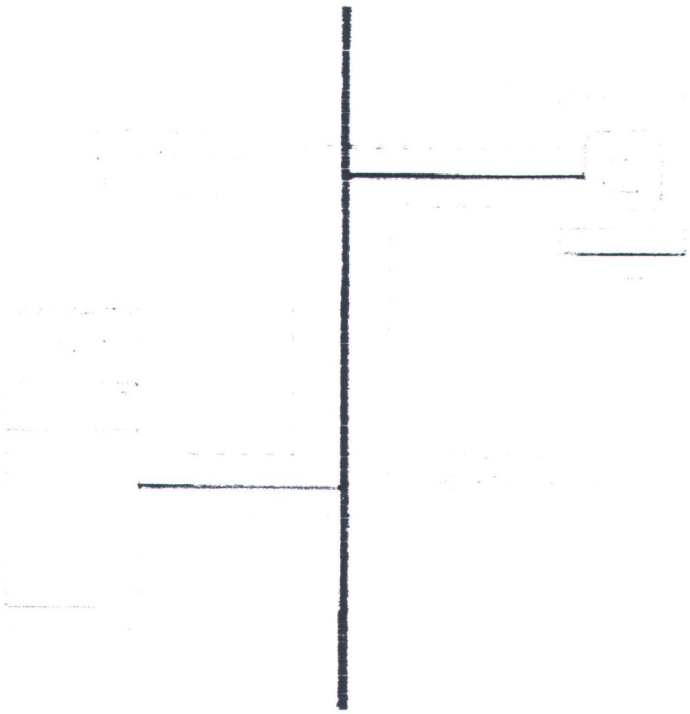
Komponen pada X Window

X display server (atau disingkat X Server), berjalan di tiap workstation dan siap menerima request untuk menampilkan apa yang diminta client. Suatu X server sendiri berperan dalam melakukan tugas :

- Menentukan izin suatu aplikasi untuk menampilkan hasilnya di layar.
- Mengintepretasikan permintaan dari client untuk melakukan operasi client dan menyediakan informasi lainnya.

- Melakukan de-multipleksing event dari masukan user dan memberikan pada client yang sesuai.
- Meminimalkan lalu lintas di jaringan, dengan memungkinkan client menyimpan informasi display, seperti informasi font dan sebagainya.

Prinsip kerja Client/Server pada X Window dapat dijelas sebagai berikut



gambar 2.4

Prinsip client-server pada X Window

2.2 Pengolahan Citra Digital

Suatu citra (image) adalah fungsi intensitas cahaya dua dimensi $f(x,y)$, dimana x dan y menyatakan koordinat-koordinatnya dan nilai f pada titik-titik (x,y) menyatakan tingkat intensitas (grey level) dari titik-titik itu.[1]

Citra digital (digital image) adalah suatu citra $f(x,y)$ yang telah didiskritkan baik koordinat spasialnya maupun tingkat intensitasnya.[2]

Istilah Pengolahan Citra Digital *digital image processing) secara umum menunjuk pada pengolahan citra $f(x,y)$ dua dimensi oleh komputer digital, atau dengan pengertian pengolahan data-data digital array dua dimensi.

2.2.1 Image Enhancement

Yang dimaksud dengan image enhancement atau perbaikan citra adalah proses yng dikenakan pada suatu citra yang bertujuan memperoleh hasil yang lebih baik dari citra aslinya, yang sesuai dengan kriteria untuk aplikasi tertentu. Dikatakan untuk aplikasi tertentu karena tidak ada proses enhancement standar yang sesuai untuk semua aplikasi, misalnya proses enhancement yang terbaik untuk pengenalan pola belum tentu merupakan proses enhancement yang terbaik untuk proses sharpening (penajaman gambar).

Secara global metode yang dapat digunakan untuk perbaikan citra dapat dibagi menjadi dua bagian utama, yaitu metode yang berdasarkan Bidang Spasial dan metode yang berdasarkan Bidang Frekuensi.

---

[1] Rafael C Gonzales, Digital Image Processing, hal. 5
[2] Rafael C Gonzales, Digital Image Processing, hal 5

Proses enhancement yang berdasarkan pada bidang frekuensi yaitu metode yang prosesnya didasarkan pada modifikasi Transformasi Fourier dari citra input. Karena proses enhancement yang digunakan dalam Tugas Akhir ini hanya metode yang berdasarkan pada bidang spasial, maka metode enhancement yang berdasarkan pada bidang frekuensi tidak akan dibahas lebih lanjut lagi.

Sedangkan metode bidang spasial adalah suatu teknik pengolahan citra yang didasarkan pada pengoperasian secara langsung terhadap piksel-piksel yang membentuk citra itu sendiri

Fungsi pengoperasian dalam bidang spasial dapat diekspresikan dengan persamaan sebagai berikut :

$$G(x,y) = h(f(x,y))$$

Dimana $f(x,y)$ adalah citra masukan (input), $G(x,y)$ adalah citrahasil pengoperasian (output) dan h adalah suatu operator pengoperasian. Bentuk h yang sering digunakan untuk perbaikan citra adalah konvolusi mask (window atau filter). Mask adalah suatu array dua dimensi, dimana koefisien-koefisiennya dapat dipilih sesuai dengan kebutuhan.

Misalkan suatu citra yang mempunyai piksel-piksel yang terisolasi, dimana intensitas piksel-piksel tersebut berbeda dari intensitas latar belakannya, maka titik ini dapat dideteksi dengan menggunakan mask seperti gambar 2.1

-1	-1	1-
-1	8	-1
-1	-1	-1

Gambar 2.5

Mask untuk mendeteksi piksel yang terisolasi

Prosedur pengoperasiannya adalah sebagai berikut : mask digerakkan mulai dari posisi kiri atas ke seluruh bagian matrik citra, masing-masing piksel yang posisinya terlingkupi mask dikalikan dengan koefisien dari mask yang bersangkutan yaitu bagian pusat dikalikan 8 dan disekitarnya dikalikan dengan –1. Hasil dari sembilan perkalian ini kemudian dijumlahkan. Apabila jumlahnya sama dengan 0, berarti piksel-piksel yang berada pada mask mempunyai intensitas yang sama. Apabila hasil penjumlahan tidak sama dengan 0, maka titik yang tepat berada pada pusat mask adalah titik yang terisolasi. Apabila titik yang terisolasi tidak berada pada pusat mask, hasil penjumlahan juga tidak sama dengan 0, tetapi intensitasnya akan melemah, ini bisa dihilangkan dengan memberikan harga batas atau threshold.

2.2.2 Image Smoothing.

Proses smoothing biasanya digunakan untuk mengurangi atau mengilangkan noise dalam citra digital yang disebabkan karena proses pengambilan citra (sampling) yang kurang bagus. Atau dengan kata lain proses

smoothing bertujuan untuk menghilangkan gambar (piksel-piksel) yang bukan merupakan bagian dari citra.

2.2.2.1 Neighborhood Averaging.

Suatu citra $f(x,y)$ dengan ukuran N x N akan dibuat suatu citra yang bebas dari noise $G(x,y)$, yang merupakan hasil dari pengolahan terhadap citra $f(x,y)$ dengan menggunakan metode Neighborhood Averaging. Caranya adalah setiap piksel (x,y) pada g ditentukan dengan menghitung besar harga rata-rata grey level dari piksel-piksel yang berada disekitar (tetangga) piksel (x,y) pada citra f atau dapat dituliskan dalam persamaan sebagai berikut :

$$g(x,y) = \frac{1}{M} \sum_{(n,m)\in S} f(n,m)$$

Dimana x,y = 0,1, . . . , N – 1.

S adalah set dari koordinat piksel-piksel yang merupakan tetangga tetapi bukan termasuk piksel pada (x,y).

M adalah banyaknya piksel yang didefinisikan dalam S,

Tetangga-tetangga dari suatu piksel (S) bisa ditentukan berdasarkan titik –titik yang ada di dekat piksel (x,y) seperti contoh berikut :

$$S = ((x,y-1),(x,y+1),(x-1,y),(x+1,y))$$

Atau lebih jelasnya tampak pada gambar 2.2 (a).

Atau bisa juga berdasarkan titik-titik yang mengeleilingi piksel (x,y) seperti berikut:

$$S=((x,y-1),(x+1,y-1),(x+1,y),(x+1,y+1),(x,y+1),(x-1,y+1),(x-1,y),(x-1,y-1))$$

Atau lebih jelasnya tampak pada gambar 2.6 (b).[3]

(a) (b)

Gambar 2.6

Tetangga dari suatu piksel

2.2.2.2 Lowpass Filter.

Noise di dalam suatu citra biasanya mempunyai spektrum frekuensi spasial yang lebih tinggi daripada komponen citra normal. Sehingga lowpass spasial filter diharapkan secara efektif dapat menghilangkan noise.

---

[3] Rafael C Gonzales, Digital Image Processing, hal 137

Suatu citra hasil (output) $Q(m,m)$ berukuran M x M dibentuk dari konvolusi diskrit citra input $f(n,n)$ berukuran N x N dengan operator H berukuran L x L dengan persamaan berikut :

$$Q(m1,m2) = \sum_{n1} \sum_{n2} F(n1,n2)H(m1-n1+1,m2-n2+1)$$

untuk noise smoothing H dapat berbentuk Lowpass dengan semua konstanta positif. Beberapa bentuk filter (array) Lowpass adalah sebagi berikut :

mask 1 $$H = \frac{1}{91}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

mask 2 $$H = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

mask 3 $$H = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Mask ini, yang sering disebut dengan noise–cleaning mask, dinormalkan dengan pemberatan unit sehingga proses pembersihan noise tidak terjadi bias intensitas dalam pemrosesan citra.[4]

2.2.2.3 Median Filter.

Metode smoothing yang lain yaitu dengan menggunakan median filter. Filter ini hampir sama dengan average filter tetapi nilai titik di pusat mask bukan merupakan harga rata-rata dari nilai titil-titik disekitarnya, tetapi nilai titik di pusat mask adalah median dari kumpulan nilai titik-titik disekitarnya. Misalnya nilai titik-titik disekitar pusat mask adalah sebagai berikut : (10,20,20,25,15,20,30,25), maka median (nilai tengahnya) adalah 20. Sehingga nilai titik pada pusat mask adalah 20.

Pada prinsipnya fungsi dari median filter adalah memaksa titik dengan intensitas terang menjadi seperti sekelilingnya. Hasil dari median filter ini akan lebih baik dari pada average filter, yaitu citra tetap tajam dibandingkan dengan average yang hasilnya agak kabur

2.3 Pengolahan Pola

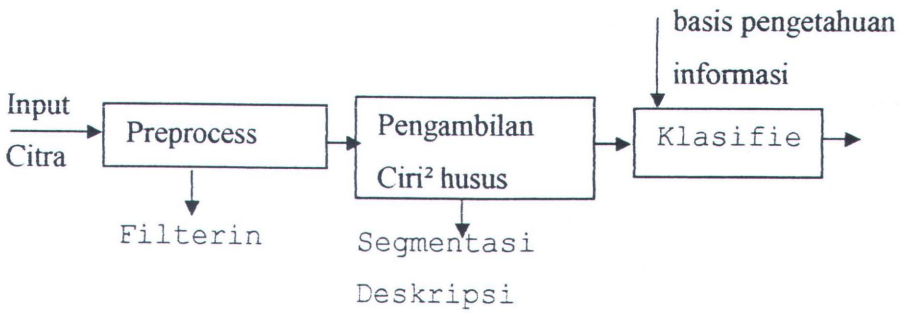
2.3.1 Pengertian Umum

Secara umum yang dimaksud dengan sistem pengenalan pola yaitu suatu sistem yang mengolah/memproses suatu obyek (citra) untuk diklasifikasikan

---

[4] William K Pratt, Digital Image Processing, hal. 319.

menjadi kelas-kelas sesuai dengan yang diinginkan atau dalam gambar blok tampak sebagai berikut :



Gambar 2.7

Gambar blok dari sistem pengenalan pola

Citra input, bisa diperoleh dari alat-alat pengambil gambr, misalnya kamera atau scanner dimana citra tersebut telah diubah menjadi citra digital, atau bisa juga diperoleh dengan langsung menggambar misalnya melalui mouse, keybord dan lain-lain.

Citra hasil input ini seringkali tidak sesuai dengan apa yang diinginkan. Agar proses pengenalan dapat berjalan dengan benar, maka citra input ini harus diproses terlebih dahulu (preprocessing). Tujuan utama dari preprocessing ini adalah untuk mempermudah memperoleh ciri-ciri khusus dari citra, sehingga nantinya dapat dikenali dengan cepat dan benar.

2.3.2 Preprocessing.

Pada tahap proprocessing biasanya dilakukan pengolahan terhadap citra input dengan tujuan antara lain untuk menghilangkan noise atau piksel-piksel yang terisolasi, menghilangkan patahan-patahan, menghaluskan permukaan, mempertajam gambar dan lain-lain. Untuk melakukan ini biasanya digunakan

proses smoothing seperti yang telah dijelaskan di atas. Dalam Tugas Akhir ini metode yang digunakan adalah teknik skeleton (penulangan) yang populer disebut metode Thinnig. Metoe Thinning digunakan untuk mendapatkan kerangka (tulang) dari suatu citra. Hal ini dimaksudkan untuk memudahkan pendeteksian piksel demi piksel dari suatu citra digital.

Algoritma Thinning.

Piksel-piksel pada lintasan tertutup bergray level 1, sedangkan yang terletak di luar bergray level 0.

- Step 1 : Tandai piksel-piksel pada lintasan tertutup yang memenuhi syarat-syarat sebagai berikut :

$2 \leq N(P1) \leq 6$

$S(P1) = 1$

$P_2P_4P_6 = 0$

$P_4P_6P_8 = 0$

- Step 2 :

Syarat a dan b sama dengan step 1

$P_2P_4P_8 = 0$

$P_2P_6P_8 = 0$

Ketarangan :

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Misalkan yang diperiksa piksel P_1 (titik tengah) itu perlu dihapus atau tidak.

N (P_1) adalah jumlah total piksel-piksel yang mempunyai gray level 1 ($P_2 + P_3 + P_4 + \ldots + P_9$)

S(P_1) adalah jumlah perhubungan dari gray level 0 ke 1 dari piksel-piksel tetangga.

Contoh :

1	1	0
0	1	0
1	0	1

Mask bedimensi 3 x 3 di atas memiliki $N(P_1) = 4$ dan $S(P_1) = 3$.

Langkah-langkah Thinning (dalam satu iterasi) :

Lakukan step 1.

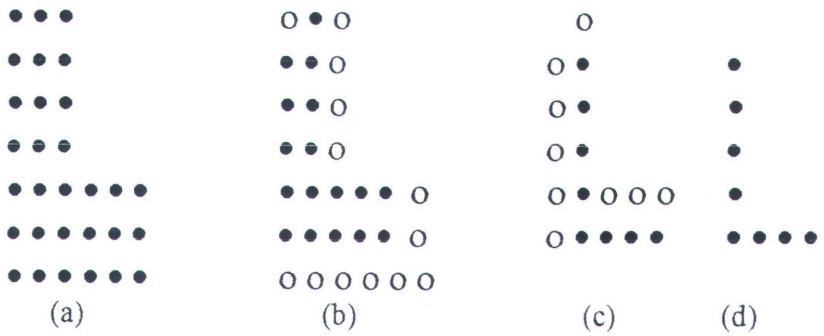
Hapus piksel-piksel yang ditandai pada step 1.

Ulangi step 2.

Hapus piksel-piksel yang ditandai pada step 2.

Langkah-langkah di atas diulangi sampai tidak ada piksel yang terhapus.

Adapun contoh implementasi metode Thinning dapa dilihat sebagai berikut :



o : pixel yang terhapus

• : pixel yang tidak terhapus

Gambar 2.8

Citra yang sesungguhnya (b) Setelah melewati

step 1 (c) Setelah melewati step 2 (d) Hasil akhit

2.3.3 Pengambilan Ciri-ciri Khusus.

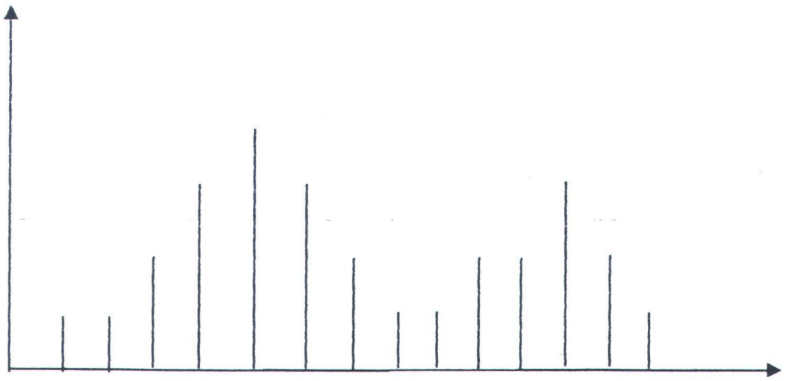
Untuk mendapatkan ciri-ciri khusus dari suatu citra, maka diperlukan dua hal berikut, yaitu segmentasi dan deskripsi.

2.3.3.1 Segmentasi.

Tujuan dari segmentasi adalah untuk membagi-bagi citra menjadi bagian-bagian yang akan diproses lebih lanjut, yaitu dibagi menjadi gambar-gambar kecil yang akan dikenali. Dalam aplikasi pengenal karakter atau simbol, dimana citra input terdiri dari kumpulan beberapa karakter atau simbol, maka masing-masing karakter atau simbol akan dipisah-pisahkan, kemudian masing-masing karakter atau simbol tersebut akan dikenali.

2.3.3.1.1 Metode Thresholding.

Salah satu metode segmentasi adalah dengan menggunakan metode thresholding. Pada metode ini gray level tiap piksel akan dibandingkan dengan sebuah nilai batas (threshold). Sebagai contoh histrogram intensitas yang ditunjukkan pada gambar 2.9 menggambarkan suatu citra $f(x,y)$ yang tersusun atas objek yang terang pada latar belakang gelap



Gambar 2.9

Histrogram intensitas suatu citra

Piksel-piksel citra tersebut intensitasnya terkelompok menjadi dua. Teknik untuk memisahkan obyek dari latar belakangnya adalah dengan memilih suatu intensitas tertentu untuk membedakan intensitas obyek dengan intensitas latar belakangnya. Intensitas tertentu tersebut dinamakan dengan threshold (T).

Piksel-piksel dimana $f(x,y) > T$ merupakan piksel-piksel obyek dan piksel-piksel dimana $f(x,y) \leq T$ merupakan piksel-piksel latar belakang. Untuk membuat citra biner $g(x,y)$ dengan menggunakan metode threholding dapat digunakan difinisi sebagai berikut

$g(x,y) = 1$ jika $f(x,y) >$ T yaitu piksel obyek

$g(x,y) = 1$ jika $f(x,y) \leq$ T yaitu piksel latar belakang

2.3.3.1.2 Metode Edge Detection.

Metode segmentasi yang lain yaitu dengan jalan mendeteksi tepi dari suatu citra, dengan menggunakan suatu operator yang disebut dengan gradien operator. Misalkan bentuk gradien operator tampak seperti pada gambar 2.10

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Gambar 2.10

(a). Untuk mendeteksi edge horisontal (b). Untuk mendeteksi edge vertikal

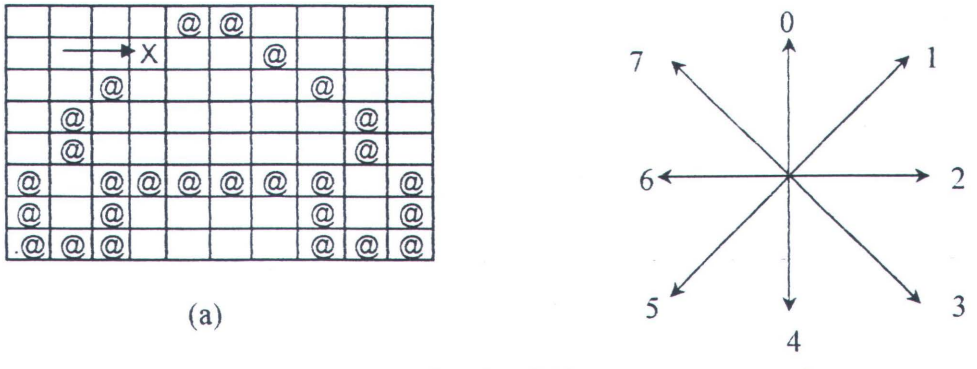
2.3.3.2 Deskripsi.

Deskripsi adalah cara kita untuk menggambarkan suatu citra, termasuk juga cara kita untuk menyimpan informasi dari gambar sehingga suatu saat kita dapat memperoleh kembali gambar tersebut tanpa berubah dari asalnya.

2.3.3.2.1 Deskripsi Regional.

Ketika region dari sebuah citra sudah ditemukan, maka karakteristik dari citra tersebut dalam hal ini garis tepinya (boundary) bisa dicari dengan menggunakan deskriptor. Tujuan dari digunakannya deskriptor tidak hanya untuk mengurangi besar data yang disimpan, tetapi juga untuk mengambil ciri khusus (feature) yang membedakan antara satu citra dengan citra lainnya. Salah satu contoh dari deskriptor regional adalah Deskriptor Kode Rantai.

Setelah titik-titik pada boundary ditemukan, maka dari titik-titik itu dapat dibentuk suatu kode yang disebut kode rantai (chain code).



(a)

Gambar 2.11

(a) Contoh boundary (b) Kode rantai 8 arah

contoh dari kode rantai untuk deskripsi citra seperti pada gambar 2.11 dengan menggunakan kode 8 arah adalah sebagai berikut :

1 , 2 , 3 , 3 , 3 , 4 , 3 , 4 , 4 , 6 , 6 , 0 , 0 , 6 , 6 , 6 , 6 , 5 , 4 , 6 , 6 , 0 , 0 , 1 , 0 , 1 , 1

agar tidak sensitif terhadap perputaran maka kode tersebut diubah menjadi bentuk selisih (beda) dari angka-angka pembentuk kode, yaitu diubah menjadi :

0 , 1 , 1 , 0 , 0 , 1 , 7 , 1 , 0 , 2 , 0 , 2 , 0 , 6 , 0 , 0 , 0 , 7 , 7 , 2 , 0 , 2 , 0 , 1 , 7 , 1 , 0

Agar tidak sensitif terhadap titik awal (inisial) dari penelusuran maka kode tersebut diputar (rotate) dengan mencari misalnya nilai numerik dari kode tersebut untuk nilai yang terkecil, yaitu diubah menjadi sebagai berikut :

0 , 0 , 0 , 7 , 7 , 2 , 0 , 2 , 0 , 1 , 7 , 1 , 0 , 0 , 1 , 1 , 0 , 0 , 1 , 7 , 1 , 0 , 2 , 0 2 , 0 , 6

Disamping itu untuk mediskripsikan suatu objek terdapat suatu cara yang disebut sebagai *topologycal descriptor* yang menggunakan beberapa paramter yang unik antara lain :

- *Topologycal descriptor* yang berpatokan pada jumlah hole dan jumlah *connected component* yang terdapat pada suatu region. Connected component terjadi, yaitu bilamana lintasan (garis) yang menghubungkan dua pixel (di dalam region) letaknya di dalam region tersebut. Karakter "A" memiliki suatu hole dan suatu connected component, sedangkan karakter "B" memiliki suatu connected component dan dua hole.
- *Topologycal descriptor* yang berpatokan pada pengurangan dari jumlah connected component (C) dengan jumlah hole (H). Jadi, E = C –H. berdasarkan rumus di atas, maka Karakter "A" mempunyai angka Euler 0 dan Karakter "B" mempunyai angka Euler –1

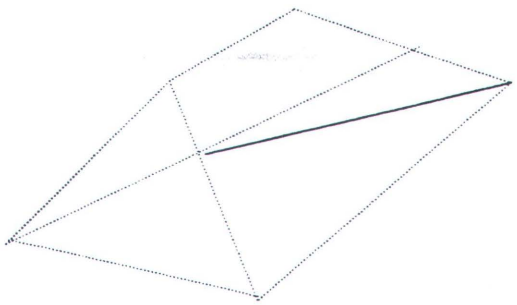
- Untuk objek yang lebih komplek, seperti polygonal network selain berpatokan pada angka Euler, dapat juga mengambil patokan pada jumlah face (F), jumlah edge (Q), dan jumlah titik sudut (w). Kedua patokan tersebut memiliki hubungan sebagai berikut :

$$W - Q + F = C - H = E$$

Persamaan di atas dinamakan rumus Euler.

Gambar 2.12 memiliki 7 titik sudut, 11 edge, 2 face, 1 connected region, dan 3 hole, sehingga rumus Eulernya menjadi :

$$7 - 11 + 2 = 1 - 3 = -2$$



Gambar 2.12

Region yang berisi polygonal network

2.3.3.2.2 Deskripsi Similaritas (Kesamaan)

Deskripsi similaritas berfungsi untuk melihat kesamaan dari dua buah citra. Contoh dari deskriptor ini adalah korelasi antara dua citra. Korelasi ini dapat dipakai untuk menentukan apakah suatu citra sama (similar) dengan citra lain

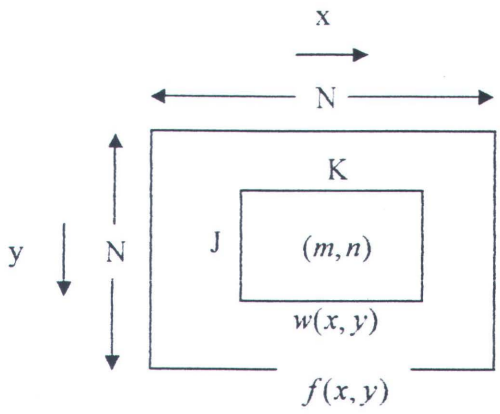
(model). Dalam hal ini deskripsi ini digunakan dalam aplikasi Template Matching.

Diberikan suatu citra digital $f(x,y)$ dengan ukuran M x N. Akan dicari apakah terdapat region $w(x,y)$ dengan ukuran J x F, dimana J < M dan K < N. Salah satu metode yang sering digunakan untuk memecahkan masalh ini yaitu dengan menunjukan korelasi antara $w(x,y)$ dengan $f(x,y)$.

Pada bentuk yang sederhana korelasi antara dua fungsi ini diberikan oleh persamaan :

$$R(m,n) = \sum_{X}\sum_{Y} f(x,y)w(x-m,y-n)$$

dengan m = 0,1, . . . , M-1 , n=0,1, . . . , N-1 dan penjumlahan dilakukan pada semua dareh w(x,y) yang telah ditentukan seperti terlihat pada gambar 2.13



Gambar 2.13 Deskrepsi Kesamaan

Untuk beberapa posisi dari $w(x,y)$ di dalam $f(x,y)$,akan didapatkan nilai R yang bervariasi. Nilai maksimal dari R menunjukkan posisi dimana $w(x,y)$ adalah paling match dengan $f(x,y)$.

Korelasi yang didefinisikan di atas adalah untuk sub image bukan untuk mencari karakteristik yang khusus, misalnya deteksi garis, tepi dan lain-lain. Untuk korelasi yang lebih lengkap yang dapat menggambarkan karakteristik dari suatu citra didefinisikan dengan persamaan :

$$R(m,n) = \frac{\sum_{X}\sum_{Y} f(x,y)w(x-m,y-n)}{\left[\sum\sum f^{2}(x,y)\right]1/2}$$

faktor normalisasi $\sum\sum f(x,y)$ dimaksudkan untuk memperjelas tepi dari $R(m,n)$.

2.3.4 Klasifikasi atau Pengelompokan.

Pada tahap ini gambar-gambar yang diperoleh dari tahap sebelumnya akan diproses untuk dikenali. Dengan berdasar pada syarat-syarat yang ada dan informasi yang ada pada gambar, maka gambar tersebut akan diklasifikasikan untuk selanjutnya pola dari gambar tersebut dapat dikenali.

Dalam tugas akhir ini metode untuk pengenlannya yaitu dengan menggunakan metode skeleton fitting. Konsep dari skeleton fitting yaitu jika dua simbol atau karakter dikatakan sama manakala dua simbol tersebut memiliki kerangka yang

sama atara satu dengan yang lainnya. Sebagai contoh dua macam citra digital dengan dua macam gray level yaitu 0 dan 1. Citra input tersbut memiliki kerangka F(i,j) dan kerangka citra model yang ada G(i,j) dengan ukuran lebar L dan Tinggi T, maka dengan menghitung :

$$\in = \sum_{j=1}^{T} (\sum_{i=1}^{L} (F(i,j) * G(i,j)))$$

Akan dicari model yang paling menyerupai dengan input, yaitu dengan mencari harga e yang paling maksimal.

2.4 Optical Charakter Recognition (OCR).

2.4.1 Pengertian Umum.

Optical character Recognition (OCR) adalah salah satu alat yang merupakan aplikasi dari ilmu Pengenalan Pola. Kehadiran OCR sangat efisien digunakan dalam pemasukan data secara langsung kedalam komputer dengan cara menangkap informasi dari lembaran data, buku atau hasil cetakan yang lain.

Aplikasi OCR ini banyak sekali, misalnya sebagai pembaca teks dan data secara otomatis, alat komunikasi antar manusia dengan komputer, pengolah bahasa dan mesin penerjemah. Selain itu kehadiran mesin OCR sanangat membantu untuk menangani data-data yang sangat besar, misalnya digunakan pada kantor perbankan, kantor perpajakan dan bidang keuangan yang lain yang menangani berjuta-juta pembayaran dan pemeriksaan tiap tahunnya. Juga digunakan dalam perusahaan asuransi, kartu kredit, bidang kesehatan dan lain-

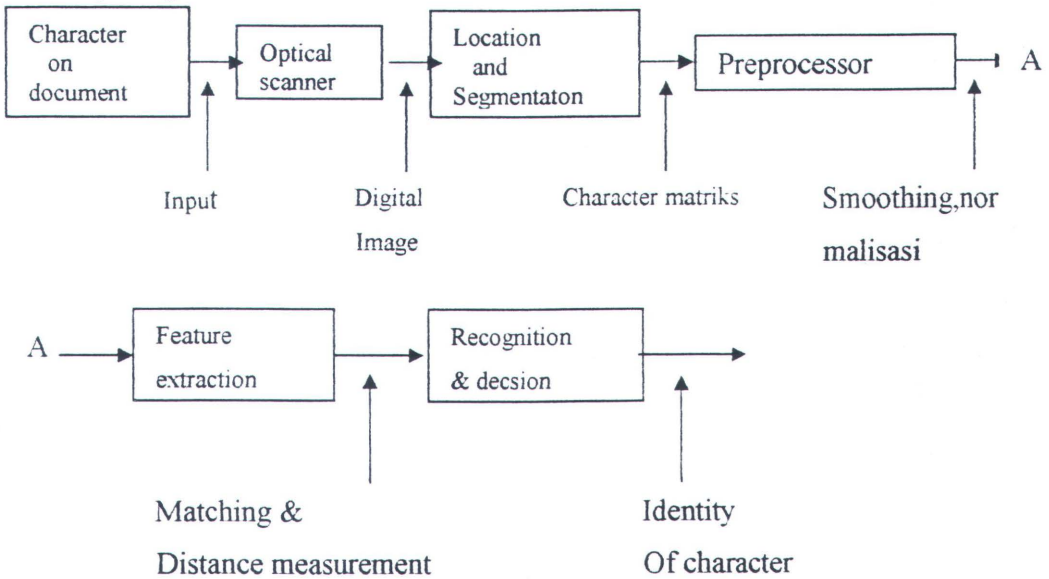
lain, dimana biasanya data diproses (entry) secara manual yang tentunya mempunyai kendala-kendala lambat dan menjemukan. Kehadiran OCR diharapkan dapat menekan kendala-kendala tersebut, sehingga diperoleh waktu dan biaya yang efisien.[5]

2.4.2 Bagian-bagian dari OCR

Pada kenyataannya, inti dari OCR adalah aplikasi dari bidang ilmu Pengolahan Citra Digital dan Pengalan Pola. Secara global, fungsi-fungsi dan operasi yang ada dalam mesin OCR adalah sebagai berikut :

- Digitization, preprocessing dan smoothing.
- Standarization dan feature extraction.
- Standarization dan feature extraction.
- Klasifikasi dari karakter.

Atau dapat digambarkan dalam gambar diagram blok dari system OCR seperti terlihat pada gambar 2.14.



[5]Ching Y Suen,

Gambar 2.14

System OCR

2.4.2.1 Digitisasi, Preprocessing dan Smoothing.

Pada bagian input, karakter pada lembaran data dibaca oleh scanner yang menghasilkan citra digital. Karana gambar yang didapat terdiri dari bermacam-macam gray level, maka secara umum akan dirubah menjadi dua gray level saja, yaitu hitam dan putih.

Setelah tahap ini, tahap berikutnya adalah tahap preprocessing, yaitu untuk memperbaiki kualitas gambar, menghilangkan noise dan memperhalus gambar (smoothing). Algoritma smoothing kebanyakan didasarkan pada teknik menggerakkan window (misal matrik 3 x 3) digerakkan sepanjang matrik dari citra input.

Pada dasarnya proses smoothing terdiri dari dua fungsi yaiu mengisi piksel yang kosong dan menghilangkan piksel yang tidak perlu (filling dan thinning) yang bertujuan untuk mengilangkan noise, patahan-patahan, benjolan atau untuk merubah karakter menjadi bentuk dasar (penulangan). Operasi-operasi yang ada seperti ditunjukkan dalam table II

2.4.2.2 Standarisasi dan Pengambilan Ciri Khusus.

Standarisasi atau normalisasi digunakan untuk menghasilkan pola yang sama (uniform) yaitu sama dalam ukuran, lebar garis dan arahnya.

Feature extraction atau pengambilan ciri-ciri khusus adalah hal yang sangat penting dalam sistem pengenalan pola. Dalam aplikasi OCR, pengelompokan yang baik dari ciri-ciri karakter akan mempermudah system untuk membedakan secara benar atau kelompok karakter dengan yang lain.

Karena karakter-karakter mempunyai bentuk yang bermacam-macam, maka harus dicari bentuk atau ciri-ciri yang khusus yang dapat membedakan antara satu karakter dengan karakter yang lain.

Secara global feature (ciri khusus) dapat kelompokkan menjadi dua kelompok, yaitu Global Analysis dan Structure Analysis, dimana kedua kelompok ini debentuk oleh enam jenis feature, seperti terlihat dalam tabel I.

Dari tabel tersebut dapat dilihat bahwa feature structural dapat menggambarkan topologi dari karakter lebih mudah dan akurat, sehingga cara ini (structure analysis) lebih toleransi terhadap deformasi dari gambar dibandingkan dengan feature global. Di lain pihak feature global dapat dideteksi lebih mudah dan lebih tahan terhadap noise dan sedikit penyimpanan-penyimpanan lokal daripada feature structure. Sehingga kombinasi yang tepat antara 2 type feature di atas dapat menghasilkan pengenalan karakter sangat bagus.

2.5 File Citra

Berdasarkan pembuatannya dan penyimpanannya, file citra (grafik) dapat dibedakan menjadi dua kelompok dasar, yaitu file yang disimpan menggunakan format vektor dan file yang disimpan dalam format bitmap.

2.5.1 File Vektor

File vektor merupakan file gambar yang cara penyimpanannya menggunakan aturan-aturann tertentu. Sehingga ketika file tersebut akan ditampilkan kembali maka program harus mengerti tentang aturan-aturan tersebut, agar gambar atau citra tersebut dapat ditampilkan sesuai dengan ketika waktu menyimpan.

Dengan cara ini maka tidak semua bagian dari gambar atau citra akan disimpan, tetapi hanya bagian-bagian tertentu saja, sehingga file jenis ini tentunya akan menghemat memori. Tetapi masalah yang dihadapi adalah kerumitan dalam pemrogramannya, dan juga akan memakan waktu relatif lebih lama dalam pengolahannya.

2.5.2 File Bitmap

Gambar yang terdapat pada bffer adaptor video pada dasarnya adalah merupakan gambar dalam format bitmap. Dengan menyimpan isi dari pada buffer tersebut ke disk maka file yang dihasilkan tersebut merupakan gile bitmap. Untuk menampilkan kembali gambar dari disk ke layar cukup dengan meletakkan kembali file yang terlah disimpan tersebut ke buffer adaptor video.

Meskipun file yang dihasilkan akan berukuran cukup besar, tetapi file jenis ini relatif lebih mudah dalam pemrogramannya.

Tabel I

Macam-macam Feature dari Matrik karakter[6]

Group	Kelompok Feature	Features	Metode Extract
Global Analysis	Distribusi dari Piksel	a. Posisi dari piksel dan jaraknya dengan titik refere	Menempatkan piksel-piksel dalam koord. x & y dan menyimpan dalam daerah tertentu
		b. Menghitung pergeseran	Menghitung pergeseran sudut tertentu, misalnya 0,45,90 & 135
		c. Kerapatan Matriks	Menghitung piksel-piksel pada yang berbeda
	Transformasi	Series,spectra vectors	Fourier,Haar,Hadamard, Walsh dan lain-lain.
	Pengukuran Fisik	Lebar,tinggi,panjang dari macam-macam cabang dan segemen-segmen garis	Menyimpan bari/kolom yang pertama&terakhir dari matriks karakter dan mengukur panjang dari garis-garis
Structure Analysis	Garis dan Tepi	Lurus dan Miringnya garis, panjang garis dan garis tambahan	Mendeteksi garis/tepi dengan menggerakkan window dalam matriks karakter.
	Outline dari karakte	Arah garis,perpotongan gar	Traking untuk mencari kontur
	Titik tengah dari karakter	Garis diskontinyu,kurva, titik berat.	Thinning,dengan menghilangkan lapisan plaing luar

Tabel II

Smoothing dan Normalisasi dalam Preprocessing

Teknik	Operasi	Tujuan
Smoothing	Filling	Untuk menghilangkan patahan, lobang dan spasi yang kosong
	Thinning	Untuk menghilangkan noise, benjolan dan pixel yang terisolasi. Untuk mengubah karakter menjadi bentuk dasar (tanpa ketebalan) sehingga mudah untuk di extract
	Size	Untuk menyamakan ukuran matrik dari karakter
	Position	Untuk meletakkan matrik karakter pada posisi yang lebih baik, misalnya pada center, kiri-atas
Normalisasi	Skew	Untuk mengarahkan karakter pada posisi atas
	Linewidth	tambahan untuk menghasilkan tebal garis yang sama dalam semua segment garis

[6] Ching Y Suen, Op. Cit. Hal 574-576

BAB III

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

BAB III

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

3.1 Umum

Pada bab ini akan dibahas tentang perancangan perangkat lunak pengenal karakter optik atau alfabet. Pembahasan ditekankan kepada cara kerja perangkat lunak, dalam hal ini berkaitan dengan fungsi-fungsi yang terdapat di dalam program/perangkat lunak itu sendiri.

3.2 Batasan karakter

Sistem pengenalan karakter yang dirancang memerlukan batasan-batasan tertentu terhadap karakter-karakter atau simbol-simbol yang hendak dikenali. Batasan-batasan itu berkaitan dengan jenis dan bentuk karakter atau simbol itu sendiri. Mengenai bentuk karakter, walaupun bentuk karakter alfabetnya tidak baku harus memiliki unsur kemiripan dengan bentuk bakunya.

Jenis karakter atau simbol yang bisa dikenali hanya karakter alfabet dari A sampai Z dan nomor dari 0 sampai 9.

3.3 Kebutuhan Sistem

Sebagaimana telah disebutkan sebelumnya, perangkat lunak ini membutuhkan beberapa modul atau perangkat lunak pembantu. Modul-modul atau perangkat lunak ini bisa ditemukan dalam instalasi untuk sistem operasi

komputer berbasis Linux baik sebagai server maupun sebagai client. Untuk sistem operasi berbasis Linux sebagai servernya dibutuhkan beberapa modul atau perangkat lunak tambahan seperti C atau GCC compiler, modul pemrograman berbasis X Window, dan library Xlib/X11.

Kebutuhan sistem dapat dibedakan menjadi dua bagian yaitu kebutuhan perangkat lunak (sistem operasi dan perangkat lunak lain yang dibutuhkan) dan perangkat keras.

3.3.1 Perangkat Keras

Untuk perangkat kerasnya, minimal dibutuhkan 1 PC sebagai front end. Pada percobaan nanti penulis mengaplikasikan perangkat lunak ini pada sebua PC sebagai front end yaitu Intel Pentium Celeron 333 Mhz dengan RAM 128 Mbytes..

3.3.2 Perangkat Lunak

Untuk perangkat Lunak yang digunakan dalam percobaan adalah untuk program pengenal karakter sebagai front end digunakan sistem operasi linux RedHat 7.2

3.4 File-file yang Diperlukan

3.4.1 File Input

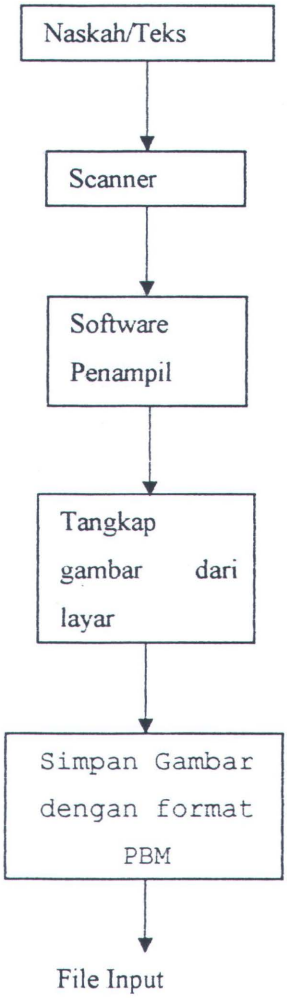
File input dari program ini adalah file citra yang diperoleh dari hasil scanner. Tetapi karena file yang dihasilkan dari scanner ini (yang digunakan

penulis) tidak mendukung pola penyimpanan gambar dalam format PBM (portable bitmap), maka diperlukan sofware tambahan yang dapat mengkonvert file yang dihasilkan dari scanner menjadi file yang berformat PBM yang diperlukan dalam aplikasi ini.

Karena citra input pada program ini hanya terdiri dari gray level hitam dan putih saja atau gray leve 0 dan 1, maka format penyimpanan filenya adalah sebagai berikut :

Baris pertama dari bitmap yang memiliki lebar (w) akan tersimpan dalam array (w/8)+((w%8)!=0). Untuk bit yang sebelah kiri pada masing-masing byte adalah bit yang sangat penting (black atau "on" diwakili dengan 1 dan white atau "off" diwakili dengan 0).

Dimana masing-masing angka akan disimpan dalam bentuk kode ASCII-nya, sehingga tiap angka anka hanya membutuhkan 1 karakter saja (1 byte), misalnya angka 65 akan disimpan dalam bentuk huruf A.



Gambar 3.1 Proses Pembuatan File Input

3.4.2 File Library

Karena perangkat lunak ini berjalan dalam sistem X Window pada linux. , maka diperlukan sekumpulan model yang akan dijadikan acuan dalam pembuatan grafik user interface. Sekumpulan model ini disimpan dalam suatu file yang disebut file Xlib/X11. Di samping itu dibutuhkan juga sekumpulan model yang digunakan untuk proses pengenalan karakter/simbol yang tersimpan dalam file

common.h dan file alphabet.c. Adapun struktur dari tiap item model yang ada dalam kedua file tersebut

a) Structure Alfabet

```
typedef struct {
    char *LN;
    char *ln;
    unsigned char cc;
    unsigned char cbm[16];
    unsigned char sc;
    unsigned char sbm[16];
    unsigned a;
    unsigned m;
    unsigned char l;
    char va;
} alpharec;
extern alpharec latin[];
extern alpharec number[];
extern alpharec *alpha;
extern int latin_sz,
           number_sz,
           max_sz,
           alpha_sz,
           alpha_cols;

extern int l_align[256];
extern int g_align[256];
extern int n_align[256];
#define MAX_ALPHA 100
extern char inv_balpha[MAX_ALPHA];
#define MAXFNL 256
```

Gambar 3.2 Structur Alfabet

- Macro LATIN, NUMBER, IDEOGRAM, digunakan untuk membatasi tipe-tipe karakter/simbol yang akan dikenali.
- Variable LN yang bertipe pointer char digunakan untuk menunjukkan nama karakter/simbol kapital (karakter/simbol besar).
- Variable ln yang bertipe pointer char digunakan untuk menunjukkan nama karakter/simbol lower (karakter/simbol kecil).

- Variable cc yang bertipe unsigned char digunakan untuk menunjukkan kode kapital yang berdasarkan ISO-8859-X.
- Variable cbm[16] yang bertipe unsugned char digunakan untuk menunjukan pemetaan karakter/simbol kapital (8 x 16 bitmap).
- Variable sc yang bertipe unsigned char digunakan untuk kode small karakter/simbol yang berdasarkan ISO-8859-x.
- Variable l yang bertipe unsigned char digunakan untuk pemetaan karakter/simbol latin pada keyboard.

Adapun karakter-karakter/simbol yang telah ada dalam file common.h dan alphabet.c adalah sebagai berikut :

- Huruf kecil Latin dari a sampai dengan z.
- Huruf besar Latin dari A sampai dengan Z

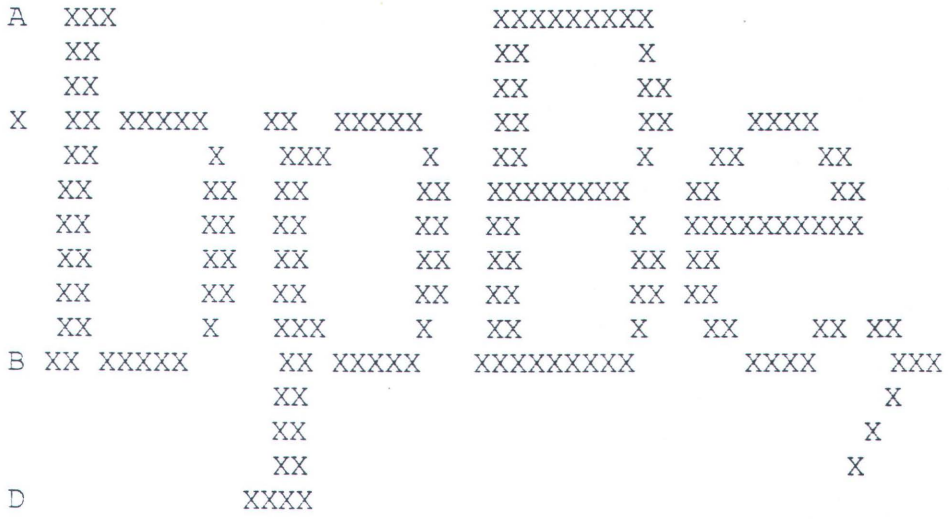
Angka dari 0 sampai dengan 9.

Sedangkan untuk karakter-karakter/simbol-simbol yang lain bisa ditambahkan lagi pada kedua file tersebut.

b) Structure Simbol

Pensejajaran (*Alignment*) simbol secara vertikal sangat penting untuk bermacam-macam heuristik,sebagai contoh, garis vertikal dari karakter "p" dapat dikenali sebagai karakter "l", tetapi dengan menggunakan pensejajaran kita dapat memperbaiki kesalah tersebut

Adapun empat posisi pensejajaran vertikal dilambangkan sebagai (ascent, baseline dan descent), dalam Tugas akhir ini digunakan variable tambahan X untuk menunjukkan tinggi dari huruf kecil tanpa ascenders.



Gambar 3.3 Pensejajaran simbol secara vertikal

A (0) .. ascent (Knuth asc_height)

X (1) .. x_height

B (2) .. baseline

D (3) .. descent (Knuth desc_depth)

Berdasarkan gambar di atas dapat dikatakan bahwa pensejajaran karakter "b" dan karakter "B" dapat diwakili dengan nilai 02, pensejajaran karakter "p"adalah 13, pensejajaran karakter "e" adalah 12, dan pensejajaran simbol koma adalah 23. Sedangkan untuk simbol "."(titik) adalah 23 dan titik dari karakter "i"

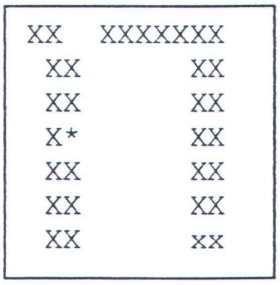
adalah 00 serta semua karakter (huruf) kecil mempunyai tinggi yang sama. Yang mana nilai-nilai tersebut digunakan pada proses klasifikasi simbol atau karakter agar dapat membedakan salah satunya karakter i dengan karakter h, dengan cara memisahkan dot atau titik yang dimiliki oleh karakter i dengan body dari karakter i tersebut.

```
typedef struct  {
    /* fundamental */
    int ncl;        /* jumlah klosur */
    int *cl;        /* daftar klosur*/
   /* data geometric */
    int l,r,t,b;    /* geometric limits */
    int nbp;        /* jumlah piksel hitam*/
   char va; /* pensejajaran vertikal (vertical alignment) */
    char c;          /* kolom text */
    int f;             /*flag simbol*/
    /* neighborhood data */
    int N,S,E,W;     /* simbol-simbol terdekat */
    int sw;  /* daerah disekitar word (kata) dalam dokumen */
    int sl;
    /* transliterations */
    trdesc *tr;      /* transliteration  pertama*/
    char tc;         /* transliteration class */
    char bq;         /* best quality */
    int pb;
    int bm;          /* best match pattern ID */
    int lfa;         /* last pattern ID analisis */
} sdesc;
```

Gambar 3.4 Structur simbol

c) Structure Closure

Seperti dalam gambari dibawah ini,software ini akan menghitung set piksel hitam yang ditandai dengan "X" dan "*", adapun variabel l, r, t, b adalah batas kotak dari closure tersebut . closure-closure yang berpixel hitam yang saling berhubungan adalah point pertama untuk mengidentifikasikan simbol-simbol atau karakter yang ada dalam dokumen.



Gambar 3.5 Penghitungan Closure

Ketika proses loding input dokumen, pragram ini akan menghitung semua closure yang terdapat dalam dokumen tersebut dan meletakkannya pada array. Ketika session file dibuat, closure-closure tersebut diletekkan dalam format/variable CML

```
typedef struct {
char type;                /* komponen graphi */
int  l,r,t,b;  /*{left,right,top,bottom}
coordinate kotak closure */
unsigned char *bm; /* bitmap */
int nbp;                 /* jumlah black pixels */
short seed[2];           /*closure computation */
int *sup;                  /* list symbols */
int supsz;  /* ukuran memory buffer variabel sup */
} cldesc;
cldesc *cl;
extern int topcl,clsz;
extern int *clx,*cly;
```

Gambar 3.6 Structur Closur

d) Structure Pattern

Struktur ini digunakan untuk mencatat pattern-pattern yang dimiliki oleh simbol atau karakter baik pada dokumen maupun pada simbol inputan untuk proses pencocokan. Ketika pattern yang dimiliki oleh simbol atau karakter dikenali pada dokumen, hasil transliterasinya dikirim pada output. Pada umumnya

transliteration adalah panjang dari string 1 (satu) seperti "a","D","," dan lain-lain, akan tetapi dalam Tugas Akhir ini panjang string sampai MFTL sebesar 30 yang digunakan untuk membatasi ukuran dari beberapa string yang lain yang tidak didukung oleh X font pada sistem X-Window pada Linux (untuk pengembangan selanjutnya)

```
typedef struct {
 int id;        /* uniqidentifier */
 char a;        /* alphabet (latin,etc) */
 short pt;      /* pattern type */
short fs;      /* font size (10pt, 12pt, etc) */
 short bw;      /* bitmap width (pixels) */
 short bh;      /* bitmap height (pixels) */
short bb;      /* bitmap baseline (pixels) */
short bp;  /* jumlah black pixel pada bitmap */
short sw;       /* skeleton width (pixels) */
short sh;      /* skeleton height (pixels) */
short sb;      /* skeleton baseline (pixels) */
short sp;/*jumlag black pixels  pada skeleton*/
short sx,sy;       /* ukuran x dan y skeleton */
short bs;
unsigned char *b;  /* bitmap */
unsigned char *s;  /* skeleton */
int f;             /* flags */
int act;           /* act */
int cm;            /* cumulative matches */
short ts;          /* size of the field tr */
short ds;          /* size of the field d */
 char *tr;          /* transliteration */
char *d;          /* name of the originating page */
int e;             /* mc */
float sl;          /* (skew) */
short l,t;  /*   top-left position pada simbol */      short
p[8]; /* parameters untuk skeleton computation */
} pdesc;
```

Gambar 3.7 Pattern

e) Simbol/Karakter

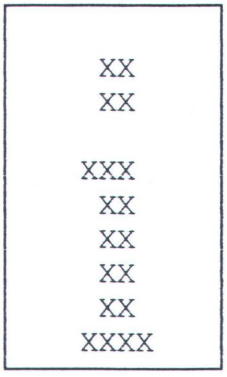
Salah satu karakter pada dokumen mungkin dibentuk dua atau lebih closure, karena satu karekter/simbol yang terbentuk lebih dari dua closure, sehingga untuk

mengenali satu karakter tidak perlu dengan closure-closure tapi bisa langsung dengan satu set closure. Sehingga dalam Tugas Akhir ini konsep "symbol" adalah sekumpulan satu atau lebih closure. Inisialisasi dari Tugas Akhir ini menghasilkan satu kesatuan simbol untuk masing-masing closure, adapun rangkain langkah (step) dalam Tugas Akhir ini akan mendefinisikan simbol-simbol batu yang terbentuk dua atau lebih closure.

Untuk lebih jelasnya terdapat tiga closure yang tidak saling berhubungan pada (atom/piksel) bagian yang membentuk simbol, sebagai contoh : karakter "a" dan "I" dihubungkan oleh satu closure dan karakter "u" dibentuk oleh dua closure.

Pada prinsipnya aplikasi ini tidak berusaha untuk memecah closure-closure tersebut menjadi closure yang lebih kecil lagi, akan tetapi dengan klasifikasi heuristic berusaha untuk membentuk berbagai macam pattern-patern untuk memecahkan simbol-simbol (karakter) seperti karakter "ai". Pada karakter "u", dengan menggunakan klasifikasi heuristik diharapkan dapat menggabungkan dua closure menjadi satu closure dan akan mengaplikasikannya sebagi karakter atau pattern "u" untuk memecahkan masalah tersebut.

Masing-masing simbol diletakkan dalam "sdesc" striktur. Simbol yang telah dibuat atau dihasilkan tidak akan pernah dihapus, tetapi akan diberikan indek pada variable array "mc" (pada sdesc). Closure dan simbol telah dinomeri pada dokumen yang bersangkutanNote that closures and symbols are. Jika satu set closure yang mendefinisikan satu simbol tidak akan berubah, maka batas kotak simbol dan total jumlah piksel hitam juga tidak akan berubah. Sehingga dua masukan yang berbeda pada mc array tidak akan mempunyai closure yang sama.



Gambar 3.9 Karakter i

f) Identifikasi Simbol

satu closure yang sama mungkin digunakan oleh dua atau lebih simbol yang lain. Seperti hanya klosure sebelah kiri dari huruf "u" dapat didefinisikan sebagai bada (body) dari huruf "I". Dalam hal ini meskipun tidak ditemukannya titik pada closure sebelah kiri dari huruf "u" seperti huruf atau karakter "I". Untuk memecahkan masalah tersebut dicari closure yang terdekat dari closure yang telah ditemukan atau dicocokkan dalam hal ini closure sebelah kanan dari huruf atau karakter "u" dan selanjutnya digunakan untuk mencocokkannya dengan beberapa pattern dari font atau huruf yang lainnya.

g) Font Size

Ukuran font digunakan untuk mengklasifikasikan semua semua simbol atau karakter yang diinputkan. dalam Tugas Akhir ini ukuran suatu font (huruf) dinyatakan dalam besaran "point" sebagai contoh : jika sati inch (1 inch) sama dengan 72,27 point pada printer dengan menggunakan kedalam piksel 600 dpi,

maka masing-masing pt atau satu pt sama dengan 600,72,27 = 8,3 piksel. Untuk 10 point karakter roman (Roman Font) maka tinggi karakter atau huruf kecil roman tersebut adalah sama dengan 155/36 pt. Dalam Tugas Akhir ini untuk menghitung besarnya ukuran font (huruf) kecil dari salah satu karakter atau simbol dengan menggunakan ketinggian dalam piksel/height in pixel (h) adalah sebagai berikut :

$$F = 10 * h / 35.7$$

3.4.3 Procedure-procedure (Algoritma)

a) Funsi Load File Image

Pada procecure ini digunakan pertama kali untuk membaca file inputan, membaca patern, dan menghitung skeleton sebagai persiapan dalam melakukan proses pengenalan. Sebagaimana yang tertulis dalam fungsi dibawah ini bahwa ketika user memilih load file maka fungsi ini akan memanggil fungsi setview (PAGE_LIST) yang digunakan untuk menampilkan file data yang akan diproses untuk pengenalan:

```
int step_2(int reset)
{
  if (reset) {
     setview(PAGE_LIST);
     st = 0;
     myreset = 1;
 } if (st == 0) {
      if (act == NULL)
          recover_acts(acts);
      }else if (st == 1) {
      if (pattern == NULL) {
          recover_patterns(patterns);
          st = 2;
```

```
k = 0;
      } else
     } else if (st == 2) {
       if (k<topp) {
           if (k % DPROG == 0)
      show_hint(0,"Penghitungan                    kerangka/skeleton
%d/%d",k,topp);
            pskel(k++);
            myreset = 0;
        } else {
            st = 3;
            myreset = 1;
        }
        r = 1;
    }else if(st==3){r = load_page(to_ocr,myreset);
   if (r == 0) { redraw_dw = 1; }else myreset = 0;
    } return (1)
}
```

Gambar 3.10 Fungsi Load File

b) Fungsi Membaca File Image

Procedure ini berfungsi untuk membaca file citra yang bertipe PBM dan mengkonvert closure file citra tersebut ke dalam bitmap yang akan diproses ke dalam layar monitor. Karena file citra disimpan dalam format PBM, maka sebelum ditampilkan di layar monitor datanya harus diproses terlebih dahulu. Adapun fungsi tersebut adalah sebagai berikut :

```
int pbm2bm(char *f,int reset)
{   static FILE *F;
    F = zfopen(f,"r",&pio);
    if ((fgetc(F) != 'P') || (fgetc(F) != '4')) {
            fatal(DI,"%s Bukan Format PBM File",f);
        }
       for (n=rc=0, w=h=0; (n<4) && ((c=fgetc(F)) != EOF); ) {
           if ((c == '#') || ((rc==1) && (c!='\n'))) {
            rc = 1;
} else if ((c!=' ') && (c!='\t') && (c!='\r') && (c!='\n')) {
   if ((c < '0') || ('9' < c)) {
            fatal(DI,"%s Bukan Format PBM File",f);}
               if (n <= 1) {
```

```
                    n = 1;
                    w = w * 10 + c - '0';
                }else {
                    n = 3;
                    h = h * 10 + c - '0';
                }
            } else {
                if (n == 1)
                    n = 2;
                else if (n == 3)
                    n = 4;
              rc = 0;}
}
```

Gambar 3.11

Proses pembacaan file image yang bertipe pbm

Proses diatas dimulai dengan mencoba membuka file pbm atau membaca file pbm, kemudian akan dicheck apakah file tersebut sesuai dengan aturan format file pbm dengan mengecek *megic number* dari file pbm yaitu P dan 4 serta aturan-aturan pbm file lainnya. Jika dalam file inputan tersebut tidak ditemukan P dan 4 serta aturan pbm file seperti mengecek karakter-karakter whitespace, maka akan ditampilkan pesan error bahwa file tersebut bukan file pbm.

Setelah proses pembacaan file inputan yang bertipe pbm dapat terbaca, maka selanjutnya membaca line dan menghitung jumlah closure yang ada dalam file tersebut. Adapun subprogram yang menghitung closure tersebut adalah sebagai berikut :

```
unsigned char *q,m;
    int i,j,la,k;
    if (n < h) {
        if (fread(s,1,bl,F) < bl) {
           fatal(IO,"error Baca  file \"%s\"",f);
        }
     } else
       memset(s,0,bl);
        for (i=j=0, la=-1; i<abs; ++i) {
            if (ab[i].x < 0) {
                if (la < 0)
                    la = j = i;
                else {
                    ab[j].y = i;
                    ab[i].y = -1;
                    j = i;
                }
            }
        }
        for (q=s, m=128, i=1; i<=w; ++i) {
            kl[i] = -1;
            if ((*q & m) == m) {
                join(kl+i,kl+i-1,ab);
                join(kl+i,ll+i-1,ab);
          join(kl+i,ll+i,ab);
          join(kl+i,ll+i+1,ab);
        if (kl[i] < 0) {
           if (la < 0){
 ab=c_realloc(ab,(abs+AB_INCR)*sizeof(bdesc),"pbm2bm");
   for (j=0; j<AB_INCR; ++j) {
         ab[abs+j].x = -1;
        ab[abs+j].y=(j==AB_INCR-1) ? -1 : bs+j+1;
   ab[abs+j].b = NULL;
   ab[abs+j].s = 0;
}
la = abs;
abs += AB_INCR;
}
j = la;
la = ab[la].y;
kl[i] = j;
ab[j].x = i-1;
ab[j].y = n;
ab[j].w = 0;
ab[j].h = 0;
if (ab[j].b == NULL) {
   ab[j].b = c_realloc(NULL,ab[j].s=FS*FS/(4*8),"pbm2bm");
}
memset(ab[j].b,0,ab[j].s);
}
}
```

```
if (m == 1) {
    m = 128;
    ++q;
}
else
   m >>= 1;
}
for (i=1; i<=w; ++i) {
     while (((j=ll[i]) >= 0) && (ab[j].x<0))
            ll[i] = ab[j].y;
     while (((j=kl[i]) >= 0) && (ab[j].x<0))
           kl[i] = ab[j].y;
}
for (i=1; i<=w; ++i) {
    if (((j=ll[i]) >= 0) && (ab[j].x>=0) && (ab[j].y>=0)) {
         ab[j].y = -ab[j].y-1;
    }
}
for (i=1; i<=w; ++i) {
     if (((j=kl[i]) >= 0) && (ab[j].y<0)) {
          ab[j].y = -ab[j].y-1;
     }
}
for (i=1; i<=w; ++i) {
    if (((j=ll[i]) >= 0) && (ab[j].x>=0) && (ab[j].y<0)) {
          ab[j].y = -ab[j].y - 1;
          new_cl(ab[j].x,ab[j].y,ab[j].w,ab[j].h,ab[j].b);
          ab[j].x = -1;
          ab[j].y = 0;
    }
}
memcpy(ll,kl,(w+2)*sizeof(int));
for (i=1; i<=w; ++i) {
    if ((k=kl[i]) >= 0) {
       unsigned char *p,m;
       int l;
       for (j=w; (j>i) && (kl[j]!=k); --j);
           if (i-1 < ab[k].x)
              if ((j-1)-ab[k].x+1 > ab[k].w)
                  extend(ab+k,i-1,ab[k].y,(j-1)-(i-
1)+1,ab[k].h+1);
              else
                  extend(ab+k,i-1,ab[k].y,ab[k].w+ab[k].x-(i-
1),ab[k].h+1)
              else if ((j-1)-ab[k].x+1 > ab[k].w)
                      extend(ab+k,ab[k].x,ab[k].y,(j-1)-
ab[k].x+1,ab[k].h+1);
              else
extend(ab+k,ab[k].x,ab[k].y,ab[k].w,ab[k].h+1);
                p = ab[k].b + (n-ab[k].y)*byteat(ab[k].w) + (i-
1-ab[k].x)/8;
                m = ((unsigned) 1) << (7-((i-1-ab[k].x)%8));
```

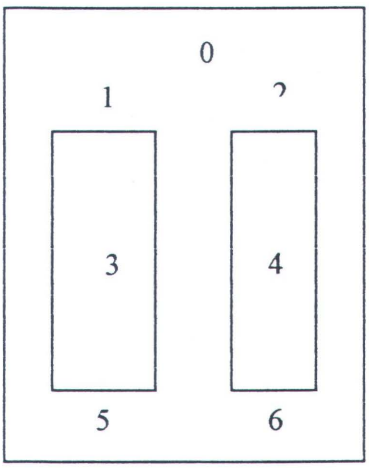
```
for (l=i; l <= j; ++l) {
                    if (kl[l] == k) {
                        *p |= m;
                        kl[l] = -1;
                    }
                    if (m == 1) {
                        m = 128;
                        ++p;
                    }
                    else
                        m >>= 1;
                }
            }
        }
    }
    if (++n > h) {

        c_free(s);
        c_free(ll);
        c_free(kl);
        if (ab != NULL) {
            while (--abs >= 0)
                  if (ab[abs].b != NULL)
                     free (ab[abs].b);
                     c_free(ab);
        }
        snprintf(mba,MMB,"Pembacaan file selesai %s",pagename);
        show_hint(1,mba);
        zfclose(F,pio);
        return(0);
    }
    return(1);
}
```

Gambar 3.12 Penghitungan Closur pada file PBM

c) Deteksi Batas Simbol (Block Detection)

Pada field "c" untuk masing-masing symbol akan manyimapan block yang didalamnya terdapat simbol tersebut. Symbol-simbol yang berada di luar block ditandai dengan nilain -1(unknown) atau -2 (isolated noise). Adapun contoh block tersebut adalah sebagai berikut :



Gambar 3.13 Ilustrasi Block dokumen

Untuk blocks 3 dan 4 adalah kolom text dalam sebuah document. blocks 1 dan 2 adalah headings dari kolom text columns, block 5 dan 6 adalah footings dan block 0 adalah judul (title).

```
void detect_blocks(void)
{
    int i;
    /* inisialisasi block data pada semua symbols */
    for (i=0; i<=tops; ++i) {
        mc[i].c = -1;
        C_UNSET(mc[i].f,F_SDIM);
    }
    /* menandai semua ukuran symbols/karakter pada document */
    for (i=0; i<=tops; ++i) {
    }
    /* mendeteksi margins dan ignore marginal symbols */
    if (TC == 2) {
    }
    else if (TC == -2) {
    }
}
```

Gambar 3.14

Fungsi Block Detection

d) Klasifikasi Simbol/Karakter

Dalam tahap ini simbol atau karakater yang ada dalam dokumen akan dikalsifikasikan dengan simbol atau karakter yang telah inputkan. Adapun segala informasi dari simbol-simbol atau karakter tersebut telah tersimpan dalam buffer mc.

```
int step_7(int reset)
{    static int k;
    /* dimulai pada simbol pertama*/
    if (reset) {
        k = (justone) ? curr_mc : 0;
        selbc(-1);
        return(1);
    }
     if (selbc(k) == 0) {
         if ((!C_ISSET(mc[k].f,F_SS)) &&
            (mc[k].ncl == 1) &&
            (bf_auto) &&
            ((mc[k].tr   ==   NULL)   ||   (!C_ISSET(mc[k].tr-
>f,F_BAD))) &&
            (sdim(k))) {
            /*baca alpabet pada kata(word) */
            update_pattern(k,NULL,-1,UNDEF,-1,-1,0);
            C_SET(mc[k].f,F_SS);
            mc[k].bm = pattern[topp].id;
            pattern[topp].cm = 1;
        }
        /*next step */
        if ((justone == 1) || (++k > tops)) {
            k = -1;
            return(0);
        }
        /* next symbol */
        else {
            selbc(-1);
        } }
      return(1);}
```

Gambar 3.15 Kalsifikasi Simbol

e) Skeleton

Pada fungsi ini digunakan untuk menghitung pixel kerangka(*skeleton)* image yang berformat PBM yang telah diconvert kedalam format bitmap dan diletakkan dalam variable cb. Skeleton pixel yang telah dicetak atau ditandai dengan warna hitam dan yang lainnya ditandai dengan warna gray.

Dan fungsi ini juga menghitung koordinat batas kiri, batas kanan, batas atas dan batas bawah dari skeleton pixel yang mana dalam aplikasi ini masing-masing koordinat tersebut dinyatakan dengan variable fc_bp, lc_bp, fl_bp, ll_bp.

Untuk algoritma 0,1 dan 2, jika (i0,j0) adalah koordinat pixel yang valid dari skeleton pixel yang terletak pada variable cb, maka pixel tersebut akan digunakan hanya untuk menghitung pixel-pixel skeleton yang berada pada daerah (i0,j0). Jika i0==-2 maka hitung skeleton pixel yang dimiliki oleh simbol j0 dengan asumsi bahwa variable cb disejajarkan pada batas atas (top) dan batas kiri dari simbol j0. Untuk lebih jelasnya seperti pada procedure skeleton dibawah ini :

```
void skel(int i0,int j0)
{
    char cb3[FS*LFS];
    int i,j,x,y;
  /* limit untuk black pixels */
    fc_bp = FS;
    lc_bp = -1;
    fl_bp = FS;
    ll_bp = -1;
 if (SA == 6) {
        int d;
        for (i=0; i<LFS; ++i) {
            for (j=0; j<FS; ++j) {
                if (cb[i+j*LFS] == BLACK)
                    cb3[i+j*LFS] = GRAY;
                else
                    cb3[i+j*LFS] = WHITE;
            }
        }
```

```
    cb_border();
     for (i=d=0; i<FS; ++i) {
          for (j=0; j<FS; ++j) {
               if (cb[i+j*LFS] == GRAY) {
                    int ia,ib,ja,jb;
                    ia = i-1;
                    ib = i+1;
                    ja = j-1;
                    jb = j+1;
                    if(((i==0)||(cb[ia+j*LFS]!=BLACK))&&
((ib==FS) || (cb[ib+j*LFS] != BLACK)) &&
                                   ((j==0)   ||   (cb[i+ja*LFS]   !=
BLACK)) &&
                                   ((jb==FS)   ||   (cb[i+jb*LFS]   !=
BLACK)) &&
                                   ((i==0)      ||      (j==0)      ||
(cb[ia+ja*LFS] != BLACK)) &&
                                   ((i==0)      ||      (jb==FS)      ||
(cb[ia+jb*LFS] != BLACK)) &&
                                   ((ib==FS)      ||      (ja==0)      ||
(cb[ib+ja*LFS] != BLACK)) &&
                                   ((ib==FS)      ||      (jb==FS)      ||
(cb[ib+jb*LFS] != BLACK))) {

                              cb3[i+j*LFS] = BLACK;
                              }
                              cb[i+j*LFS] = WHITE;
                         }
                    else if (cb[i+j*LFS] == BLACK)          {
                              d = 1;
                         }
                   }
              }


    }
memcpy(cb,cb3,FS*LFS);
for (y=0; y<FS; ++y) {
char *p;
p = cb + y*LFS;
for (x=0; x<FS; ++x,++p) {
    if (*p == BLACK) {
       if (fc_bp >= FS)
            fc_bp = x;
            lc_bp = x;
            if (fl_bp >= FS)
               fl_bp = y;
               ll_bp = y;
            }
        }
      }
    return;
    }
```

```
/*Hitung distance dari masing-masing pixel*/
    if (SA == 5) {
        int f,t;
        cb_border();
/*Skeleton pixel diberi nilai 0,sedangkan yang lain bitpmap
pixels diberi nilai-1 dan untuk pixel putih ditandai daengan
nilai -2.*/
        for (i=0; i<FS; ++i) {
            for (j=0; j<FS; ++j) {
                if (cb[i+j*LFS] == GRAY)
                    cb[i+j*LFS] = 0;
                else if (cb[i+j*LFS] == BLACK)
                    cb[i+j*LFS] = -1;
                else
                    cb[i+j*LFS] = -2;
            }
        }
for (t=0, f=1; f; ++t) {
            f = 0;
            for (i=0; i<FS; ++i) {
                for (j=0; j<FS; ++j) {
                   if (cb[i+j*LFS] == -1) {
                        if(((i>0)&&(cb[(i-
1)+j*LFS]==t))||((i+1<LFS)&&(cb[(i+1)+j*LFS]==t))||((j>0)  &&
(cb[i+(j-1)*LFS]==t))||((j+1<LFS) && (cb[i+(j+1)*LFS]==t)))
{
     cb[i+j*LFS] = t+1;
     f = 1;
 }
 }
 }
 }
memcpy(cb2,cb,LFS*FS);
     for (i=0; i<FS; ++i) {
          for (j=0; j<FS; ++j) {
            int a,pa,ua,b,pb,ub,m,n,t=3,v;
            if ((m = v = cb2[i+j*LFS]) >= 0) {
/* hitung batas disekitar kotak (i,j).*/
               if ((pa = i-t) < 0)
                    pa = 0;
                if ((ua = i+t) >= FS)
                    ua = FS-1;
                if ((pb = j-t) < 0)
                    pb = 0;
                if ((ub = j+t) >= FS)
                    ub = FS-1;
for (a = pa; a <= ua; ++a) {
              for (b = pb; b <= ub; ++b) {
                 if ((n=cb2[a+b*LFS]) > m)
                     m = n;
                  }}
```

```
/*Tandai(i,j) sebagai skeleton pixel jika distancenya cukup
besar/maximum*/
            if (((v <= BT) && (m == v)) ||
               ((v > BT) && (v+1 >= m)) ||
               (v >= RX))
               cb[i+j*LFS] = -1;
                }
            }
        }
 /* Tandai skeleton pixels dengan warna hitam*/
       for (j=0; j<FS; ++j) {
           for (i=0; i<FS; ++i) {
              if (cb[i+j*LFS] == -2)
                 cb[i+j*LFS] = WHITE;
               else if (cb[i+j*LFS] == -1)
                    cb[i+j*LFS] = BLACK;
                else
                    cb[i+j*LFS] = GRAY;
            }
        }
/* hitung batas-batas pixel*/
        for (y=0; y<FS; ++y) {
            char *p;

            p = cb + y*LFS;
            for (x=0; x<FS; ++x,++p) {
                if (*p == BLACK) {
                    if (fc_bp >= FS)
                        fc_bp = x;
                    lc_bp = x;
                    if (fl_bp >= FS)
                        fl_bp = y;
                    ll_bp = y;
                }
            }
        }

        return;
    }
if (SA == 4) {
/* results, centers and distances */
   int R[16],C[16],D[16];

int dx1[16] = { 1, 1, 1, 0, 0, 0,-1,-1,-1,-1,-1, 0, 0, 0,
1, 1};
   int dy1[16] = { 0, 0,-1,-1,-1,-1,-1, 0, 0, 0,
1, 1, 1, 1, 1, 0};
   int dx2[16] = { 1, 1, 1, 1, 0,-1,-1,-1,-1,-1,-1,-1, 0,
1, 1, 1};
   int dy2[16] = { 0,-1,-1,-1,-1,-1,-1,-1, 0, 1, 1, 1, 1,
1, 1, 1};
  int c,c1,c2,f,g,n,r,s,u,v;
```

```
/* original bitmap */
        memcpy(cb3,cb,LFS*FS);
        /* test masing-masing pixel */
        for (i=0; i<LFS; ++i) {
            for (j=0; j<FS; ++j) {
              /* ignore non-BLACK pixels */
                if (cb3[i+j*LFS] != BLACK)
                    continue;
                /* draw line */
             for (f=r=0; r<16; ++r) {
                   /*RX steps */
             for (g=1, s=0, u=i, v=j; g && (s<RX); ++s) {
                       /* next step */
             if (s & 1) {
                 u += dx2[r];
                 v += dy2[r];
             }
             else {
                 u += dx1[r];
                 v += dy1[r];
               }
               if ((u < 0) || (LFS <= u) ||
                 (v < 0) || (FS <= v) ||
                  cb3[u+v*LFS] != BLACK) {
                            g = 0;
                            ++f;
                        }
                    }
R[r] = g;
D[r] = s-1;
}
if (f == 0)
    continue;
if (f == 16) {
    cb[i+j*LFS] = GRAY;
    continue;
}
/* inisialisasi titik tengah*/
   for (r=0; r<16; ++r)
        C[r] = 0;
       for (c1=c2=-1, n=0, r=15, g=-1; r>g; --r)          {
   if (R[r])
      continue;
 for (u=r; R[s=(u==15)?0:u+1] == 0; u=s);
for (v=r; (v>0) && (R[v-1] == 0); --v);
     /* pilih daearah tengah*/
   if (r <= u) {
      c = (u+v) / 2;
      C[c] = u-v+1;
   }
```

```
else {
   g = u;
        u += 16;
        c = (u+v) / 2;
        if (c >= 16)
            c -= 16;
            C[c] = u-v+1;
         }
if (++n == 1)

            c1 = c;
         else if (n == 2)
            c2 = c;
                    /*next search */
                    r = v-2;
                }
if (n == 1) {
          if (C[c1] > 3)
              cb[i+j*LFS] = GRAY;
          }
          else if (n == 2) {
                  /* width 2 */
              if ((D[c1] == 1) && (D[c2] == 0))
                   cb[i+j*LFS] = GRAY;
              }
              else {
                    cb[i+j*LFS] = GRAY;
                }
            }
        }
        /* compute limits */
        for (y=0; y<FS; ++y) {
            char *p;
            p = cb + y*LFS;
            for (x=0; x<FS; ++x,++p) {
                if (*p == BLACK) {
                    if (fc_bp >= FS)
                        fc_bp = x;
                    lc_bp = x;
                    if (fl_bp >= FS)
                        fl_bp = y;
                    ll_bp = y;
                }
            }
        }
        return;
    }
```

```
    if (SA == 3) {
        int n;
        char *p;
        memcpy(cb3,cb,LFS*FS);
        for (n=0; n<BT; ++n) {
            cb_border();
            for (i=0; i<LFS; ++i)
                for (j=0; j<FS; ++j)
                    if (cb[i+j*LFS] == GRAY)
                        cb[i+j*LFS] = WHITE;
        }
        /* warnai non-skeleton pixels dg gray */
        for (i=0; i<LFS; ++i)
            for (j=0; j<FS; ++j)
                if ((cb3[i+j*LFS] == BLACK) && (cb[i+j*LFS] ==
WHITE))
                    cb[i+j*LFS] = GRAY;
        /* compute limits */
        for (y=0; y<FS; ++y) {
            p = cb + y*LFS;
            for (x=0; x<FS; ++x,++p) {
                if (*p == BLACK) {
                    if (fc_bp >= FS)
                        fc_bp = x;
                    lc_bp = x;
                    if (fl_bp >= FS)
                        fl_bp = y;
                    ll_bp = y;
                }
            }
        }
        return;
    }
/*1. Hitung batas klosure yang ditunjukan pada (i0,j0)*/
    if ((0<=i0) && (i0<LFS) && (0<=j0) && (j0<FS)) {
        memcpy(cb2,cb,LFS*FS);
        border(i0,j0);
    }
/*2.Hitung batas klosur simbol pada ke j0*/
    else if (i0 == -2) {
        sdesc *m;
        cldesc *c;
        int n;
        memcpy(cb2,cb,LFS*FS);
        m = mc + j0;
        for (n=0; n < m->ncl; ++n) {
            c = cl + m->cl[n];
            border(c->seed[0]-c->l,c->seed[1]-c->t);
        }
    }
```

```
 if((ma=fabs(ro(xbp[i],ybp[i])-ro(xbp[j],ybp[j]))) > PI)
                            ma = 2*PI - ma;
                        if (ma < MA)
                            continue;
                    }

                    if ((debug) &&

(joined(x+xbp[i],y+ybp[i],x+xbp[j],y+ybp[j]) !=

joined(x+xbp[j],y+ybp[j],x+xbp[i],y+ybp[i]))) {

                        db("assymetric behaviour of joined");
                    }


((joined(x+xbp[i],y+ybp[i],x+xbp[j],y+ybp[j]) != 0) ||
 (joined(x+xbp[j],y+ybp[j],x+xbp[i],y+ybp[i]) != 0))
                        ma = 0;
                }
            }
        }
        if (ma >= MA)
            cb2[x+y*LFS] = BLACK;
    }
    for (y=0; y<FS; ++y) {
        for (x=0; x<LFS; ++x) {
            if (cb2[x+y*LFS] == BLACK) {
                /* skeleton pixel */
                cb[x+y*LFS] = BLACK;
                if (x < FS) {
                    if (x < fc_bp)
                        fc_bp = x;
                    if (x > lc_bp)
                        lc_bp = x;
                    if (y < fl_bp)
                        fl_bp = y;
                    if (y > ll_bp)
                        ll_bp = y;
                }
            }
            else if (cb[x+y*LFS] == BLACK)
                cb[x+y*LFS] = GRAY;
        }
    }
}
```

Gambar 3.16 Fungsi Skeleton

f) Membandingkan bitmaps dengan skeleton fitting.

Pada tahap ini simbol atau karakter yang talah mengalami penuningan atau penulangan akan dibandingkan dengan simbol yang bentuk (pattern) dan kerangkanya (skeleton) yang ada pada file library. Adapun simbol yang akan dibandingkan tersimpan dalam variable mcbm sedangkan kerangka yang dihasilkan dari fungsi skel akan disimpan dalam variable mskel. Sedangkan tinggi dan lebar bitmap dari skeleton simbol tersebut tersimpan dalam variable skw dan skh yang digunakan pada fungsi di bawah ini :

```
int bmpcmp_skel(int c,int st,int k,int mode)
{
   static unsigned work[BMS/4],aux[BMS/4];
    int l,lf;
    static int u,lu,lv,v,du,dv;
    static pdesc *d;
    if (st == 1) {
       if (mode == 2) {
            /* copy pattern ke mcbm */
            memcpy(mcbm,pattern[c].b,BMS);
            /* copy pattern skeleton ke mcskel */
            memcpy(mcskel,pattern[c].s,BMS);
        }
        /* classifikasi symbol c */
        else {
            int x,y;
            unsigned char m,*p,*q;
            /* copy symbol ke mcbm */
            copy_mc(mcbm,c);
            /*Hitung symbol skeleton */
            memset(work,0,BMS);
            pixel_mlist(c);
            skel(-2,c);
            skw = lc_bp - fc_bp + 1;
            skh = ll_bp - fl_bp + 1;
           /* copy mc skeleton ke mcskel buffer */
            memset(mcskel,0,BMS);
            for (y=fl_bp; y<=ll_bp; ++y) {
                p = ((unsigned char *) cb) + fc_bp + y*LFS;
                q = ((unsigned char *) mcskel) + (y-
fl_bp)*(FS/8);
                m = 128;
                for (x=fc_bp; x<=lc_bp; ++x) {
                    if (*p == BLACK) {*q|= m;  }
```

```
                if ((m >>= 1) <= 0) {
                    ++q;
                    m = 128;
                    }
                    ++p;
                }
            }
        }
    }
    if (st == 2) {
        int dnbp=0;
        d = pattern + k;
        if ((d->tr != NULL) && (strlen(d->tr) == 1) &&
            (0 <= d->pt) && (d->pt <= toppt)) {
            int p;
            p = pt[d->pt].sc[((unsigned char *)(d->tr))[0]];
            if ((p & (1 << (CL_SKEL-1))) == 0)
        return(0);
    } if (mode == 1) {
       du = mc[c].r - mc[c].l + 1 - d->sw;
       dv = mc[c].b - mc[c].t + 1 - d->sh;
       dnbp = ((PNT*abs(d->bp-mc[c].nbp)) > (d->bp+mc[c].nbp));
     }
     else if (mode == 0) {
       du = d->bw - skw;
       dv = d->bh - skh;
       dnbp = ((PNT*abs(d->bp-mc[c].nbp)) > (d->bp+mc[c].nbp));
     }
     else if (mode == 2) {
       du = pattern[c].bw - d->sw;
       dv = pattern[c].bh - d->sh;
       dnbp    =    ((PNT*abs(d->bp-pattern[c].bp))    >    (d-
>bp+pattern[c].bp));
        }
        if ((du < 0) || (dv < 0) ||
            (((du > 2*MD) || (dv > 2*MD))) ||
            (dnbp)) {
            return(0);
        }else {
            if (du > MD)
                du = MD;
             u = lu = 0;
             v = lv = 0;
            if (mode == 1)
                memcpy(work,mcbm,BMS);
            else if (mode == 0)
                memcpy(work,d->b,BMS);
            else if (mode == 2)
                memcpy(work,mcbm,BMS);
        }
         return(1); }
```

```
        while (st == 3) {
         if ((shift_opt != 0) && (lv == v) && (u > 0)) {
             int i,j,l;
             unsigned a;
             for (j=0; j<FS; ++j) { l = j * FS/8;
                 for (i=FS/8-1; i>=0; --i) {
                     *(((unsigned char *)aux)+l+i) >>= (u-lu);
                     if (i > 0) {a = (1 << (u-lu)) - 1;
                        a &= *(((unsigned char *)aux)+l+i-1);
                         a <<= 8 - (u-lu);
                         *(((unsigned char *)aux)+l+i) |= a;
                     }
                 }
             }
         }
         else if (shift_opt) {
             memset(aux,0,v*BLS);
             if(mode==1)memcpy(((unsignedchar     *)aux)+v*BLS,d-
>s,BMS-v*BLS);
             else if (mode == 0)
                 memcpy(((unsigned char *)aux)+v*BLS,mcskel,BMS-
v*BLS);
             else if (mode == 2)
                 memcpy(((unsigned   char   *)aux)+v*BLS,d->s,BMS-
v*BLS);
         }
         else {
             int i,j,rj;
             unsigned char m,n,*p,*q,*a;
             if (mode == 1)
                 a = ((unsigned char *) d->s);
                else if (mode == 0)
                 a = ((unsigned char *) mcskel);
             else /* if (mode == 2) */
                 a = ((unsigned char *) d->s);
             memset(aux,0,BMS);
             for (j=0; j<d->bh; ++j) {
                 p = a + BLS*j;
                 rj = j+v;
                 if ((0 <= rj) && (rj < FS)) {
                     q=(unsigned char*) (aux+rj*(FS/32));
                     m = 128;
                     n = 128 >> u;
                     for (i=u; i<FS; ++i) {
                         if ((*p & m) != 0)
                             *q |= n;
                             if ((m >>= 1) <= 0) {
                             ++p;
                             m = 128;
                         }
                         if ((n >>= 1) <= 0) {
                             ++q;
                             n = 128;
                         }
```

```
}}}}
        lv = v;
        lu = u;
        /* compare */
        if (shape_opt != 0) {
            l = v * FS/32;
            lf = l + (d->bh) * FS/32;
            if (lf > BMS/4)
                lf = BMS/4;
for(;(l<lf)&&((work[l]&aux[l])==aux[l]); ++l);
        }
        else {
            lf = BMS/4;
            for  (l=0;  (l<lf)  &&  ((work[l]&aux[l])==aux[l]);
++l);
        } if (l >= lf) {
            return(10);
        }
/*Hitung kesamaan antara aux[l] dg work[l] */
                { unsigned m;
                  unsigned char *p;
                  int i;
                  m = aux[l];
                  p = (unsigned char *) &m;
                  do {
                     for (i=3; i>=0; --i) {
                         p[i] >>= 1;
                         if ((i > 0) && (p[i-1] & 1))
                             p[i] |= 128;
                     }
                  } while ((++u<=du) && ((m&work[l]) != m));
                  if (u > du) {
                      ++v;
                      u = 0;
                  }
                  if (v > dv) {
                      return(0);
                  }
            }
        } else {
            if (++u > du) {
                if (++v > dv)
                    return(0);
                else
                    u = 0;
            }
        }
    } if (st == 0) {
        display_match(work,aux,1);
    }
    return(0);}
```

Gambar 3.17 Skeleton Fitting

g) Start OCR

Fungsi ini digunakan manakala user menekan tombol "PROSES" pada tampilan GUI untuk melakukan proses pengenalan karakter/symbol untuk semua data yang terdapat dalam page data atau hanya data yang dipilih. Sebagaimana yang terlihat dari fungi dibawah ini jika p = -1 maka fungsi tersebut akan melakukan proses pengenalan pada semua data inputan dan jika p < -1 maka fungsi tersebut hanya melakukan proses pengenalan hanya pada data yang ada page yamg bersangkutan. Sedangkan nilai s digunakan untuk melakukan langkah-langkah pengenalan.

```
void start_ocr(int p,int s,int r)
{
     if (ocring)
        return;
    if (s > LAST_STEP) {
        fatal(DI,"invalid page or step number when starting
OCR");
    }
    if (!ocring) {
        starting = 1;
        ocring = 1;
        onlystep = s;
        if ((onlystep != OCR_SAVE) && (onlystep != OCR_LOAD))
{
            button[bocr] = 1;
            redraw_button = bocr;
        }
        if (p == -1)
            ocr_all = 1;
        else {
            ocr_all = 0;
            to_ocr = (p == -2) ? cpage : p;
        }
        recomp_cl = r;
        to_tr[0] = 0;
    }
}
```

Gambar 3.18 Fungsi start_ocr

h) Generate Output

Hasil pengenalan dari program ini yang terdapat pada bagian PAGE_OUTPUT akan di simpan dalam file yang berformat ".txt" (Hasil.txt), sehingga hasil dari proses pengenalan tersebut yang merupakan tujuan utama dari Tugas akhir ini dapat dibaca dan di edit sesuai kebutuhan oleh word processor baik yang ada dalam OS unix/linux seperti Koffice atau Ms Office yang ada dalam OS Windows.

Fungsi ini akan dijalankan ketika user mengklik "save file" yang ada pada menu file dari user interface program ini.

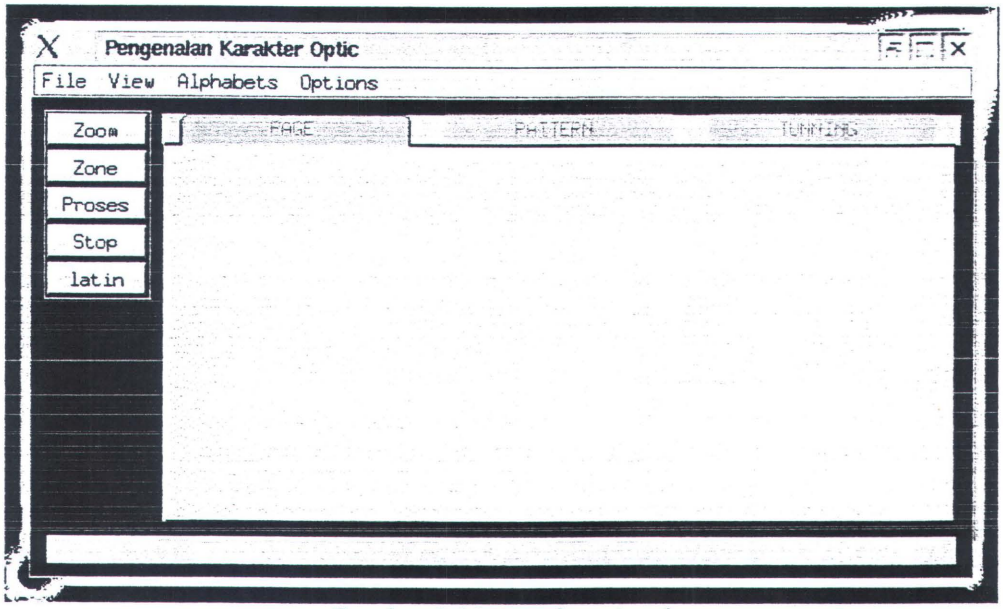
```
else if (it == CM_F_REP) {
            int F;
            cur_dw = CDW;
            CDW = PAGE_OUTPUT;
            ge2txt();
            F                                                    =
open("Hasil.txt",O_WRONLY|O_CREAT|O_TRUNC,0644);
            write(F,text,strlen(text));
            close(F);
            show_hint(2,"Hasil Pengenalan Tersimpan Dalam File
Hasil.txt");
            CDW = cur_dw;
            new_mclip = 0;
        }
```

Gamba 3.19

Sub program untuk menghasilkan file Hasil.txt

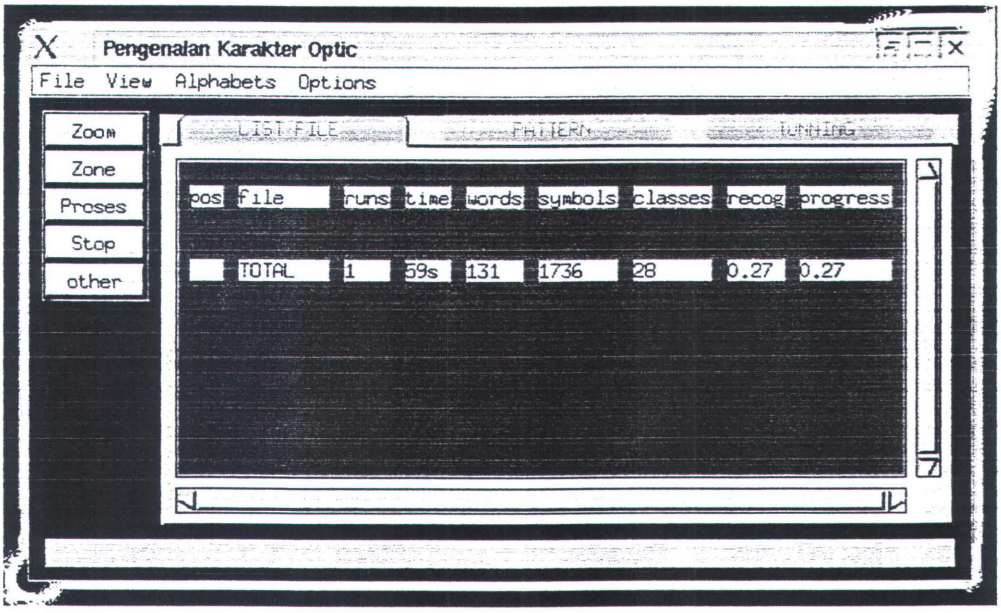
3.5 Rancangan User Inetrface

Tampilan saat program dijalankan



Gambar 3.20 Interface Awal

Load File



Gambar 3.21 Load File

Tampilan di atas digunakan untuk menampilkan file-file dokumen yang akan diproses untuk melakukan proses pengenalan karakter-karakter yang ada pada dokumen tersebut. Adapun maksimal input yang dapat ditampilkan dibatasi hanya sampai 4 dokumen. Interface di atas muncul ketika user memilih Load file pada menu File atau mengkilik tab "PAGE". Ketika user memilih file mana yang akan diproses, fungsi sekaligus akan menghitung jumlah karakter atau simbol serta closure yang dimiliki dokumen tersebut yang digunakan untuk proses selanjutnya.

BAB IV

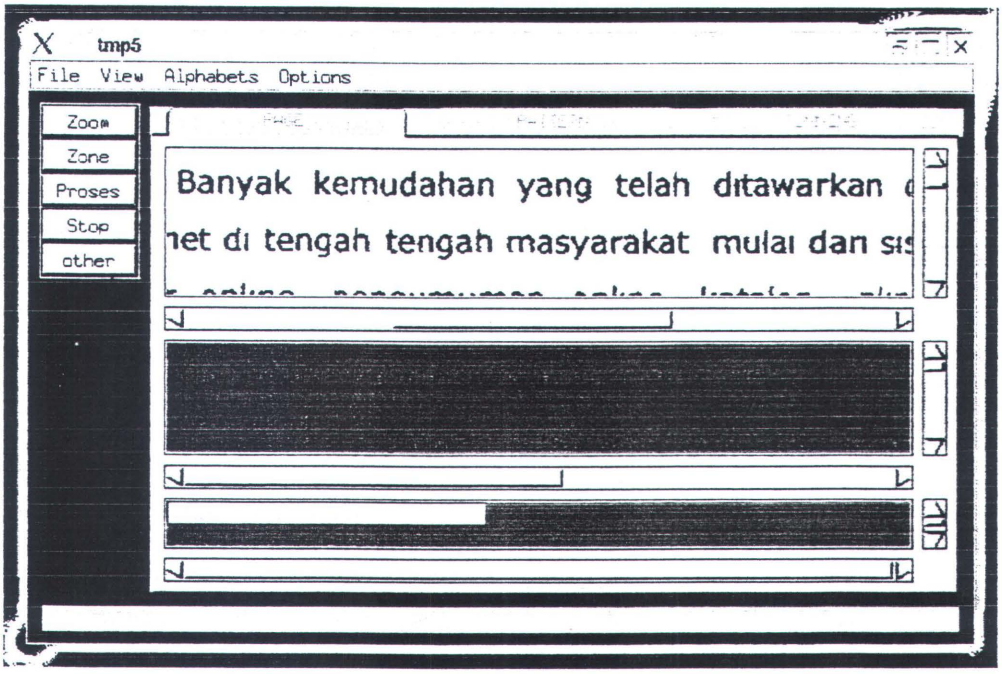
IMPLEMENTASI DAN EVALUASI PERANGKAT LUNAK

BAB IV

IMPLEMENTASI DAN EVALUASI PERANGKAT LUNAK

Pada bab ini berisi hasil pengamatan terhadap prilaku atau jalannya program, yaitu prilaku dari tiap-tiap procedure yang berpengaruh pada proses pengenalan karakter. Pada dasarnya semua procedure dapat berfungsi dengan baik atau sebagai mana mestinya.

❖ Proses Training



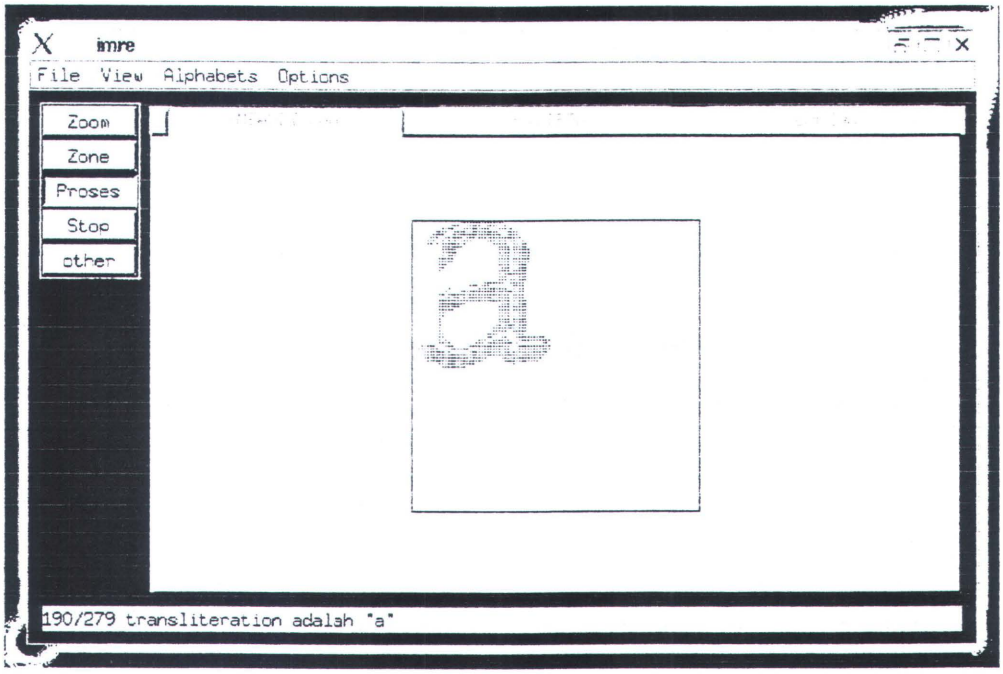
Gambar 4.1 Training

Proses training yang dilakukan ketika user telah menginputkan karakter-karakter yang bersesuaian dari keyboard dengan dokumen seperti yang terlihat pada gambar di atas karakter-karakter yang memiliki kesamaan bentuk setelah

proses Training yang dilakukan dengan mengklik tombol proses akan ditandai dengan warna hitam. Adapun gambar di atas dibagi menjadi tiga bagian, yang pertama merupakan tempat dimana file input ditampilkan, bagian yang kedua merupakan tempat output yang nantinya akan tersimpan dalam file Hasil.txt dan yang window ke tiga merupakan tempat inputan yang dimasukan oleh user dengan keyboard untuk proses Training.

Adapun simbol-simbol yang belum dikenali akan diberi nomor index yang berurutan sesuai dengan urutan simbol pada file dokumen.

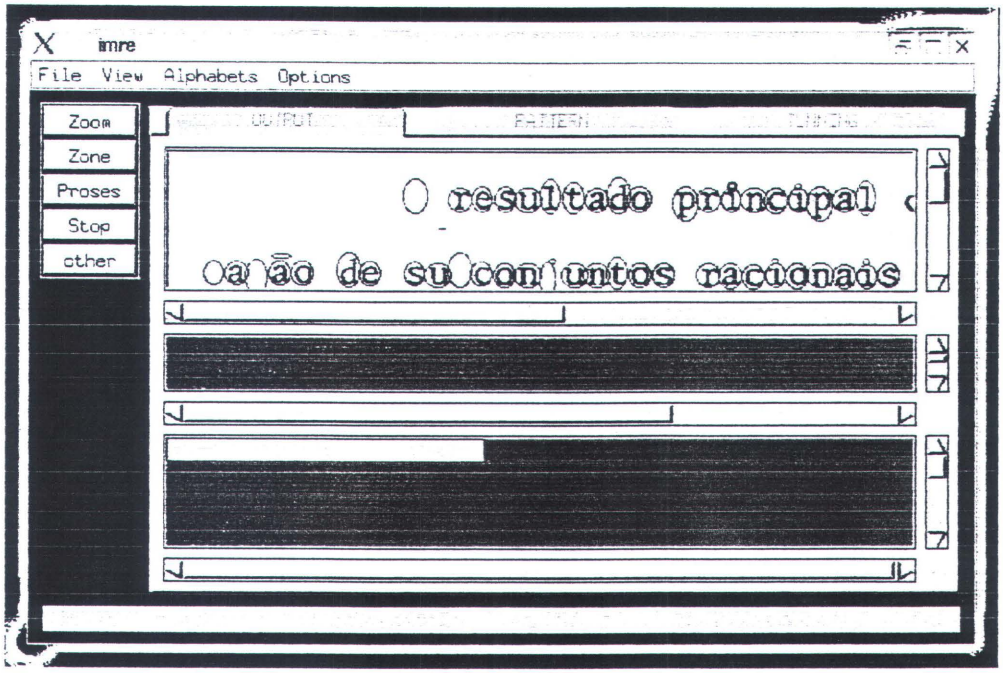
❖ **Proses Pembandingan**



Gambar 4.2 Pembandingan

Interface di atas ditampilkan ketika user memilih proses pembandingan dari menu view. Proses di atas akan membandingkan karakter-karakter yang dimasukan dengan karakter-karakter yang ada pada file dokumen, proses di atas akan terus dilakukan sampai karakter yang ada pada file dokumen habis dibandingkan. Interface di atas akan memperlihatkan apakah karakter atau simbol "match" dengan dengan yang diinputkan.

- **Batas-batas Simbol**



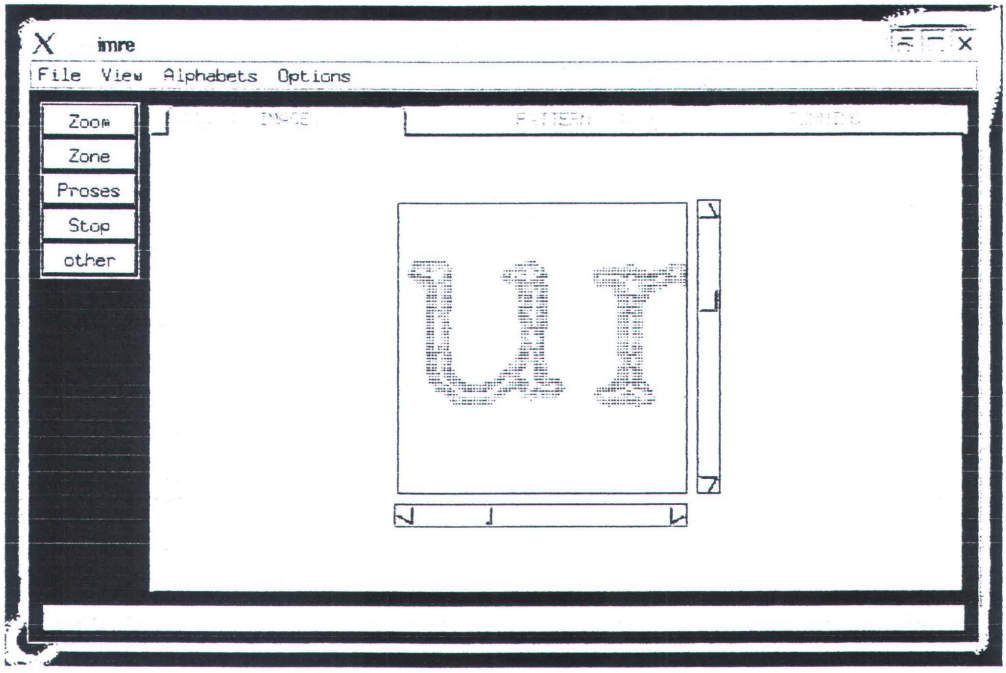
Gambar 4.3 Batas-batas simbol

Simbol-simbol yang ada pada dokumen akan ditandai atau di batasi dengan lingkaran yang berbentuk elip yang mengelilingi karakter/simbol tersebut baik

yang belum di-Training maupun yang sudah di-Training. Ini ditampilkan ketika user memilih "Simbol" pada menu View.

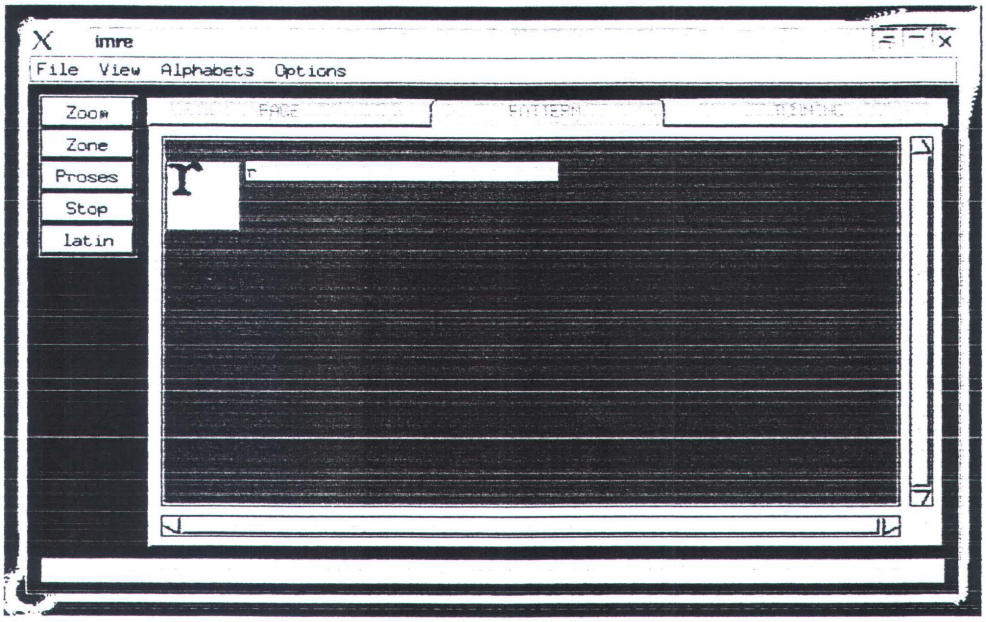
❖ **Skeleton**

Karakter-karakter yang telah mengalami penulangan dapat diperlihatkan bentuk atau rupa kerangkanya dari masing-masing karakter atau simbol dengan memilih menu "Skeleton" pada menu "View". Seperti yang terlihat pada gambar dibawah ini karakter "u" dan "r" memiliki kerangka dasar seperti yang terlihat di bawah ini.



Gambar 4.4 Skeleton karakter

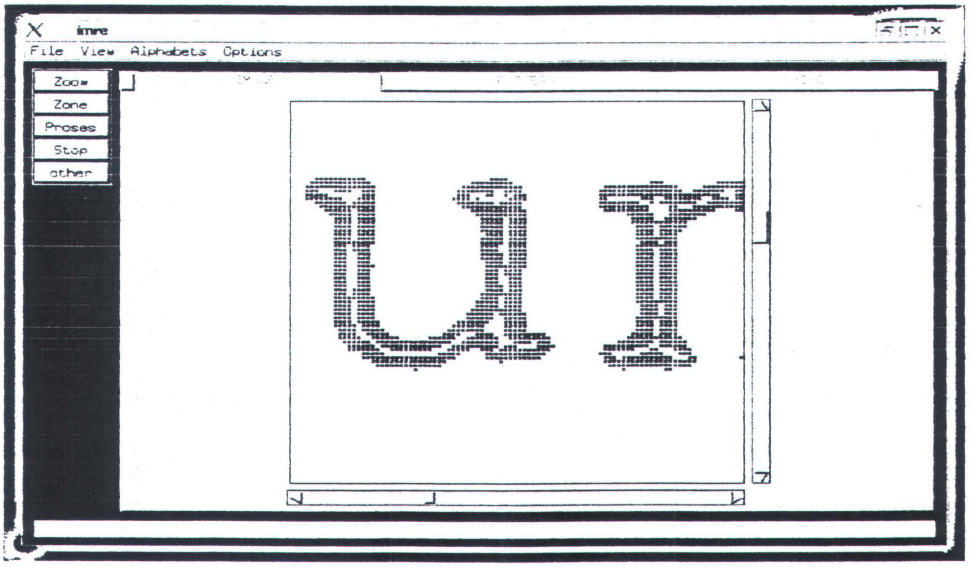
❖ **Patttern**



Gambar 4.5 Pattern

Berisi informasi tentang karakter atau simbol yang ada pada file dokumen beserta pembandingannya dengan karakter input yang bersangkutan.

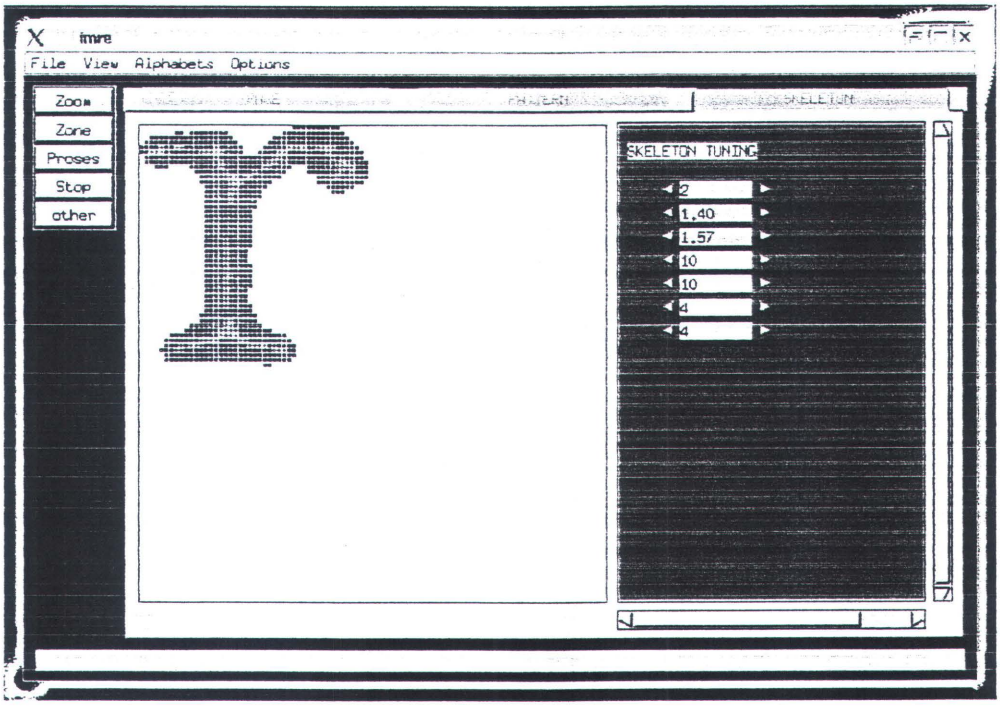
❖ **Zooming**



Gambar 4.6 Zooming

Interface tambahan yang digunakan untuk memperjelas titik-titik pixel yang membentuk karakter pada file dokumen. Sebagai contoh pada gambar di atas menampilkan karakter dengan kerangkanya yang mengalami pembesaran ukuran. Ini tidak mempengaruhi penghitungan atau proses pengenalan karakter.

❖ **Proses Skeleton**



Gambar 4.7 Pembentukan Skeleton

Proses penulangan yang dilakukan pada aplikasi ini menggunakan bantuan parameter-parameter seperti yang terlihat pada gambar di atas, yang mana parametr-parameter tersebut dapat di-edit sesuai kebutuhan untuk menentukan bentuk kerangka yang tepat pada karakter yang akan dikenali. Dalam Tugas Akhir

ini parameter-parameter tersebut dibatasi dengan nilai yang telah ditetapkan dengan asumsi nilai tersebut adalah nilai yang paling match untuk semua karakter yang akan dikenali.

Metode yang digunakan untuk melakukan klasifikasi pada proses di atas adalah skeleton fitting, dimana dua simbol dikatan memiliki kesamaan ketika dua simbol tersebut memiliki skeleton (kerangka) yang sama dengan yang lainnyan. Dalam melakukan klasifikasi simbol digunakan 5 parameter heuristik antara lain SA yang memiliki nilai 0,1,2,3 dan 4.Heuristic 0,1, dan 2 menunjukkan pixel dari pixsel skeleton simbol yang bersangkutan, jika heuristic tersebut terletak di tengah-tengah dari pada closur, dan Heuristics 0, 1 and 2 dapat dijadikan pixel yang merupakan bagian dari skeleton pixel jika pixel tersebut terletak ditengah-tengah dari daerah closure simbol yang bersangkutan dan garis singgung (tangent) ke batas pattern (pattern boundary)lebih dari satu titik. Adapun implementasinya pada tugas Akhir ini adalah sebagai berikut:

Untuk masing-masing pixel dari closure p, hitung jarak minimum d dari p ke beberapa daerah batas (boundary) pixel. Setelah itu dicari dua pixel pada daerah batas (boundary) closure seperti jarak dari masing dua pixel tersebut ke p yang tidak jauh beda dengan jarak dari d (harus kurang atau samadenga RR). pixel tersebut diwakili dengan "BPs".

Untuk membuat algoritma lebih cepat ,jarak maksimum dari p ke boundary pixel yang dinyatakan dengan (considered) "RX", maka jarak maksimum tersebut dinyatakan dengan kuadrat dari 2*BT+1 yang terketak ditengah p, kemudian p dinyatakan sebagai skeleton pixel.

- Heuristic 1 jika (SA = 1) jarak minimum (linear) antara dua BPs yang dispesifikasikan sebagai faktor (ML) pada kuadrat radius. Dengan kondisi ini akan mencegah perubahan koordinat dari rectangular ke koordinat polar.

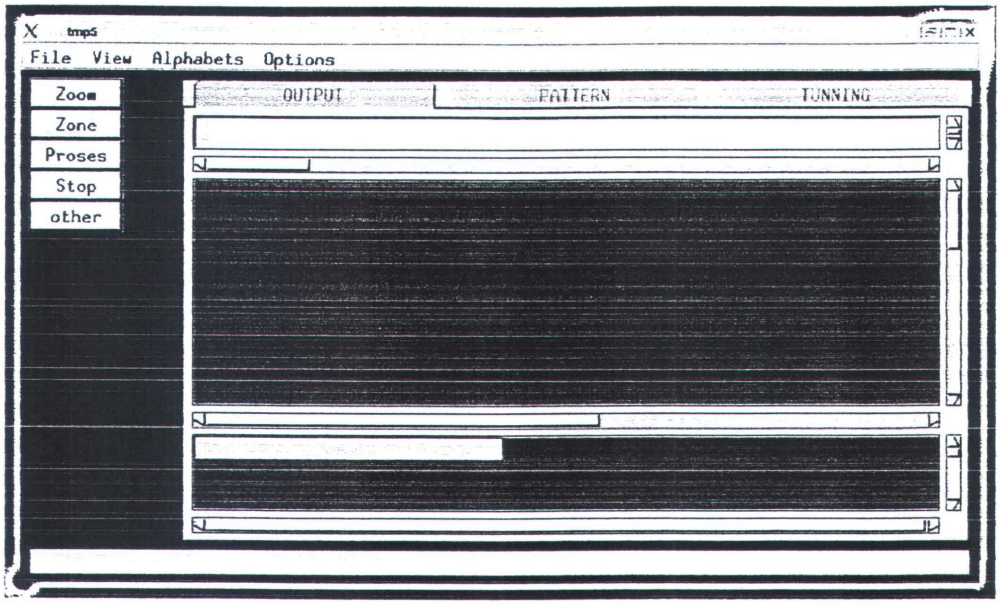
- Heuristic 2 jika (SA = 2) tidak mengecek jarak (distance) minimum, tetapi menggunakan parameter MB dan BPS untuk menentukan pixel p sebagai skleton pixel.

- Heuristic 3 digunakan untuk menghitung perubahan skeleton sebesar BT pada batas pixel..

- Heuristic 4 digunakan untuk "growing lines". Untuk sudut yang berlainan ditetapkan kira-kira sebesar 22 derajat, panjang garis pixel RX digambarkan dari masing-masing pixel, kemudian heuristic akan mengecek apakah garis(line) tersebut dapat atau tidak digambarkan sebagai piksel hitam (black pixel). Tergantung pada hasil pengecekan tersebut, maka akan diputuskan apakah pixel tersebut termasuk skeleton pixel atau bukan. Sebagai contoh : jika semua line atau garis dapat digambarkan sebagai pixel hitam, kemudian pixel tersebut terletak ditengah-tengah pada daerah lingkaran, maka pixel tersebut dapat dijadikan bagian dari skeleton pixel

- Heuristic 5 menghitung jarak dari masing-masing pixel ke perbatasan pixel

❖ **Output**



Gambar 4.8 Hasil output

Setelah proses pengenalan selesai, maka hasil karakter telah dikenali akan di cetak pada page "OUTPUT" yang kemudian dapat disimpan dalam File Hasil.txt, yang merupakan file text yang dapat dibuka oleh word processor baik yang berjalan pada OS Unix/Linux maupun yang berjalan pada OS Windows.

BAB V

KESIMPULAN DAN SARAN

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengamatan selam melaksanakan perencanaan dan perancangan perangkat lunak yang beruang lingkup pengenalan karakter alfabet dan simbol ini dapat ditarik beberapa kesimpulan :

- Pengenalan karakter alfabet dan simbol merupakan salah satu lingkup aplikasi dari Pengenalan Pola yang berbasis Pengolahan Citra Digital. Hal yang mendasar dari aplikasi yang berbasis Pengolahan Citra Digital ini adalah berlangsungnya dua proses pokok, yaitu pengimputan karakter atau simbol dari keyboard dan pemrosesan data citra secara otomatis berdasarkan inputan dari keyboard.
- Sistem pengenalan karakter alfabet yang dirancang perlu menetapkan batasan-batasan tertentu terhadap karakter alfabet atau simbol yang hendak dikenali. Suatu karakter alfabet atau simbol akan dapat dikenali apabila memiliki bentuk yang mirip dengan bentuk bakunya. Adapun jenis karakter atau simbol yang dapat dikenali adalah karakter "A" sampai dengan "Z" dan angka "0" sampai dengan "9".
- Dalam software aplikasi ini, proses pengenalan tidak bisa berhasil seratus persen. Hal ini disebabkan karena anatara lain :
    1. Peralatan scanner yang digunakan untuk mengambil data memiliki ketelitian yang terbatas.

2. Pada saat pengambilan data (menjalankan scanner) terjadi kekurangan-kekurangan yang dapat menyebabkan mutu kualitas citra input kualitasnya kurang bagus

- Secara umum program untuk membaca teks dalam huruf standar yang merupakan output dari scanner ini dapat dikatakan telah berhasil melakukkan fungsinya, yaitu untuk mengenali karakter-karakter atau simbol dalam bentuk grafik untuk kemudian diubah menjadi karakter-karakter dalam mode teks, sehingga dapat dimengeti oleh komputer dan dapat diedit dengan menggunakan word processor yang ada dipasarkan baik OS Linux maupun yang ada dalam OS Windows.
- Kemiripan bentuk karakter pada karakter alfabet dan adanya pixel yang saling terhubung antara karakter satu dengan yang lain membuat proses pengenalan karakter berjalan gagal dan sulit mengenali batas-batas karakter tersebut, sehingga karakter tersebut tidak dapat dikenali oleh program ini.
- Kesalahan dalam melakukan penyalinan (transliteration) pada proses training menyebabkan simbol yang ada pada dokumen tidak dapat dikenali dengan benar. Sebagai contoh jika simbol yang ada pada dokumen adalah "a" dan user mengiputkan karakter "u" sengaja maupun tidak sengaja akan menyebabkan proses pengenalan tersebut tidak match.

5.2 Saran

- Kelemahan yang dijumpai pada sistem pengenalan karakter alfabet maupun simbol yang telah dibuat mungkin dapat diatasi dengan menggunakan metode lain. Seperti menggunakan metode neural network dan metode fuzzy dimana pengenalan suatu karakter atau simbol secarara lebih matematis.
- Sistem pengenalan karakter atau simbol yang telah dibuat masih terbatas pada karakter-karakter baku, sehingga untuk pengembangan lebih lanjut sistem pengenalan tersebut dapat mengenali karakter-karakter yang tidak baku seperti karakter arab,greek dan lain-lain.
- Sistem pengenal karakter atau simbol yang dibuat belum memuat menu edit yang digunakan untuk mengedit file dokumen, sehingga jika terdapat dua simbol yang bersinggungan tidak dapat dikenali. Dengan adanya pengeditan kelemahan di tersebut dapat teratasi
- Dengan pengembangan aplikasi lebih lanjut dapat dikembangkan menjadi aplikasi yang tidak hanya berjalan pada desktop tetapi dapat berjalan juga pada web.

DAFTAR PUSTAKA