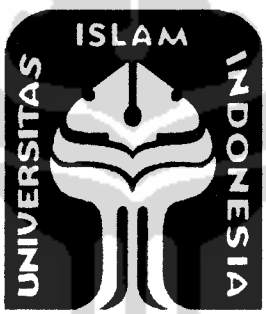
STUDI DAN IMPLEMENTASI ALGORITMA *STENTIFORD THINNING* PADA SEBUAH CITRA BINER

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika



Disusun oleh:

Nama : Sri Handayani
No. Mahasiswa : 03 523 030

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

LEMBAR PENGESAHAN PENGUJI

STUDI DAN IMPLEMENTASI ALGORITMA *STENTIFORD* PADA SEBUAH CITRA BINER

TUGAS AKHIR

Oleh :

Nama : Sri Handayani

No. Mahasiswa : 03 523 030

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Oktober 2007

Tim Penguji

Yudi Prayudi, S.Si., M.Kom
Ketua

Ami Fauzijah, ST., MT
Anggota I

Hendrik, ST
Anggota II

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

(Yudi Prayudi, S.Si, M.Kom)

Dengan penuh rasa syukur yang dipanjatkan kehadirat Allah SWT, kupersembahkan Tugas Akhir ini untuk :

Yang Tercinta :

Bapak (H. Tugiman) dan Ibu (Hj. Misnatun)

Yang senantiasa menjadi tenagaku untuk berbuat, menjadi inspirasiku untuk bertindak, yang telah memberikan Do'a, Cinta, Sayang, Kehangatan, Semangat, Pengorbanan dan Dukungan yang tiada tara kepadaku selama Ini.

Adik-adikku tersayang :

Dian, Pram dan Dede, *makasih buat support dan kasih sayang kalian buat mbak, Terus belajar yang rajin Ya...!!! dan jadi anak yang Sholeh dan Sholehah. Mbak sayang kalian...*

Jleque tersayang :

Terima kasih tak terhingga untuk, semua semua semua dukungan, kasih sayang, nasihat dan selalu menemaniku dalam susah dan senang. Luph u jleque...

Sahabat-sahabatku tersayang :

Teh Nha, Amal, dan Q-dank, *makasih buat support, sayang, dan udah mau jadi tempat berbagiku sejak dari SMP sampai sekarang. Luv u 4ever...(That's what friends are for...)*

Keluarga besar Juandi dan Sarkim :

Makasih banyak buat dukungan, cinta, kasih sayang, dan keluarga yang menyenangkan. Luv u all..

Teman-temanku :

Lia, Jenny, Echa, Andin, Ria, Mukhyar, Fifi, Don-don, Mas Dhona, Asti, Tiara, Re... mb... Hilda, mbak Dilla, mbak Aftri, kak Tracy, Daru, Puti, Prima, Daniel, ... Saiful, Eldy, A... i, *n meine freundin am deutschkurs, makasih buat supportnya dan selalu bikin aku ketawa...makasih banyak,..*

Siapapun yang tidak bisa kusebutkan satu persatu :

Untuk segala kasih sayang, *perkataan dan tindakan yang sangat berarti.*

Almamaterku UII.

MOTTO

" Allah tidak akan merubah nasib suatu kaum kecuali kaum itu yang berusaha untuk merubahnya "

" Ridho Ibu merupakan rezeki yang tidak ada nilainya "

" Sesungguhnya Allah S.W.T akan membantu orang-orang yang berusaha, sekalipun ia tidak memiliki kekuatan dan kemampuan, melainkan kemauan yang kuat serta niat yang tulus dan ikhlas "

" Sholat Dapat Menjernihkan Fikiran Dan Hanya Sholatlah Yang Dapat Meninggikan Derajatmu Dihadapan – Nya "

" APA YANG MENIMPAMU BERUPA KEBAIKAN, ITULAH YANG DATANG DARI ALLAH SWT DAN APA YANG MENIMPAMU BERUPA KEJAHATAN (KEBURUKAN) ITU DATANG DARI DIRIMU SENDIRI ".

(QS. AN-NISA 4:79)

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr. Wb.

Alhamdulillah, segala puji dan syukur penyusun panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat, taufik serta hidayahnya. Sholawat dan salam kepada junjungan kita Nabi Muhammad SAW beserta keluarga dan para sahabat, serta orang-orang yang bertaqwa, sehingga penyusun dapat menyelesaikan Tugas Akhir ini sebagaimana mestinya.

Tugas Akhir ini merupakan salah satu penerapan ilmu yang telah didapatkan selama kuliah. Dengan adanya penelitian ini, penyusun InsyaAllah akan dapat memahami penggunaan Studi dan Implementasi Algoritma *Stentiford* pada sebuah Citra Biner ini.

Penyusun menyampaikan ucapan terimakasih dan penghargaan yang setinggi-tingginya atas bantuan, bimbingan dan dukungan dari berbagai pihak yang ikut serta demi kelancaran pelaksanaan Tugas Akhir kepada :

1. Bapak Fathul Wahid ST, MSc selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bapak Yudi Prayudi, S.Si., M.Kom selaku Ketua Jurusan Teknik Informatika Universitas Islam Indonesia.
3. Bapak Yudi Prayudi, S.Si., M.Kom selaku dosen pembimbing . Terima kasih atas segala bantuan, dukungan, semangat, dan pengetahuannya, serta kemudahan yang telah diberikan.
4. Teman-teman 'Informatika 03', terimakasih atas kekompakan dan kebersamannya selama ini.
5. Semua pihak yang tidak dapat penyusun sebutkan satu persatu yang telah membantu sejak pengumpulan data sampai penyusunan Tugas Akhir ini.

Semoga amal ibadah dan kebaikan yang telah diberikan mendapatkan imbalan yang setimpal dari Allah SWT.

Penyusun menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan, untuk itu penyusun sangat mengharapkan kritik serta saran yang bersifat membangun untuk perbaikan di masa mendatang. Semoga Tugas Akhir ini bermanfaat untuk kita semua. Amin.

Wassalum'alaikum Wr. Wb.

Yogyakarta, September 2007

penyusun

ABSTRAKSI

Thinning merupakan proses penting dalam operasi analisis citra, seperti pada aplikasi pengenalan karakter dan pengenalan pola sebagai *pre-processing*. Operasi ini sangat berguna ketika kita lebih tertarik pada pola relatif obyek ketimbang ukuran obyek. *Image Thinning* merupakan proses *morphology image* yang merubah bentuk asli *binary image* menjadi *image* yang menampilkan batas-batas obyek/*foreground* hanya setebal satu piksel.

Ada beberapa algoritma yang dapat digunakan pada proses *thinning*. Salah satu algoritma *thinning* yaitu algoritma *Stentiford*. Algoritma *thinning* jenis ini menggunakan template untuk dicocokkan dengan citra yang akan di*thinning*, di mana piksel objek yang cocok dengan template, maka piksel tengahnya akan di-*delete*.

Hasil dari proses *thinning* dengan Algoritma *thinning Stentiford* lebih sempurna (tipis) karena algoritma ini melakukan penghapusan secara tidak langsung (*mark-and-delete*). Hasil yang lebih sempurna ini juga disebabkan karena algoritma *Stentiford* memberikan syarat tambahan untuk memperhalus citra yang dihasilkannya. Metode ini cukup terkenal karena *reliable* dan kefektifannya.

Kata kunci : *image processing, image binary, morphology, skeleton, thinning algorithm, Stentiford algorithm*

TAKARIR

image	citra
binary	biner
processing	pengolahan
image processing	pengolahan citra
pattern recognation	pengenalan pola
algorithm	algoritma
thinning	penipisan
morphology	bentuk
skeleton	rangka
skeletonizing	perangkaan
stroke	garis tepi
scan	pindai
enhancement	perbaikan
edge	tepi
sharpening	penajaman
pseudocoloring	warna semu
noise	derau
filtering	penapisan
restoration	pemugaran
deblurring	samar
compression	pemampatan
segmentation	pemecahan
analysis	pengorakan
detection	deteksi
boundary	batas
region	daerah
reconstruction	rekonstruksi
raw data	data mentah
complete	lengkap

external	luar
internal	dalam
characteristic	karakteristik
high	tinggi
low	rendah
true color	banyak warna
thresholding	pengambangan
greyscale	skala keabuan
neighborhood	tetangga
connected	terhubung
redundant	kelebihan
background	latar belakang
foreground	objek
based	dasar
marked	tandai
delete	hapus
connectivity	terhubung
number	nilai
endpoint	nilai akhir
necking	leher
tailing	ekor
input	masukan
output	keluaran
printout	cetakan
flowchart	diagram alir
open	buka
save	simpan
feedback	balasan

DAFTAR ISI

DAFTAR TABEL

DAFTAR FLOWCHART

DAFTAR GAMBAR

LEMBAR PENGESAHAN PEMBIMBING

STUDI DAN IMPLEMENTASI ALGORITMA *STENTIFORD*
PADA SEBUAH CITRA BINER

TUGAS AKHIR

Oleh :

Nama : Sri Handayani
No. Mahasiswa : 03 523 030

Yogyakarta, September 2007

Pembimbing

(Yudi Prayudi, S.Si., M.Kom)

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pada masa sekarang ini, teknologi multimedia sedang berkembang dengan pesat. Salah satunya adalah teknik pengolahan citra. Dalam teknik pengolahan citra, dikenal suatu teknik pengenalan pola atau *pattern recognition*.

Dalam operasi analisis citra (seperti aplikasi pengenalan pola dan pengenalan karakter) *thinning* merupakan suatu langkah yang penting. Proses *thinning* mengidentifikasi piksel-piksel dari suatu objek yang dianggap mewakili bentuk objek tersebut. Dengan *thinning*, suatu fitur dari objek pada suatu citra dapat dikenali dan diekstrak sehingga dapat digunakan untuk pemrosesan lebih lanjut.

Operasi *thinning* merupakan salah satu dari beberapa operasi *morphology*, operasi *thinning* ini dilakukan dengan cara mereduksi titik-titik atau layer terluar dari sebuah citra sampai semua garis atau kurva hanya tinggal setebal satu piksel. Proses *thinning* biasa digunakan dalam *skeletonization*. *Skeleton* atau rangka memberikan representasi simpel dari suatu bentuk objek citra yang tetap mempertahankan karakteristik ukuran, posisi dan topologi dari bentuk aslinya.

Dengan melihat banyaknya manfaat dari proses *thinning* dalam teknik pengolahan citra, maka pada penelitian ini akan dijelaskan tentang *thinning*, termasuk algoritma dan langkah-langkah eksekusi yang digunakan dalam proses *thinning*.

1.2. Rumusan Masalah

Ada beberapa algoritma yang dapat digunakan dalam proses *thinning*. Masing-masing algoritma memiliki kelebihan dan kekurangan. Masing-masing algoritma hanya memberikan hasil yang bagus pada bentuk objek tertentu saja, tidak untuk bentuk lainnya. Oleh karena itu, sulit untuk menentukan algoritma yang umum dan dapat memberikan hasil yang baik untuk berbagai macam bentuk

objek dari sebuah citra. Di samping itu, terdapat masalah lamanya waktu yang digunakan untuk menyelesaikan sebuah operasi *thinning* pada beberapa metode.

Tantangan berikutnya adalah bagaimana membangun sebuah perangkat lunak pengolahan citra biner dengan menerapkan teknik representasi algoritma *thinning* agar lebih bermanfaat untuk proses deskripsi citra pada tahap selanjutnya.

1.3. Batasan Masalah

Dalam suatu penelitian perlu adanya pembatasan masalah agar penelitian lebih terarah dan memudahkan dalam pembahasan sehingga tujuan penelitian dapat tercapai.

Batasan-batasan dari penelitian ini adalah :

1. Aplikasi hanya berjalan pada sistem operasi Windows.
2. Algoritma yang dipakai pada proses penyelesaian masalah adalah algoritma *Stentiford.*
3. Citra digital yang diolah adalah citra biner.
4. Citra digital yang akan diolah bukan merupakan citra abstrak dan rusak.
5. File grafik yang digunakan adalah file grafik berformat bitmap (*.bmp).
6. Ukuran citra yang akan diolah maksimal 290 piksel x 290 piksel.
7. Citra yang sudah diolah disimpan dalam format bitmap (*.bmp) dan dapat didokumentasikan dalam bentuk *printout.*

1.4. Tujuan Penelitian

Penelitian ini bertujuan untuk memberikan penjelasan mengenai konsep algoritma *Stentiford*, memaparkan kelebihan dan kekurangan, dan implementasi dari algoritma ini. Selain itu, penelitian ini juga bertujuan untuk membandingkan hasil dan efisiensi dari metode *thinning Stentiford* dengan hasil penelitian sebelumnya yang menggunakan metode *thinning Hilditch.*

1.5. Manfaat Penelitian

Hasil dari penelitian dan perancangan perangkat lunak dari algoritma *Stentiford* ini, diharapkan dapat memberikan manfaat sebagai berikut :

1. Studi awal untuk pengembangan aplikasi pengolahan citra, khususnya citra biner dengan algoritma *thinning* agar citra biner yang telah diolah dapat lebih bermanfaat untuk proses deskripsi selanjutnya.
2. Sebagai langkah awal dalam pengenalan pola yang berguna pada pra-analisis dokumen untuk mengenali garis-garis pada gambar dan *stroke* karakter pada teks. Yang secara aplikatif dapat berupa aplikasi pengenalan karakter, pengenalan kromosom, dan lain sebagainya.

1.6. Penelitian Sejenis

Sebelum penulis melakukan penelitian mengenai proses *thinning* citra menggunakan metode *Stentiford*, terdapat penelitian sejenis mengenai *thinning* citra, namun dengan metode yang berbeda. Metode yang digunakan dalam penelitian tersebut adalah metode *Hilditch*, yang akan penulis jelaskan pada bagian lain dalam laporan ini.

1.7. Hipotesa

1. Algoritma *Stentiford* cukup terkenal dalam teknik pengolahan citra karena algoritma ini sangat *reliable* dan efektif.
2. Algoritma *Stentiford* menggunakan teknik *template based mark-and-delete*. Di mana piksel citra yang cocok dengan template yang disediakan oleh algoritma *Stentiford* akan dihapus.
3. Algoritma *Stentiford* bersifat iteratif yang berguna untuk mengikis lapisan piksel terluar sampai tidak ada lagi lapisan yang dapat dihilangkan.

1.8. Metodologi Penelitian

Metode yang digunakan dalam penelitian Tugas Akhir ini meliputi metode pengumpulan data dan pengembangan sistem.

1. Studi Literatur

   Dengan mengambil bahan penelitian dari buku-buku yang sudah ada dan mengambil sumber dari website-website sebagai acuan dalam menganalisis dan perancangan sistem.

2. Studi Pustaka

   Dengan melakukan kunjungan ke pustaka untuk mencari buku yang berkaitan dengan penelitian ini.

3. Metode pengembangan sistem

   Metode ini meliputi analisis perangkat lunak, perancangan perangkat lunak, implementasi perangkat lunak dan analisis kinerja perangkat lunak.

1.9. Sistematika Penulisan

BAB I PENDAHULUAN

Bab ini berisikan tentang latar belakang munculnya masalah, merumuskan masalah dalam penelitian, menegaskan batasan-batasan masalah yang terdapat dalam penelitian, tujuan dari penelitian, manfaat penelitian bagi peneliti dan dunia akademik, hipotesis yang akan dibuktikan dalam penelitian, dan sistematika penulisan laporan.

BAB II LANDASAN TEORI

Bab ini memuat dasar teori yang berfungsi sebagai sumber atau alat dalam memahami permasalahan yang berkaitan dengan konsep citra, pengolahan citra, format citra, operasi *morphology*, *skeletonization*, *pattern recognition*, *thinning*, algoritma *thinning*, algoritma *Stentiford* dan konsep citra dalam delphi.

BAB III METODOLOGI

Bab ini memuat uraian tentang metode analisis kebutuhan perangkat lunak yang dipakai, serta hasil analisis kebutuhan perangkat lunak yang berupa analisis kebutuhan proses, analisis kebutuhan masukan, analisis kebutuhan keluaran, kebutuhan perangkat keras dan kebutuhan antarmuka.

Pada bagian perancangan perangkat lunak dibahas mengenai metode perancangan yang digunakan, hasil perancangan yang berupa perancangan diagram arus data dan perancangan basis pengetahuan.

Pada bagian implementasi perangkat lunak dibahas tentang batasan implementasi aplikasi yang dibuat dan memuat dokumentasi atau tampilan form-form yang telah dibangun.

BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas mengenai analisis kinerja dari perangkat lunak, analisis hasil pengujian terhadap sistem yang dibandingkan dengan kebenaran dan kesesuaiannya dengan kebutuhan perangkat lunak yang telah dituliskan pada bagian sebelumnya.

BAB V SIMPULAN DAN SARAN

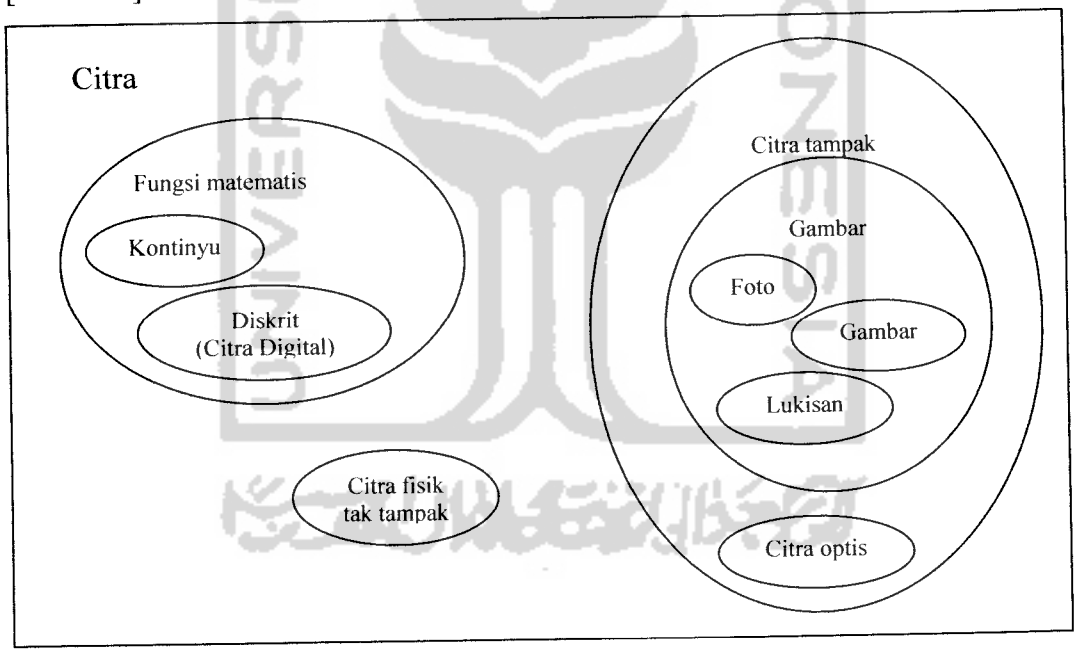
Bab ini berisikan kesimpulan-kesimpulan yang merupakan rangkuman dari hasil analisis perangkat lunak dan kinerja perangkat lunak pada bagian sebelumnya dan saran yang perlu diperhatikan berdasarkan keterbatasan yang ditemukan dan asumsi-asumsi yang dibuat selama pembuatan aplikasi.

BAB II
LANDASAN TEORI

2.1. Citra

Citra merupakan salah satu komponen penting dalam multimedia yang memuat banyak informasi. Definisi citra menurut Kamus Webster adalah suatu representasi, kemiripan, atau imitasi dari suatu objek atau benda.

Citra dapat dikelompokkan menjadi citra tampak dan citra tak tampak. Contoh citra tampak seperti : foto keluarga, apa yang nampak di layar monitor dan telivisi, serta hologram. Sedangkan citra tak tampak seperti : data gambar dalam file (citra digital) dan citra yang direpresentasikan menjadi fungsi matematis [ACH05a].



Gambar 1 : Pengelompokan jenis-jenis citra

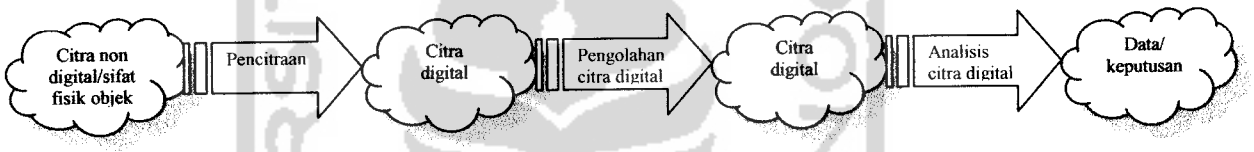
Dari jenis-jenis citra yang dijelaskan di atas, citra digital-lah yang dapat diolah menggunakan komputer. Jika jenis citra yang lain hendak diolah juga menggunakan komputer, maka citra tersebut harus diubah dulu menjadi citra digital, misalnya dengan cara di-*scan* atau dengan cara informasi densitas dan

komposisi bagian dalam tubuh manusia ditangkap dengan bantuan sinar-X dan sistem deteksi radiasi menjadi informasi digital.

Kegiatan mengubah informasi citra fisik non digital menjadi citra digital disebut sebagai pencitraan.

2.2. Pengolahan Citra

Pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer. Pengolahan citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia atau mesin (komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas yang lebih baik daripada citra masukan.



Gambar 2 : Urutan pengolahan citra digital

Operasi pengolahan citra pada dasarnya dilakukan dengan cara memodifikasi setiap titik dalam citra tersebut sesuai keperluan. Secara garis besar, modifikasi tersebut dikelompokkan menjadi [ACH05a] :

1. Operasi titik, di mana karakteristik setiap titik diolah secara tidak gayut terhadap titik yang lain.
2. Operasi temporal/berbasis bingkai, di mana sebuah citra diolah dengan cara dikombinasikan dengan citra lain.
3. Operasi geometri, di mana bentuk, ukuran, atau orientasi citra dimodifikasi secara geometris.
4. Operasi banyak titik tetangga, di mana data dari titik-titik yang bersebelahan (bertetangga) dengan titik yang ditinjau ikut berperan dalam mengubah nilai.
5. Operasi *morphology*, yaitu operasi yang berdasarkan segmen atau bagian dalam citra yang menjadi perhatian.

Operasi pengolahan citra dapat diklasifikasikan sebagai berikut [MUN04]:

1. Perbaikan kualitas citra (*image enhancement*)

   Operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Misalnya :

   - Perbaikan kontras gelap/terang
   - Perbaikan tepian objek (*edge enhancement*)
   - Penajaman (*sharpening*)
   - Pemberian warna semu (*pseudocoloring*)
   - Penapisan derau (*noise filtering*)

2. Pemugaran citra (*image restoration*)

   Operasi ini bertujuan untuk menghilangkan cacat pada citra dan dapat mengetahui penyebab degradasi gambar. Misalnya :

   - Penghilangan kesamaran (*deblurring*)
   - Penghilangan derau (*noise*)

3. Pemampatan citra (*image compression*)

   Operasi ini bertujuan untuk mereduksi ukuran citra dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit dengan tetap memperhatikan kualitas citra. Misalnya adalah metode JPEG.

4. Segmentasi citra (*image segmentation*)

   Operasi ini bertujuan untuk memecah suatu citra kedalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengenalan pola (*pattern recognition*).

5. Pengorakan citra (*image analysis*)

   Operasi ini bertujuan untuk menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Misalnya:

   - Deteksi tepi (*edge detection*)
   - Ekstraksi batas (*boundary*)
   - Representasi daerah (*region*)

6. Rekonstruksi citra (*image reconstruction*)

   Operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi ini banyak digunakan dalam bidang medis, misalnya foto rontgen dengan sinar X digunakan untuk membentuk ulang gambar organ tubuh.

Operasi-operasi pada pengolahan citra diterapkan pada citra bila :

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra.
2. Elemen di dalam citra perlu dikelompokkan, dicocokkan, atau diukur.
3. Sebagian citra perlu digabung dengan bagian citra yang lain.

2.3. Format Citra

2.3.1. Komponen Citra Digital

Setiap citra digital memiliki beberapa karakteristik, antara lain ukuran citra, resolusi, dan format nilainya. Umumnya citra digital berbentuk persegi panjang yang memiliki lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau piksel, sehingga ukuran citra selalu bernilai bulat.

Ukuran citra dapat juga dinyatakan secara fisik dalam satuan panjang (misalnya inch dan mm). Dalam hal ini tentu saja harus ada hubungan antara ukuran titik penyusun citra dengan satuan panjang. Hal tersebut dinyatakan dengan resolusi yang merupakan ukuran banyaknya titik untuk setiap satuan panjang. Biasanya satuan yang digunakan adalah dpi (*dot per inch*). Makin besar resolusi makin banyak titik yang terkandung dalam citra dengan ukuran fisik yang sama. Hal ini memberikan efek penampakan citra menjadi semakin halus.

Format citra digital ada bermacam-macam. Karena sebenarnya citra merepresentasikan informasi tertentu, sedangkan informasi tersebut dinyatakan secara bervariasi, maka citra yang mewakilinya dapat muncul dalam berbagai format. Citra yang merepresentasikan informasi yang bersifat biner untuk membedakan 2 keadaan tentu tidak sama dengan informasi citra yang lebih

kompleks sehingga memerlukan lebih banyak keadaan yang diwakilinya. Pada citra digital semua informasi tadi disimpan dalam bentuk angka, sedangkan peanampilan angka tersebut biasanya dikaitkan dengan warna[ACH05a].

Citra digital tersusun atas titik-titik yang biasanya berbentuk persegi panjang atau bujursangkar yang secara beraturan membentuk baris-baris dan kolom-kolom. Setiap titik memiliki koordinat sesuai dengan posisinya dalam citra. Koordinat ini biasanya dinyatakan dalam bilangan bulat positif, yang dapat dimulai dari 0 atau 1 bergantung pada sistem yang digunakan. Setiap titik juga memiliki nilai berupa angka digital yang merepresentasikan informasi yang diwakili titik tersebut. Format nilai piksel sama dengan format citra keseluruhan.

2.3.2. Representasi Citra Digital

Pada tahap analisis citra, setelah segmentasi akan menghasilkan *raw data* (data mentah) dalam bentuk kumpulan piksel yang termasuk sebagai batas wilayah atau piksel yang termasuk ke dalam suatu *region*. Data tersebut umumnya dapat digunakan secara langsung untuk kepentingan deskripsi objek. Namun demikian, untuk kepentingan deskripsi lebih baik, maka *raw data* tersebut diproses terlebih dahulu sehingga ukuran datanya menjadi lebih kecil.

Representasi dalam hal ini membuat suatu keputusan apakah suatu data harus direpresentasikan sebagai batas (*boundary*) atau suatu wilayah yang lengkap (*complete region*). Teknik representasi digunakan agar data menjadi lebih bermanfaat untuk proses *image* selanjutnya (deskripsi). Deskripsi dari suatu *region* tergantung dari representasi yang dipilih [PRA05].

Representasi suatu *region* dapat dilakukan dengan 2 cara :

1. *External characteristic*

   Representasi ini memfokuskan pada karakteristik bentuk eksternal (sifat *boundary*), misal sudut dan belokan.

2. *Internal characteristic*

   Representasi ini memfokuskan pada sifat internal *region*al (sifat *region*), seperti warna, tekstur, bentuk rangka (*skeleton*).

Komputer dapat mengolah isyarat-isyarat elektronik digital yang merupakan kumpulan sinyal biner (bernilai dua : 0 dan 1). Untuk itu, citra digital harus mempunyai format tertentu yang sesuai sehingga dapat merepresentasikan objek pencitraan dalam bentuk kombinasi data biner.

Pada kebanyakan kasus, terutama untuk keperluan penampilan secara visual, nilai data digital tersebut merepresentasikan warna dari citra yang diolah, dengan demikian format data citra digital berhubungan erat dengan warna. Format citra digital yang banyak dipakai adalah citra biner, skala keabuan, warna dan warna berindeks.

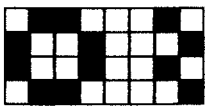
2.3.2.1. Citra Biner

Citra biner adalah citra yang hanya terdiri dari dua buah nilai yaitu nilai *high* (255) dan *low* (0). Walaupun citra ini tampak kurang menarik dibandingkan citra berwarna karena hanya terdiri dari warna hitam dan putih saja, namun dalam pengolahan citra seringkali digunakan citra biner untuk membantu menyederhanakan proses pengolahan citra[MUN04].

Citra biner adalah citra yang hanya mempunyai dua nilai derajat keabuan : hitam dan putih. Piksel-piksel objek bernilai 1 dan piksel-piksel latar belakang bernilai 0. pada waktu menampilkan gambar, 0 adalah putih dan 1 adalah hitam. Jadi, pada citra biner, latar belakang berwarna putih sedangkan objek berwarna hitam.

Pada standar citra untuk ditampilkan di layar komputer, nilai biner ini berhubungan dengan adanya tidaknya cahaya yang ditembakkan oleh *electron gun* yang terdapat di dalam monitor komputer. Angka 0 menyatakan tidak ada cahaya, dengan demikian warna yang direpresentasikan adalah hitam. Untuk angka 1 terdapat cahaya, sehingga warna yang direpresentasikan adalah putih. Standar tersebut disebut sebagai standar citra cahaya, sedangkan standar citra cat/tinta adalah berkebalikan, karena nilai biner tersebut menyatakan ada tidaknya tinta[ACH05a].

Data digital sering dinyatakan dalam bentuk bilangan heksadesimal dan pada bahasa Pascal (Delphi) diberi simbol $ didepannya. Angka 8 bit (1 byte) dapat ditulis dalam 2 digit/karakter heksadesimal.



Gambar 3 : Citra Biner dan representasinya dalam data digital

Alasan penggunaan citra biner adalah karena memiliki keuntungan sebagai berikut :

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

2.3.2.2. Citra *Greyscale*

Citra skala keabuan memberi kemungkinan warna yang lebih banyak daripada citra biner, karena ada nilai-nilai lain di antara nilai minimum (biasanya = 0) dan nilai maksimumnya. Banyaknya kemungkinan nilai dan nilai maksimumnya bergantung pada jumlah bit yang digunakan. Contohnya untuk skala keabuan 4 bit, maka jumlah kemungkinan nilainya adalah $2^4 = 16$, dan nilai maksimumnya adalah $2^4-1 = 15$, sedangkan untuk skala keabuan 8 bit, maka jumlah kemungkinan nilainya adalah $2^8 = 256$, dan nilai maksimumnya adalah $2^8-1 = 255$.

Format citra ini disebut skala keabuan karena pada umumnya warna yang dipakai adalah antara warna hitam sebagai warna minimal dan warna putih sebagai warna maksimalnya, sehingga warna antaranya adalah abu-abu.

2.3.2.3. Citra Warna

Pada citra warna, setiap titik mempunyai warna yang spesifik yang merupakan kombinasi dari 3 warna dasar, yaitu : merah, hijau, dan biru. Format citra ini sering disebut sebagai citra RGB. Setiap warna dasar mempunyai intensitas sendiri dengan nilai maksimum 255 (8 bit), misalnya warna kuning merupakan kombinasi warna merah dan hijau sehingga nilai RGB-nya adalah 255 255 0. Dengan demikian setiap titik pada citra warna membutuhkan data 3 byte.

Jumlah kombinasi warna yang mungkin untuk format citra ini adalah 224 atau lebih dari 16 juta warna, dengan demikian bisa dianggap mencakup semua warna yang ada, inilah sebabnya format ini dinamakan *true color*.

2.4. Matriks

Matriks adalah struktur penyimpanan data dalam memori utama yang setiap individu elemennya diacu dengan 2 buah indeks yang biasanya dikonotasikan dengan baris dan kolom. Jika indeks baris dinyatakan dengan i dan indeks kolom dinyatakan dengan j, maka notasi algoritmik untuk mengacu pada elemen pada baris dan kolom adalah Nama_matriks[i,j]. Jumlah baris dan kolom disebut juga ukuran matriks. Ukuran matriks digunakan sebagai penomoran indeks baris dan indeks kolom, mulai dari indeks terendah sampai indeks tertinggi. Contoh :

Matriks A :

A[1,1]	A[1,2]	A[1,3]
A[2,1]	A[2,2]	A[2,3]
A[3,1]	A[3,2]	A[3,3]

2.5. Piksel

Salah satu komponen dari citra yang menentukan resolusi dari citra tersebut adalah piksel, misalnya sebuah citra dikatakan resolusinya sebesar 800 x 600 maka berarti panjang piksel horisontalnya 800 dan panjang piksel vertikalnya 600 dan jumlah total keseluruhan piksel dari gambar tersebut yaitu 480000 atau dapat dikatakan bahwa gambar tersebut terdiri dari 480000 piksel.

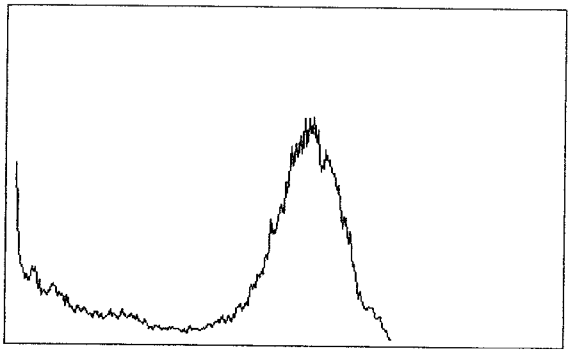
Piksel (*Picture Element*) adalah representasi sebuah titik terkecil dalam sebuah citra/*image* grafis. Biasanya gambar disimpan dalam komputer dalam bentuk piksel. Di mana piksel ini terdapat informasi tentang warna dan keterangan gambar. Editor gambar dapat mengubah piksel ini untuk memperbagus gambar dalam banyak cara. Piksel-piksel ini dapat diubah dalam grup, atau satuan, oleh algoritma rumit dalam editor gambar. Untuk menampilkan citra bitmap pada monitor atau mencetaknya pada *print*er, komputer menterjemahkan bitmap menjadi piksel (pada layar) atau titik tinta (pada *print*er)

2.6. Histogram

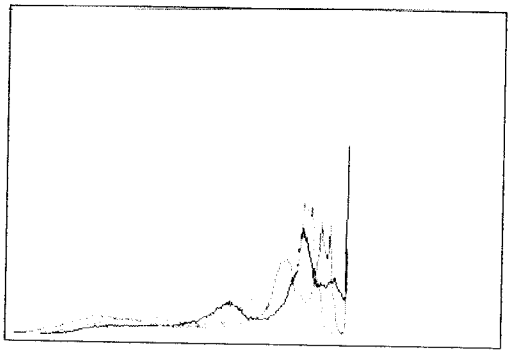
Salah satu alat bantu yang paling sederhana dan sangat berguna dalam pengolahan citra digital adalah histogram tingkat keabuan. Informasi suatu citra seringkali dapat diwakili oleh histogram ini. Komputasi histogram sangat sederhana dan cepat, serta dapat dilakukan pada saat suatu citra dipindahkan ke tempat lain (misalnya pada saat dibaca dari file)[ACH05a].

2.6.1. Pengertian Histogram Tingkat Keabuan

Histogram tingkat keabuan adalah suatu fungsi yang menunjukkan jumlah titik yang ada di dalam suatu citra untuk setiap tingkat keabuan. Absis (sumbu x)-nya adalah tingkat keabuan, dan ordinat (sumbu y)-nya adalah frekuensi kemunculan atau banyaknya titik dengan nilai keabuan tertentu. Gambar menunjukkan contoh histogram dari sebuah citra yang berformat citra skala keabuan 8 bit. Untuk citra warna, dapat pula dibuat histogram untuk masing-masing warna dasar : merah, hijau, dan biru.



Gambar 4 : Contoh histogram sebuah citra skala keabuan



Gambar 5 : Contoh histogram sebuah citra true color

2.6.2. Kegunaan Histogram

Histogram mempunyai banyak manfaat pada pengolahan citra, diantaranya untuk :

1. Penentuan parameter digitasi

   Histogram dapat digunakan sebagai indikasi visual untuk menentukan apakah suatu citra sudah berada dalam jangkau yang tepat dalam skala keabuan. Biasanya, suatu citra digital sebisa mungkin menggunakan seluruh tingkat keabuan, mulai dari nilai minimumnya sampai nilai maksimumnya, pada saat proses pencitraan. Jika tidak, maka resolusi skala keabuan dari citra tersebut menjadi jelek, dengan demikian parameter digitasinya perlu diatur lagi. Sebaliknya, jika objek pencitraan mempunyai jangkauan kecemerlangan lebih lebar dari pada yang dapat ditangani oleh perangkat pencitraan, maka histogram akan terlihat terpotong pada sisi kiri dan/atau kanannya.

2. Pemilihan batas ambang

   Salah satu teknik pengolahan citra adalah pengambangan (*thresholding*), yang diantaranya digunakan untuk menonjolkan citra suatu objek dari latar belakangnya. Ambil contoh sebuah citra dari suatu objek terang dengan latar belakang gelap. Histogram dari citra tersebut memiliki 2 buah puncak.

Titik terang dalam objek membentuk puncak sebelah kanan, sedangkan puncak sebelah kiri ditimbulkan oleh titik-titik gelap pada latar belakang. Titik-titik dengan skala keabuan sedang di sekitar tepi objek membentuk bagian lembah antara kedua puncak tersebut. Pemilihan batas yang optimal untuk operasi pengambangan dilakukan pada daerah lembah ini.

2.7. Segmentasi Citra

Proses awal dalam menganalisis objek di dalam citra biner adalah segmentasi objek. Proses segmentasi bertujuan mengelompokkan piksel-piksel objek menjadi wilayah (*region*) yang merepresentasikan objek[MUN05].

Citra biner didapat dari hasil segmentasi citra abu-abu atau citra warna melalui operasi binerisasi. Jika nilai intensitas suatu objek berada dalam suatu interval dan nilai intensitas latar belakangnya berada di luar interval, maka citra biner dapat diperoleh dengan mudah melalui operasi binerisasi. Jadi, untuk menghasilkan citra biner, operasi segmentasi dan binerisasi adalah identik.

Konversi suatu citra abu-abu atau warna menjadi citra biner adalah bentuk sederhana dari segmentasi citra, di mana citra dipartisi menjadi dua bagian.

2.7.1. Binerisasi dan Pengambangan

Proses binerisasi mengubah citra berformat 24 bit dan 8 bit menjadi citra biner (format 1 bit). Sedangkan proses pengambangan adalah mengelompokkan nilai derajat keabuan setiap piksel ke dalam 2 kelas, hitam dan putih.

Prinsip kerja dari metode ini adalah dengan mengurangi intensitas cahaya warna yang ada dalam suatu citra. Operasi pengambangan digunakan untuk mengubah kadar keabuan dari sebuah citra *non*-biner, yang mempunyai kemungkinan nilai lebih dari 2, ke citra biner, yang hanya memiliki 2 buah nilai (0 atau 1). Dalam hal ini, titik dengan nilai rentang nilai keabuan tertentu diubah menjadi berwarna hitam dan sisanya menjadi putih atau sebaliknya.

2.7.2. Konversi Citra *Greysacle* dan *True Color* menjadi Citra Biner

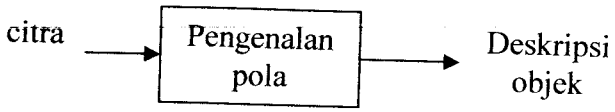
Pengkonversian citra *greyscale*/warna menjadi citra biner dilakukan untuk alasan-alasan :

1. Untuk mengidentifikasi keberadaan objek, yang direpresentasikan sebagai daerah (*region*) di dalam citra. Misalnya kita ingin memisahkan (segmentasi) objek dari gambar latar belakangnya. Piksel-piksel objek dinyatakan dengan nilai 1 sedangkan piksel lainnya dengan 0. Objek ditampilkan seperti gambar siluet. Untuk memperoleh siluet yang bagus, objek harus dapat dipisahkan dengan mudah dari gambar latar belakangnya.
2. Untuk lebih memfokuskan pada analisis bentuk *morphology*, yang dalam hal ini intensitas piksel tidak terlalu penting dibandingkan bentuknya. Setelah objek dipisahkan dari latar belakangnya, properti geometri dan *morphology*/topologi objek dapat dihitung dari citra biner. Hal ini berguna untuk pengambilan keputusan.
3. Untuk menampilkan citra pada piranti keluaran yang hanya mempunyai resolusi intensitas 1 bit, yaitu piranti penampil dua-warna (biner) seperti pencetak (*printe*r).
4. Mengkonversi citra yang telah ditingkatkan kualitas tepinya (*edge enhancement*) ke penggambaran garis-garis tepi. Hal ini diperlukan untuk membedakan tepi yang kuat yang berkoresponden dengan batas-batas objek dengan tepi lemah yang berkoresponden dengan perubahan *illumination*, bayangan, dll.

2.8. Pengenalan Pola

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

Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut, dan memberikan keluaran berupa deskripsi objek di dalam citra.



Gambar 6 : Proses Pengenalan Pola

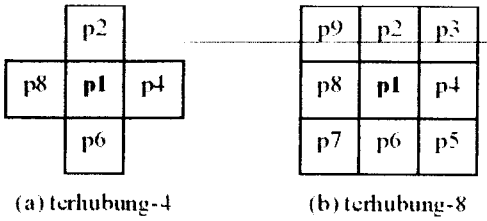
Contoh pengenalan pola misalnya gambar dibawah ini, tulisan tangan yang digunakan sebagai data masukan untuk mengenali karakter 'A'. Dengan menggunakan suatu algoritma pengenalan pola, diharapkan komputer dapat mengenali bahwa karakter tersebut adalah 'A'.

Gambar 7 : Citra karakter 'A' sebagai masukan untuk pengenalan huruf

2.8.1. Operasi Morphology

Representasi citra erat kaitannya dengan *morphology* citra. Operasi *morphology* merupakan teknik pengolahan citra yang didasarkan pada bentuk segmen atau *region* dalam citra. Biasanya segmen tadi didasarkan pada objek yang menjadi perhatian. Proses pembagian citra dilakukan dengan membedakan antara objek dan latar, antara lain dengan memanfaatkan operasi pengambangan yang mengubah citra warna dan skala keabuan menjadi citra biner. Nilai biner dari citra hasil merepresentasikan dua keadaan yaitu objek dan latar. Operasi ini biasanya diterapkan pada citra biner karena difokuskan pada bentuk objek. Meskipun demikian, operasi *morphology* sering pula digunakan pada citra skala keabuan dan warna.

Operasi *morphology* biasanya didasarkan pada nilai-nilai tetangga langsung di sekeliling titik obyek yang ditinjau (*Moore neighborhood*). Untuk operasi terhubung-4 (4-*connected*) maka tetangga yang diperhatikan hanya yang langsung bersebelahan, yaitu titik di sebelah kiri, kanan, atas dan bawah, sedangkan untuk operasi terhubung-8 (8-*connected*) tetangga diagonalnya diikutsertakan [ACH05b].



(a) terhubung-4 (b) terhubung-8

Gambar 8 : Moore *neighborhood*

Beberapa definisi yang dipakai dalam operasi *morphology*:

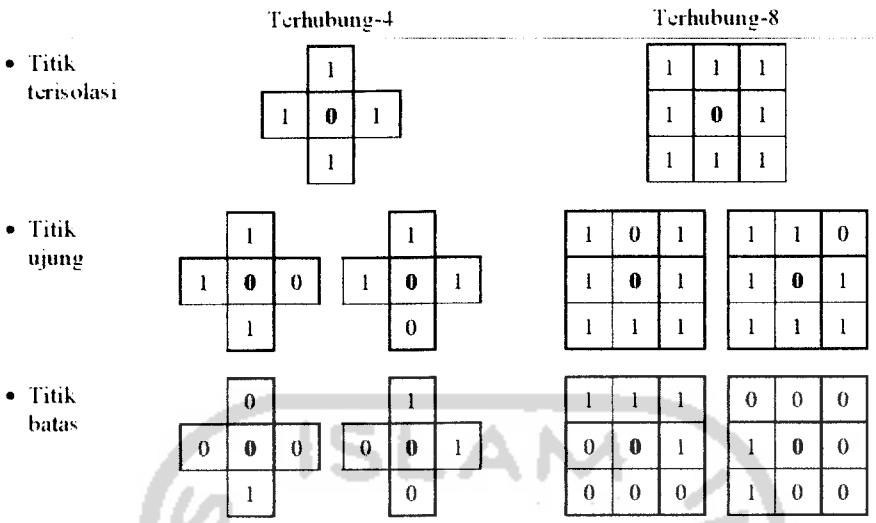
1. Titik obyek, adalah titik yang merupakan bagian dari obyek (p1 = obyek).
2. Titik latar, adalah titik yang merupakan bagian dari latar (p1 = latar).

   B(p1) = banyaknya tetangga dari p1 yang merupakan titik obyek.

   A(p1) = banyaknya pola "latar, obyek" untuk urutan p2-p4-p6-p8-p2 pada operasi terhubung-4 atau urutan p2-p3-p4-p5-p6-p7-p8-p9-p2 pada operasi terhubung-8.
3. Titik terisolasi, adalah titik obyek yang semua tetangganya adalah titik latar.

   B(p1) = 0.
4. Titik ujung, adalah titik obyek yang mempunyai tepat sebuah tetangga yang merupakan titik obyek juga. B(p1) = 1.
5. Titik batas, adalah titik obyek yang salah satu atau lebih tetangganya adalah titik latar. B(p1) < 4 pada operasi terhubung-4 dan B(p1) < 8 pada operasi terhubung-8. Apabila semua titik tetangganya adalah titik obyek maka dapat dipastikan titik tersebut berada di dalam obyek (bukan titik batas).
6. Titik simpel, adalah titik obyek yang jika diubah menjadi titik latar maka tidak mengubah kondisi hubungan antar titik-titik obyek tetangganya.

Sebagai ilustrasi, lihat contoh bagian dari citra di bawah ini dengan menggunakan nilai 0 untuk titik obyek dan 1 untuk titik latar.



Gambar 9 : Contoh Titik Terisolasi, Titik Ujung, dan Titik Batas

Operasi *morphology* pada dasarnya dilakukan secara serempak untuk semua titik dalam citra, sementara pada implementasinya operasi ini menggunakan program komputer dan harus dilakukan secara sekuensial, yaitu titik-demi-titik. Oleh karena itu, urutan titik mana yang hendak dioperasikan dahulu, haruslah tidak mempengaruhi hasil akhir. Kita bisa mulai dari titik paling kiri-atas ke arah kanan dan kemudian ke bawah, atau sebaliknya dari titik yang berada pada posisi paling bawah-kanan ke arah atas lalu ke kiri.

2.8.2. *Skeleton*

Skeleton memberikan representasi simpel dari suatu objek dengan jumlah piksel yang kecil dan tetap mempertahankan karakteristik ukuran, posisi dan topologi dari bentuk aslinya. Sehingga objek dari sebuah citra dapat dikenali dan diekstrak untuk proses pengolahan citra selanjutnya. Dengan kata lain, setelah sebagian besar titik pada obyek tersebut dihilangkan, maka pola dari obyek tersebut harus tetap dapat dikenali. Pola yang tertinggal ini disebut sebagai kerangka (*skeleton*), di mana sifat-sifatnya adalah :

1. Ketipisan : kerangka obyek berukuran setipis mungkin (1 atau 2 titik).

2. Konektivitas : kerangka dari suatu obyek terhubung satu sama lain sesuai dengan topologi pola aslinya.
3. Posisi : letak kerangka berada tepat di tengah obyek.
4. Stabilitas : setelah suatu bagian kerangka diperoleh, maka bagian tersebut tidak akan terkikis lagi oleh operasi pengikisan berikutnya.

Kebanyakan teknik untuk mengekstrak *skeleton* merupakan dasar dari operasi *thinning*.

2.8.3. *Thinning*

Penipisan citra (*thinning*) merupakan operasi pemrosesan reduksi citra biner yang dalam hal ini objek (*region*) menjadi rangka (*skeleton*) yang menghampiri garis sumbu objek. Tujuannya adalah mengurangi bagian yang tidak perlu (*redundant*) sehingga dihasilkan informasi yang esensial saja. Pola penipisan harus tetap mempunyai bentuk yang menyerupai pola asalnya [MUN04].

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

Kebanyakan prosedur *thinning* berulang kali memindahkan unsur-unsur batas sampai suatu piksel menata ketebalan yang maksimal satu atau dua yang ditemukan. Operasi *thinning* mempunyai persyaratan tertentu yang harus dipenuhi, sebab tidak semua macam bentuk akan direduksi menjadi sebuah titik. Jadi syarat-syarat operasi *thinning* sebenarnya untuk mengatur arah penghapusan

bagian objek sehingga setiap bentuk objek akan mempunyai hasil yang unik dan dapat dijadikan sebagai ciri pembeda dari bentuk-bentuk lainnya melalui bentuk *skeleton*nya. Syarat-syarat operasi *thinning* adalah sebagai berikut :

1. Daerah harus diubah menjadi struktur garis yang terkoneksi.
2. Hasil *thinning* minimal harus 8 lintasan.
3. Ujung-ujung garis harus tetap dipertahankan.
4. Hasil *thinning* harus kira-kira sama dengan garis tengah.
5. Cabang-cabang pendek disebabkan oleh *thinning* harus dieliminasi.

Cara umum yang digunakan dalam operasi *thinning* adalah dengan memeriksa setiap piksel di dalam citra dalam kaitannya dengan *neighborhood. region* minimal untuk satu kelompok *neighborhood* berukuran 3x3 piksel, dan " mengupas " batas daerah citra satu per satu piksel, sampai menjadi baris. Proses ini dilakukan berulang-ulang, pada tiap iterasi, setiap piksel diperiksa dalam kelompok piksel berukuran n x n, dan batas setebal satu piksel yang tidak digunakan untuk mempertahankan koneksi atau tidak pada posisi di ujung garis dihapus. Iterasi akan berakhir bila tidak ada perubahan yang terjadi (tidak ada lagi aksi penghapusan) pada objek yang sedang dikenakan operasi. Terdapat banyak algoritma untuk *image thinning* dengan tingkat kompleksitas, efisiensi, dan akurasi yang berbeda-beda.

Citra yang digunakan adalah citra biner, yaitu citra yang hanya memiliki 2 kemungkinan nilai pada setiap piksel-pikselnya, yaitu 0 atau 1. Nilai 0 adalah *background points*, yang bukan merupakan bagian dari citra sesungguhnya. Sedangkan nilai 1 adalah *region points*, yaitu bagian dari citra sebenarnya (bukan latar belakang). Citra hasil dari *thinning* biasanya disebut *skeleton*.

2.8.3.1. Metode *Thinning*

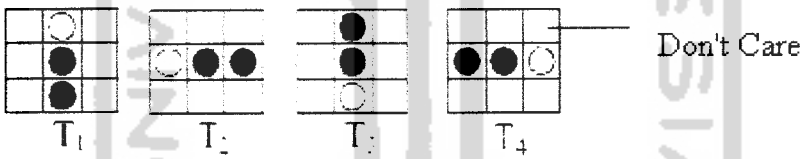
Algoritma *thinning* dapat dibagi menjadi 2 jenis, yaitu iteratif dan *non*-iteratif. Walaupun proses algoritma *thinning* yang *non*-iteratif lebih cepat dibandingkan dengan yang iteratif, tapi tidak selalu menghasilkan hasil yang bagus.

Algoritma bersifat iteratif berguna untuk mengikis lapisan piksel terluar sampai tidak ada lapisan lagi yang dapat dihilangkan. Kebanyakan algoritma iteratif *thinning* menggunakan *Template based mark-and-delete*. Metode *thinning* jenis ini menggunakan template untuk dicocokkan dengan citra yang akan di-*thinning*, di mana jika suatu kelompok piksel dari sebuah *image* cocok dengan template, maka piksel tengahnya akan di-*delete*. Metode ini cukup terkenal karena *reliability* dan keefektifannya[NN02]. Salah satu algoritma *thinning* yang bersifat iteratif ini adalah metode *Stentiford* (F.W.M.Stentiford and R.G.Mortimer, 1983). Yang merupakan metode yang dipilih oleh penulis dalam pengerjaan penelitian ini.

2.8.3.2. Karakteristik Algoritma *Stentiford*

2.8.3.2.1. *Template based mark-and-delete* Algorithm

Algoritma *Stentiford* menggunakan template untuk memindai piksel-piksel yang terdapat dalam sebuah citra. Template yang dipakai dalam algoritma *Stentiford* ini adalah empat buah template 3x3, yaitu sebagai berikut [NN02]:



Gambar 10 : Template yang dipakai dalam algoritma *Stentiford*

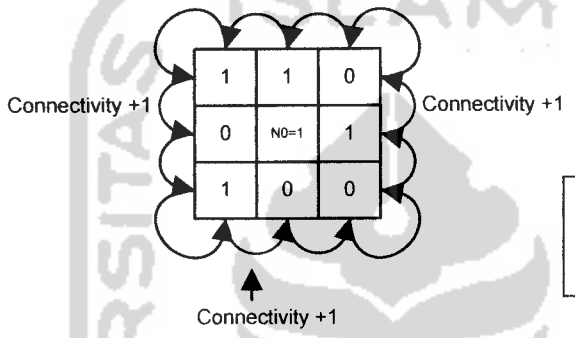
Kotak putih yang kosong merupakan tempat piksel yang tidak perlu diperhatikan warnanya.

2.8.3.2.2. *Connectivity Number*

Connectivity number merupakan sebuah ukuran berapa banyak objek yang terhubung dengan piksel tertentu. Maksudnya adalah jumlah perpindahan piksel tetangga yang bernilai 1 ke piksel tetangganya yang bernilai 0. Perhitungan nilai *connectivity* dimulai dari piksel tetangga yang pertama (N_1), kemudian dilanjutkan hingga piksel tetangga kedelapan (N_8), bergerak berlawanan arah jarum jam.

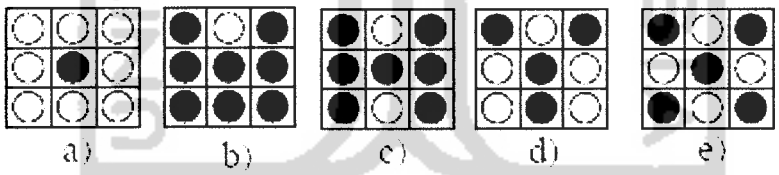
Misalkan : Jika N_1 bernilai 1 dan N_2 bernilai 0, maka nilai *connectivity* dari piksel N_0 ditambah 1 (C_0 + 1), dan seterusnya untuk piksel tetangga yang lain.

N_4	N_3	N_2
N_5	N_0	N_1
N_6	N_7	N_8



Jadi, nilai *connectivity* dari piksel $N_0 = 3$.

Contoh 2 :



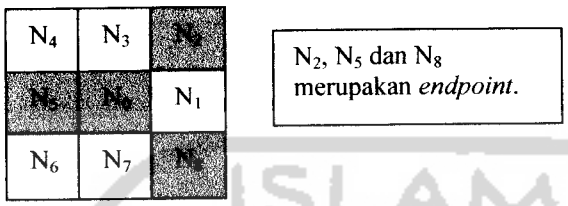
Keterangan :

a) Merepresentasikan *connectivity number* = 0,
b) Merepresentasikan *connectivity number* = 1, piksel central akan dihapus tanpa mempengaruhi *connectivity number* antara kiri dan kanan
c) Merepresentasikan *connectivity number* = 2, penghapusan piksel central tidak menyambungkan kedua sisi
d) Merepresentasikan *connectivity number* = 3
e) Merepresentasikan *connectivity number* = 4

2.8.3.2.3. *Endpoint* Piksel

Endpoint piksel adalah piksel yang merupakan batas akhir dan hanya terhubung dengan 1 piksel saja. Contoh : suatu piksel hitam hanya mempunyai satu tetangga saja yang hitam dari kemungkinan delapan tetangganya.

Contoh :



2.8.3.2.4. Algoritma

Langkah-langkah *thinning* dengan metode *Stentiford* adalah sebagai berikut [HAR03] :

1. Cari lokasi piksel (i, j), di mana *neighbour* pada *image* cocok dengan template T1.
2. Jika piksel (i, j) bukan merupakan *endpoint* (jika piksel (i, j) memiliki lebih dari satu *neighbour* di luar kedelapan *neighbour*nya) dan memiliki *connectivity* number = 1, lalu tandai piksel tersebut untuk dihapus.
3. Ulangi langkah 1 dan 2, pindai dari kiri ke kanan, atas ke bawah (untuk menghapus piksel batas yang atas).
4. Ulangi langkah 1, 2, dan 3 untuk T2, pindai dari bawah ke atas, kiri ke kanan.
5. Ulangi langkah 1, 2, dan 3 untuk T3, pindai dari kanan ke kiri, bawah ke atas.
6. Ulangi langkah 1, 2, dan 3 untuk T4, pindai dari atas ke bawah, kanan ke kiri.
7. Jika piksel-piksel sudah ditandai untuk dihapus, hapuslah.
8. Jika piksel-piksel tersebut sudah dihapus, ulangi dari langkah 1, jika pindai belum berhenti.

Setelah proses di atas dilakukan, kemungkinan akan terdapat artifak pada citra hasil *thinning*. Artifak-artifak yang mungkin terjadi adalah seperti :

necking tailing projection

Gambar 11 : Artifak-artifak yang terjadi pada saat proses *thinning*

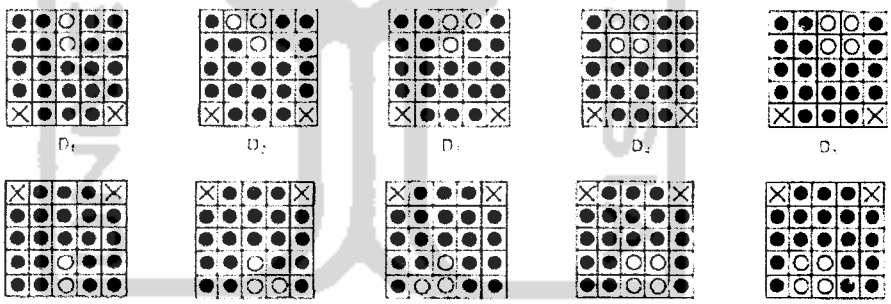
Untuk mengurangi artifak yang terjadi (*necking, tailing*, dan citra hasil yang buruk) akibat proses *thinning*, *Stentiford* memberikan beberapa penambahan pada algoritmanya, antara lain [RUP00] :

a. Untuk menghaluskan gambar.

   Hapus piksel utama yang memiliki tetangga bernilai 1 (hitam) kurang dari 3 piksel dan memiliki nilai *connectivity* kurang dari 2.

b. Untuk menghilangkan necking.

   Hapus piksel-piksel yang memenuhi template-template berikut :



2.9. Citra dalam Delphi

2.9.1. Komponen *TImage*

Delphi tidak menyediakan secara khusus rutin-rutin untuk pengolahan citra, oleh karena itu perlu dibuat sendiri program untuk mengolah citra. Namun delphi telah menyediakan sarana untuk menampilkan citra, yaitu komponen *Timage* yang terdapat pada palet komponen *Additional*.

Komponen ini memiliki properti *Picture* yang digunakan untuk menyimpan data citra. Citra yang akan ditampilkan diambil dari file gambar yang

ditentukan pada saat mendesain dengan cara mengisi nilai properti ini, atau pada saat program dijalankan dengan menggunakan prosedure *LoadFromFile*.

Subproperti yang penting pada Picture adalah :

- *Height*, berisi nilai tinggi citra
- *Width*, berisi nilai lebar citra
- Bitmap, berisi data format dan piksel citra

Dalam delphi, informasi format citra terdapat pada subproperti Bitmap, yaitu *PixelFormat*, dengan nilai :

Tabel 1 : Nilai Piksel/Format pada Bitmap

Nilai	Format Citra
pf1bit	Citra biner/monokrom
pf8bit	Citra skala keabuan
pf32bit	Citra *true color* (16 juta warna)

Dengan membaca nilai *PixelFormat* dapat diketahui cara penyimpanan piksel dalam memori sehingga mempermudah dalam pemrograman. Misalnya, untuk pf8bit, setiap piksel disimpan dalam ukuran 1 byte, sedangkan untuk pf24bit, setiap piksel disimpan dalam ukuran 3 byte yang masing-masing berisi nilai elemen merah (R), hijau (G), dan biru (B). Sebaliknya apabila nilai *PixelFormat* diubah, maka otomatis format citra akan berubah sesuai nilai yang baru.

BAB III
METODOLOGI

3.1. Analisis Kebutuhan

3.1.1. Metode Analisis

Untuk menganalisa kebutuhan dalam pembuatan aplikasi dari implementasi algoritma *Stentiford*, penyusun menggunakan metode analisis untuk mengetahui kebutuhan *input*, kebutuhan proses, dan kebutuhan *output* dari implementasi algoritma *Stentiford* ini.

Untuk itu, penyusun menganalisis segi kemanfaatan implementasi algoritma ini untuk memberikan gambaran yang lebih jelas terhadap kebutuhan *input*, kebutuhan proses, dan kebutuhan *output*. Di samping melakukan analisa pustaka untuk menentukan kemampuan pengimplementasian dari algoritma ini.

3.1.2. Hasil Analisis

Dari hasil analisis yang diperoleh dari metode analisis berarah objek, penyusun kemudian dapat menentukan kebutuhan *input*, kebutuhan proses dan kebutuhan *output* dari implementasi algoritma ini berdasarkan objek-objek yang terkait dalam perancangan sistem yang akan digunakan dalam pembangunan aplikasi implementasi algoritma *Stentiford*.

3.1.2.1. Analisis Kebutuhan *Input*

Input atau masukan yang dibutuhkan dalam aplikasi implementasi algoritma *Stentiford* ini adalah :

1. Sebuah citra digital yang berformat bitmap (*.bmp).
2. Citra digital bukan merupakan citra digital yang abstrak dan rusak.
3. Ukuran citra yang akan diolah maksimal 290 piksel x 290 piksel.

3.1.2.2. Analisis Kebutuhan Proses

Kebutuhan proses dalam aplikasi implementasi algoritma *Stentiford* ini adalah :

1. Proses Histogram.
2. Proses Binerisasi dan Pengambangan.
3. Proses *Stentiford.*

3.1.2.3. Analisis Kebutuhan *Output*

Data keluaran yang diperoleh dari proses aplikasi implementasi algoritma *Stentiford* adalah sebuah citra yang hanya tinggal setebal satu piksel (*skeleton*) dengan tetap mempertahankan karakteristik ukuran, posisi dan topologi dari bentuk aslinya. Sehingga objek dari sebuah citra dapat dikenali.

Citra yang sudah diolah disimpan dalam format bitmap (*.bmp) dan dapat didokumentasikan dalam bentuk *print*out.

3.1.2.4. Analisis Kebutuhan Antarmuka

Perancangan antarmuka untuk pembuatan aplikasi implementasi algoritma *Stentiford* dilakukan dengan menggunakan bahasa pemrograman Pascal yang terintegrasi pada media pemrograman Delphi.

3.1.2.5. Analisis Kebutuhan Perangkat Lunak

Perangkat keras komputer tidak berarti tanpa perangkat lunak begitu juga sebaliknya. Jadi perangkat lunak dan perangkat keras saling mendukung satu sama lain. Perangkat keras hanya berfungsi jika diberikan instruksi-intruksi kepadanya. Instruksi-instruksi inilah disebut dengan perangkat lunak.

Maka untuk pembuatan aplikasi implementasi algoritma *Stentiford* ini, digunakan bahasa pemrograman Pascal dan *compiler* Delphi. Delphi telah menyediakan sarana untuk menampilkan citra, yaitu komponen T*image* yang terdapat pada modul *Additional*. Komponen ini memiliki properti *Picture* yang digunakan untuk menyimpan data citra. Dalam delphi, informasi format citra terdapat pada subproperti Bitmap, yaitu *PixelFormat*.

3.1.2.6. Analisis Kebutuhan Perangkat Keras

Pada tahap ini perlu dipastikan apakah komputer yang digunakan cukup memadai atau tidak untuk menjalankan keseluruhan sistem. Spesifikasi minimal yang disarankan agar pengguna dapat menjalankan aplikasi ini dengan lancar adalah :

Tabel 2 : System Requirements Hardware

Hardware & Sistem Operasi	Rekomendasi
Processor	750 Mhz
Memory	256 Mb
Video Card	16 bit
Hard Disk	12 Gb
Sistem Operasi	Windows XP
Display Resolution	Minimum 1024x768 piksel

3.2. Perancangan Sistem

3.2.1. Metode Perancangan

Metode perancangan yang digunakan untuk membuat aplikasi implementasi algoritma *Stentiford* adalah metode perancangan berarah objek yang menggunakan flowchart untuk menggambarkan urutan-urutan proses sistem secara keseluruhan serta bagaimana objek-objek dalam sistem saling bekerjasama satu sama lain.

3.2.2. Hasil Perancangan

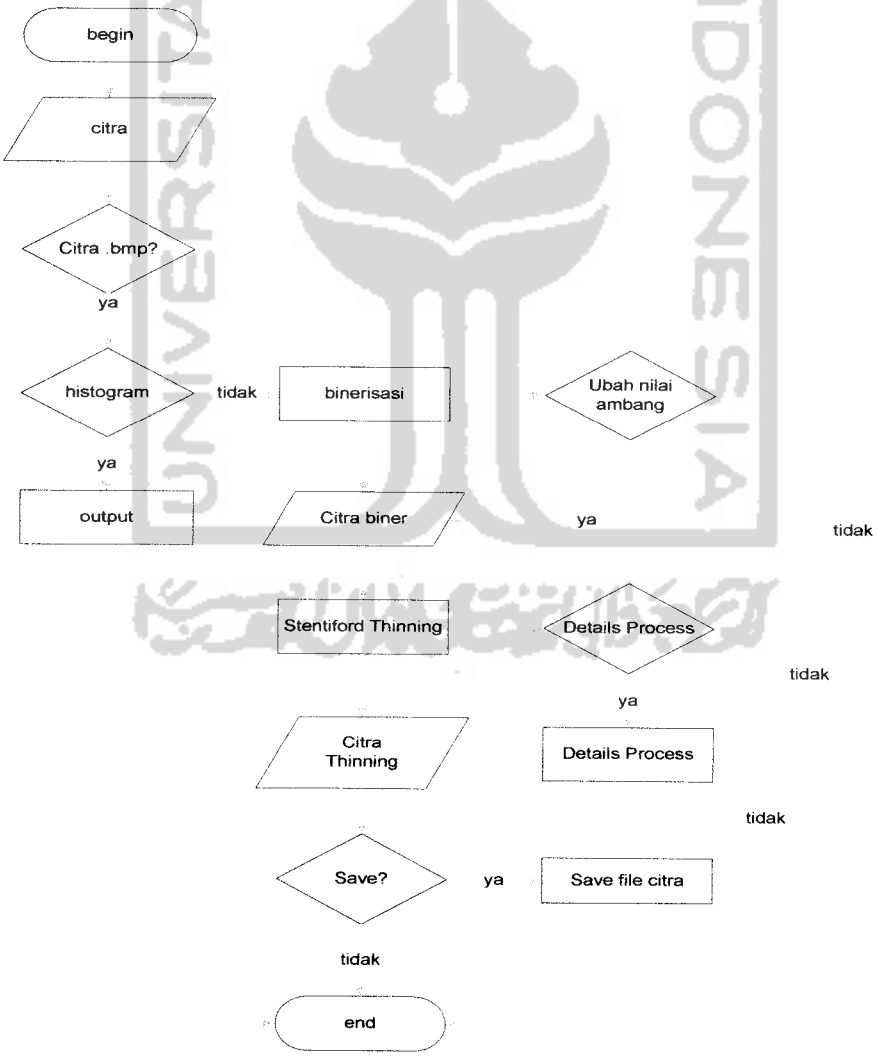
Berdasarkan analisis yang telah dilakukan maka dapat diketahui apa saja yang manjadi masukan sistem, keluaran sistem, metode yang digunakan oleh sistem, fungsi dan operasi yang ada dalam sistem, serta antarmuka sistem yang dibuat, sehingga sistem yang dibuat nantinya sesuai dengan apa yang diharapkan.

Perancangan sistem ini akan digambarkan kedalam beberapa diagram yaitu :

1. Flowchart Proses Umum
2. Flowchart Proses Histogram
3. Flowchart Proses Binerisasi dan Pengambangan
4. Flowchart Proses *Stentiford*
5. Perancangan Antarmuka

3.2.2.1. Flowchart Proses Umum

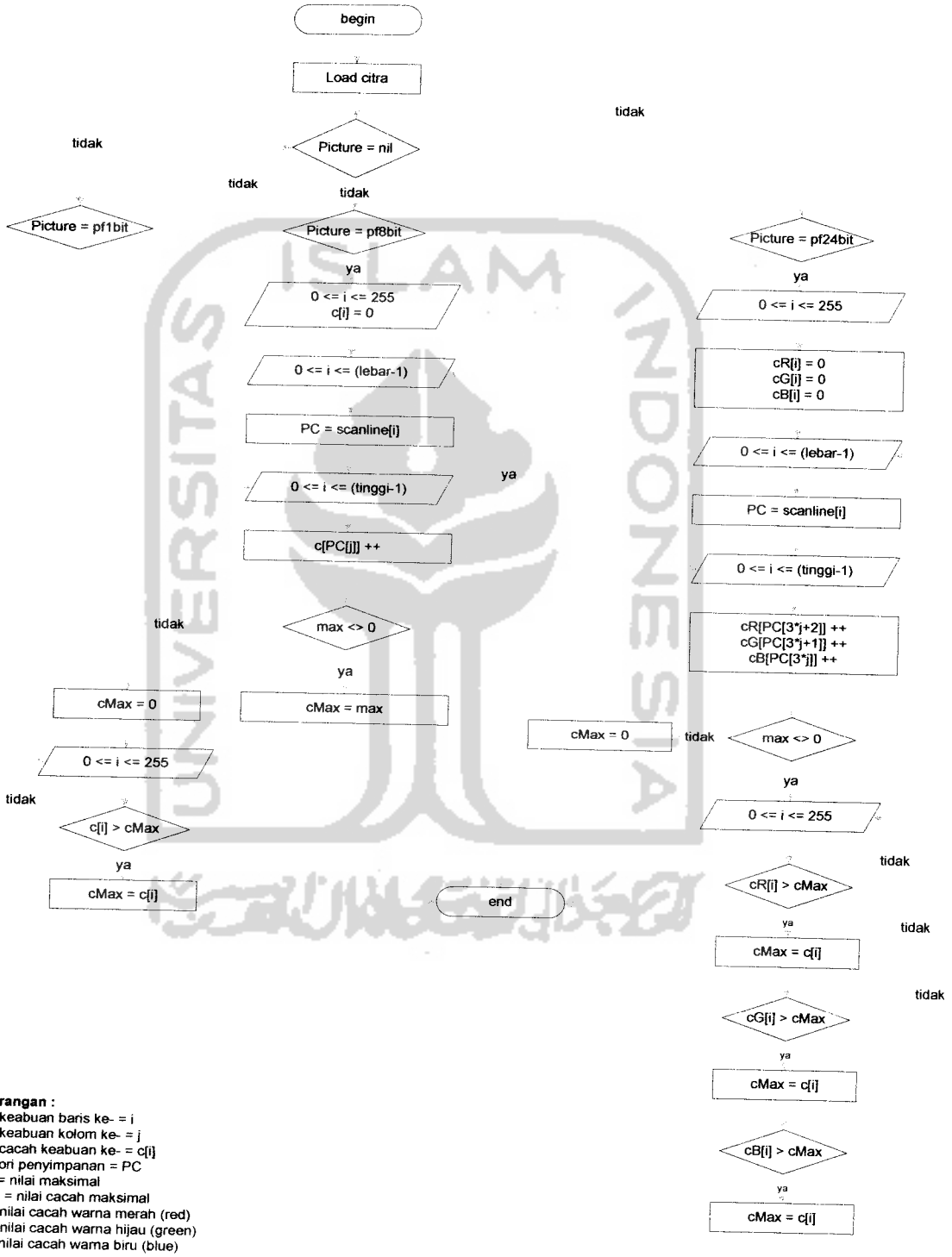
Flowchart ini menjelaskan urutan-urutan proses yang terjadi pada sistem aplikasi ini secara umum.



Flowchart 1 : Proses Umum

3.2.2.2. Flowchart Proses Histogram

Flowchart ini menunjukkan langkah-langkah dalam membuat histogram dari sebuah citra.



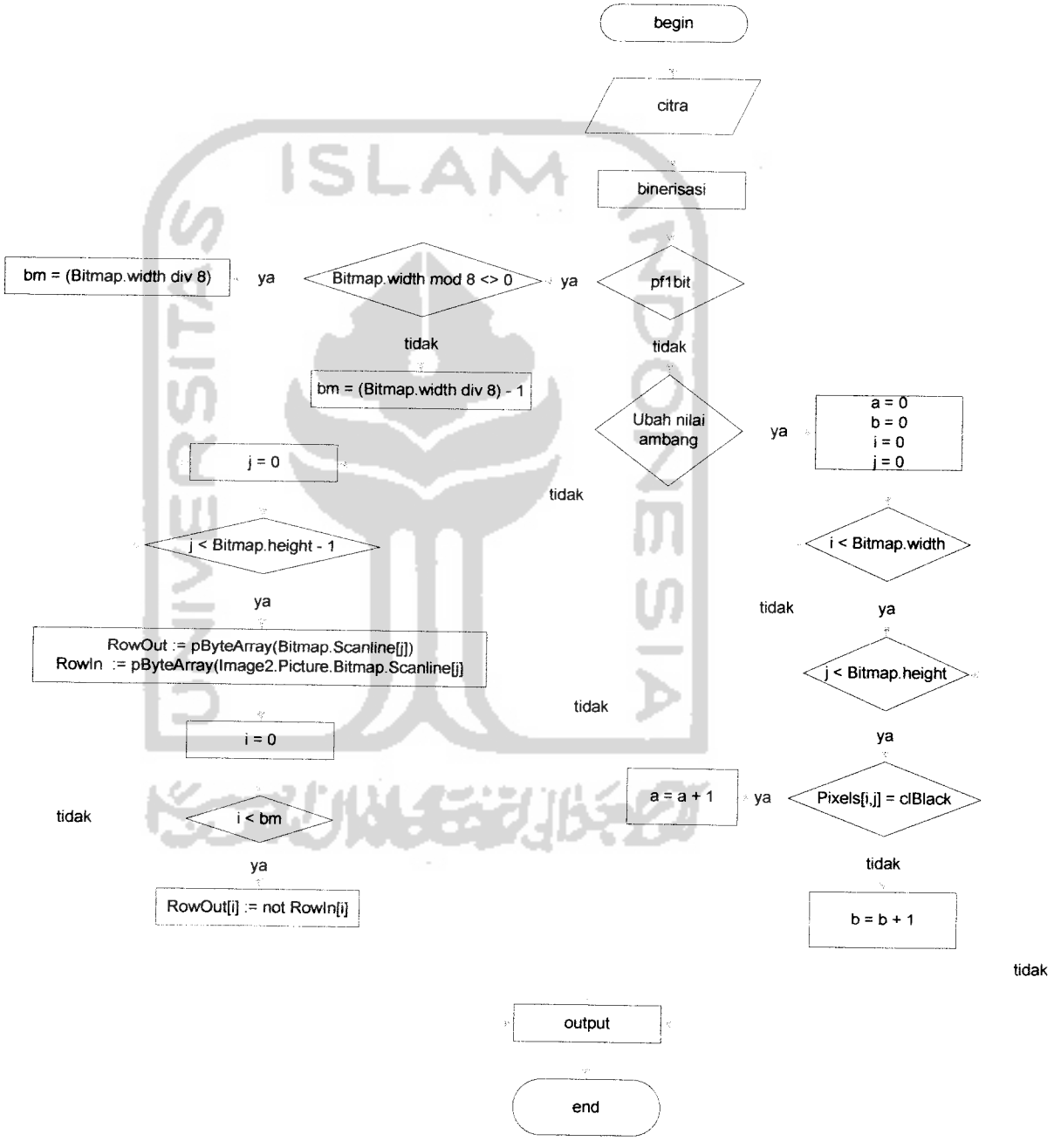
Keterangan :
Nilai keabuan baris ke- = i
Nilai keabuan kolom ke- = j
Nilai cacah keabuan ke- = c[i]
Memori penyimpanan = PC
Max = nilai maksimal
cMax = nilai cacah maksimal
cR = nilai cacah warna merah (red)
cG = nilai cacah warna hijau (green)
cB = nilai cacah warna biru (blue)

Flowchart 2 : Proses Histogram

3.2.2.3. Flowchart Proses Binerisasi dan Pengambangan

Flowchart ini menunjukkan langkah-langkah dalam proses binerisasi dari sebuah citra.

Flowchart proses binerisasi dan pengambangan ditunjukkan pada gambar sebagai berikut :



Flowchart 3 : Proses Binerisasi dan Pengambangan Citra

Pseudocode proses binerisasi adalah sebagai berikut :

```
1.  Bitmap.PikselFormat ← pf1bit
2.  pada Temp lakukan langkah 3 sampai 7
3.    Width  ← Bitmap.Width
4.    Height ← Bitmap.Height
5.    PikselFormat ← pf1bit
6.    jika RadioButton.Checked maka lakukan langkah 7
7.      Temp.Palette ← GetTwoColorPalette
8.  jika Temp.width mod 8 <> 0 maka lakukan langkah 9
9.    bm ← (Temp.width div 8)
10. jika sebaliknya maka lakukan langkah 11
11.   bm ← (Temp.width div 8)-1
12. untuk j ← 0 hingga Bitmap.Height-1 lakukan langkah 13 sampai 16
13.   RowOut ← pByteArray(Temp.Scanline[j])
14.   RowIn  ← pByteArray(Bitmap.Scanline[j])
15.     untuk i ← 0 hingga bm lakukan langkah 16
16.       RowOut[i] ← not RowIn[i]
17. Bitmap ← Temp
```

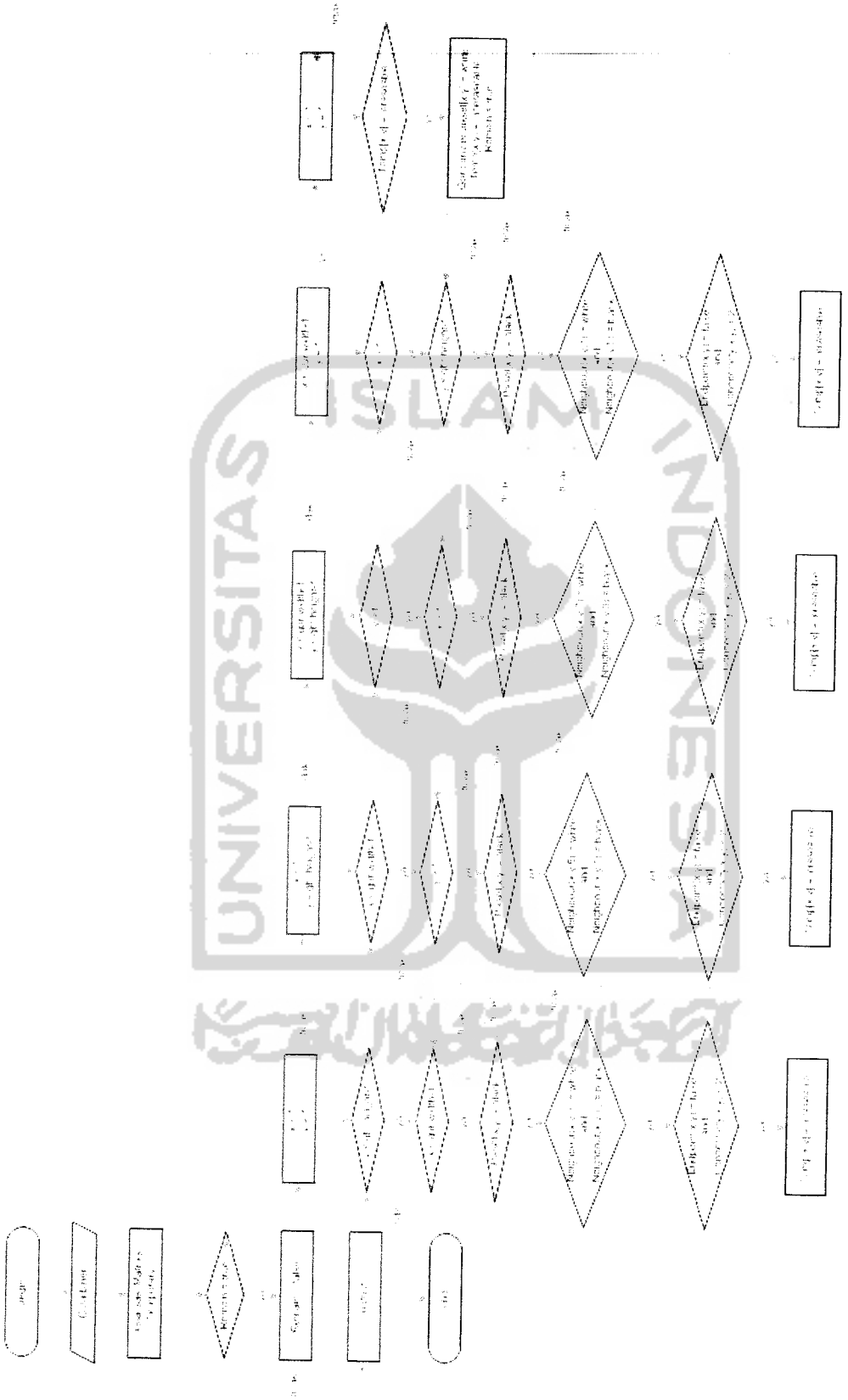
Pseudocode proses pengambangan adalah sebagai berikut :

```
1.  Ambang ← ScrollBar.Position
2.  jika (Bitmap.PikselFormat = pf8bit) maka lakukan langkah 3 sampai 10
3.    untuk i ← 0 hingga Bitmap.Height-1 lakukan langkah 4 sampai 10
4.      PC ← Bitmap.ScanLine[i]
5.      PH ← Bitmap.ScanLine[i]
6.        untuk j ← 0 hingga Bitmap.Width-1 lakukan langkah 7 sampai 10
7.          jika (PC[j] < Ambang) maka lakukan langkah 8
8.            PH[j] ← 0
9.          jika sebaliknya maka lakukan langkah 10
10.           PH[j] ← 255
11. jika (Bitmap.PikselFormat = pf24bit) maka lakukan langkah 12 sampai 24
12.   untuk i ← 0 hingga Bitmap.Height-1 lakukan langkah 13 sampai 24
13.     PC ← Bitmap.ScanLine[i]
14.     PH ← Bitmap.ScanLine[i]
15.       untuk j ← 0 hingga Bitmap.Width-1 lakukan langkah 16 sampai 24
16.         gray ← Round((PC[3*j]+PC[3*j+1]+PC[3*j+2])/3)
17.           jika (gray < Ambang) maka lakukan langkah 18 sampai 20
18.             PH[3*j] ← 0
19.             PH[3*j+1] ← 0
20.             PH[3*j+2] ← 0
21.           jika sebaliknya lakukan langkah 22 sampai 24
22.             PH[3*j] ← 255
23.             PH[3*j+1] ← 255
24.             PH[3*j+2] ← 255
```

3.2.2.4. Flowchart Proses *Thinning Stentiford*

Flowchart ini menunjukkan langkah-langkah dalam proses *thinning* dari sebuah citra. Flowchart digunakan agar mempermudah pembacaan dan penulisan dari algoritma *Stentiford.*

Flowchart proses *Stentiford* ditunjukkan pada gambar sebagai berikut:



Flowchart 4 : Proses *Stentiford*

Pseudocode fungsi *neighbour* dari proses *thinning* adalah sebagai berikut :

```
1.  jika (x >= 0) dan (x < gbr.Width) dan
         (y >= 0) dan (y < gbr.Height) dan
         (k >= 0) dan (k <= 8) maka lakukan langkah 2 sampai37
2.        pada nilai k sama dengan
3.        0 maka lakukan langkah 4 sampai 5
4.          x ← x
5.          y ← y
6.        1 maka lakukan langkah 7 sampai 8
7.          x ← x+1
8.          y ← y
9.       2 maka lakukan langkah 10 sampai 11
10.           x ← x + 1
11.           y ← y - 1
12.        3 maka lakukan langkah 13 sampai 14
13.          x ← x
14.          y ← y-1
15.        4 maka lakukan langkah 16 sampai 17
16.          x ← x-1
17.          y ← y-1
18.        5 maka lakukan langkah 19 sampai 20
19.          x ← x - 1
20.          y ← y
21.        6 maka lakukan langkah 22 sampai 23
22.          x ← x - 1
23.          y ← y + 1
24.        7 maka lakukan langkah 25 sampai 26
25.          x ← x
26.          y ← y + 1
27.        8 maka lakukan langkah 28 sampai 29
28.          x ← x + 1
29.          y ← y + 1
30.      jika (x < 0) maka lakukan langkah 31
31.         x ← 0
32.      jika (x >= gbr.width) maka lakukan langkah 33 sampai 37
33.         x ← gbr.width - 1
34.           jika (y < 0) maka lakukan langkah 35
35.              y ← 0
36.           jika (y >= gbr.Height) maka lakukan langkah 37
37.              y ← gbr.height - 1
38.  result ← Piksels[x,y]
```

Pseudocode fungsi *connectivity* dari proses *thinning* adalah sebagai berikut :

```
1.  jumlah ← 0;
2.  jika (x >= 0) dan (x < gbr.width) dan
         (y >= 0) dan (y < gbr.Height) maka lakukan langkah 3 sampai 6
3.     jika (neighbour(x,y,8) = COLOR_FOREGROUND) dan
            (neighbour(x,y,1) = COLOR_BACKGROUND) maka lakukan langkah 4
4.           count ← 1
5.     jika sebaliknya maka lakukan langkah 6
6.           count ← 0
7.  untuk n ← 1 hingga 7 lakukan langkah 7 sampai 13
8.     jika (neighbour(x,y,n) = COLOR_FOREGROUND) dan
9.          (neighbour(x,y,n+1) = COLOR_BACKGROUND) lakukan langkah 10
10.          inc(count)
11.    jika sebaliknya lakukan langkah 12
12.          count ← count
13.    jumlah ← count
14. result ← jumlah
```

Pseudocode fungsi *endpoint* dari proses *thinning* adalah sebagai berikut :

```
1. res ← false
2.   jika (x >= 0) dan (x < gbr.width) dan
3.        (y >= 0) dan (y < gbr.Height) maka lakukan langkah 3 sampai 12
4.        count ← 0
5.     untuk k ← 1 hingga 8 lakukan langkah 6 sampai 8
6.       jika (gbr.Canvas.Piksels[x,y] = COLOR_FOREGROUND) maka lakukan langkah 7 sampai 8
7.         jika (neighbour(x,y,k) = gbr.Canvas.Piksels[x,y]) maka lakukan langkah 8
8.           count ← count+1
9.     jika count < 2 maka lakukan langkah 10
10.      res ← true
11.    jika sebaliknya maka lakukan langkah 12
12.      res ← false
13.  endpoint ← res
```

Pseudocode proses *thinning* adalah sebagai berikut :

```
1.  remain ← true;
2.    untuk i ← 1 hingga gbr.width-1 lakukan langkah 3 sampai 4
3.      untuk j ← 1 hingga gbr.height-1 lakukan langkah 4
4.        marked[i,j] ← 'uneraseable'

5.  selama (remain) lakukan langkah 6 sampai 36
6.    remain ← false

      {--Template 1--}
7.    untuk y ← 1 hingga Bitmap.Height-1 lakukan langkah 8 sampai 12
8.      untuk x ← 1 hingga Bitmap.Width-1 lakukan langkah 9 sampai 12
9.        jika (piksels[x,y] = COLOR_FOREGROUND) maka lakukan langkah 10 sampai 12
10.         jika (neighbour(x,y,3) = COLOR_BACKGROUND) dan
                 (neighbour(x,y,7) = COLOR_FOREGROUND) maka lakukan langkah 11 sampai 12
11.                jika (endpoint(x,y) = false) dan (connectivity(x,y) < 2) dan
                        (marked[x,y-1] = 'uneraseable') dan
                        (marked[x,y+1] = 'uneraseable') maka lakukan langkah 12
12.                   marked[x,y] ← 'eraseable'

      {--Template 2--}
13.   untuk x ← 1 hingga Bitmap.width-1 lakukan langkah 14 sampai 18
14.     untuk y ← Bitmap.height-1 turun hingga 1 lakukan langkah 15 sampai 18
15.       jika (piksels[x,y] = COLOR_FOREGROUND) maka lakukan langkah 16 sampai 18
16.         jika (neighbour(x,y,5) = COLOR_BACKGROUND) dan
                 (neighbour(x,y,1) = COLOR_FOREGROUND) maka lakukan langkah 17 sampai 18
17.                jika (endpoint(x,y) = false) dan (connectivity(x,y) < 2) dan
                        (marked[x+1,y] = 'uneraseable') dan
                        (marked[x-1,y] = 'uneraseable') maka lakukan langkah 18
18.                   marked[x,y] ← 'eraseable'

      {--Template 3--}
19.   untuk y ← Bitmap.height-1 turun hingga 1 lakukan langkah 20 sampai 24
20.     untuk x ← Bitmap.width-1 turun hingga 1 lakukan langkah 21 sampai 24
21.       jika (piksels[x,y] = COLOR_FOREGROUND) maka lakukan langkah 22 sampai 24
22.         jika (neighbour(x,y,7) = COLOR_BACKGROUND) dan
                 (neighbour(x,y,3) = COLOR_FOREGROUND) maka lakukan langkah 23 sampai 24
23.                jika (endpoint(x,y) = false) dan (connectivity(x,y) < 2) dan
                        (marked[x,y+1] = 'uneraseable') dan
                        (marked[x,y-1] = 'uneraseable') maka lakukan langkah 24
24.                   marked[x,y] ← 'eraseable'
```

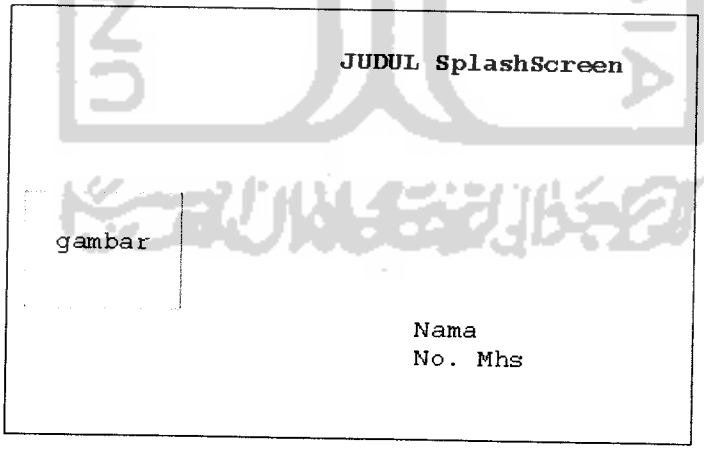
```
        {--Template 4--}
25.     untuk x ← Bitmap.width-1 turun hingga 1 lakukan langkah 26 sampai 30
26.       untuk y ← 1 hingga Bitmap.height-1 lakukan langkah 27 sampai 30
27.         jika (piksels[x,y] = COLOR_FOREGROUND) maka lakukan langkah 28 sampai 30
28.           jika (neighbour(x,y,1) = COLOR_BACKGROUND) dan
                   (neighbour(x,y,5) = COLOR_FOREGROUND) maka lakukan langkah 29 sampai 30
29.             jika (endpoint(x,y) = false) dan (connectivity(x,y) < 2) dan
                     (marked[x-1,y] = 'uneraseable') dan
                     (marked[x+1,y] = 'uneraseable') maka lakukan langkah 30
30.                marked[x,y] := 'eraseable'

        {--Deleting Piksel--}
31.     untuk j ← 1 hingga Bitmap.height - 1 lakukan langkah 32 sampai 36
32.       untuk i ← 1 hingga Bitmap.Width - 1 lakukan langkah 33 sampai 36
33.         jika (marked[i,j] = 'eraseable') maka lakukan langkah 34 sampai 36
34.           piksels[i,j] ← COLOR_BACKGROUND
35.           marked[i,j] ← 'uneraseable'
36.           remain ← true
```

3.2.3. Perancangan Antarmuka

Antarmuka dirancang untuk mendukung proses interaksi pengguna aplikasi dengan aplikasinya. Oleh karena itu antarmuka ini dirancang sedemikian rupa untuk mempermudah pengguna dalam menggunakan aplikasinya. Adapun perancangan antarmuka untuk aplikasi *Stentiford* sebagai berikut :

3.2.3.1. Perancangan Antarmuka *SplasScreen*

SplashScreen adalah screen pembuka dari program utama.

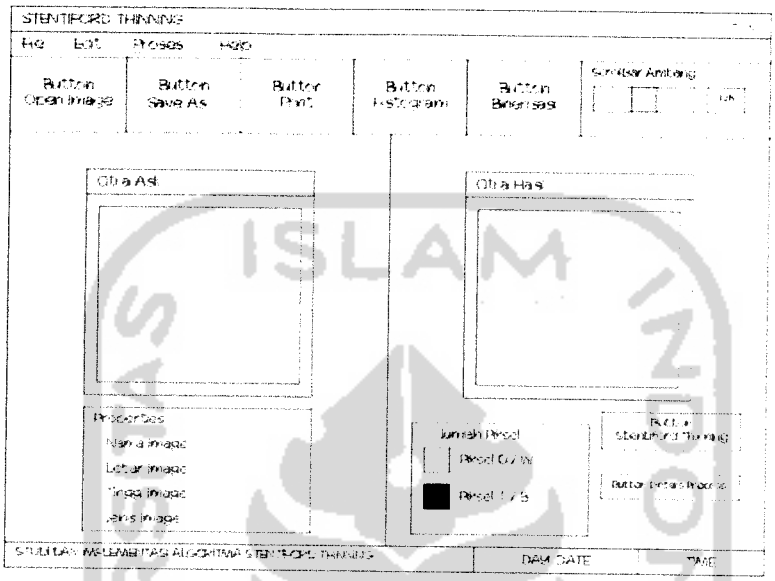


Gambar 12 : Rancangan Antarmuka *SplashScreen*

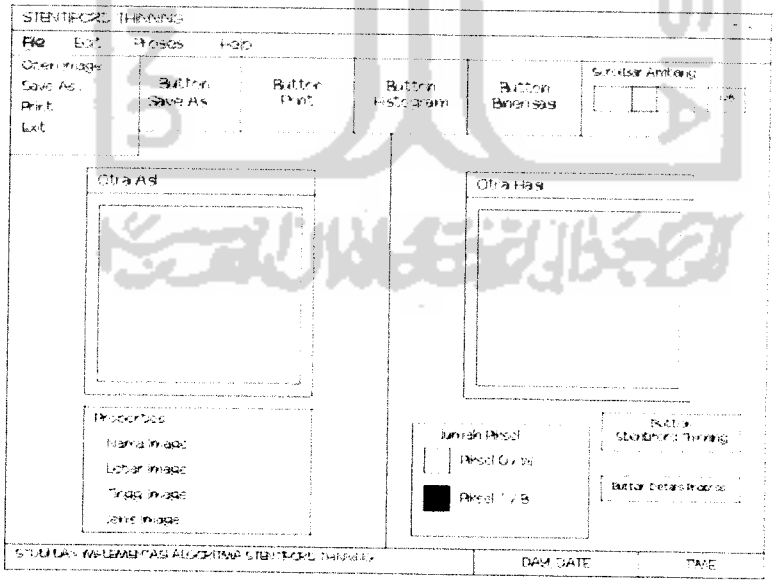
3.2.3.2. Perancangan Antarmuka Menu Utama

Menu utama adalah tampilan awal dari aplikasi yang menyajikan menu-menu utama yang dapat diakses oleh pengguna. Perancangan menu utama

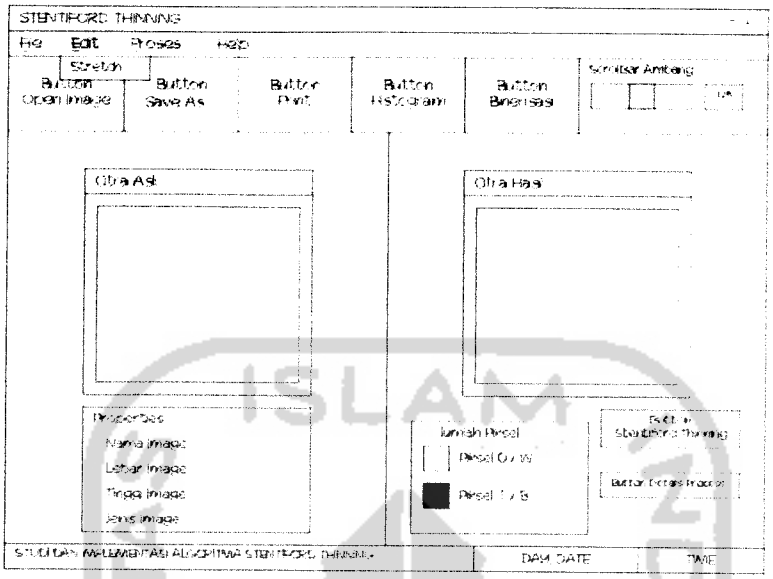
adalah perancangan form yang akan ditampilkan pertama kali dengan menyajikan menu-menu di dalamnya untuk dapat diakses pengguna. Adapun rancangan antarmuka untuk menu utama aplikasi *Stentiford* adalah sebagai berikut :



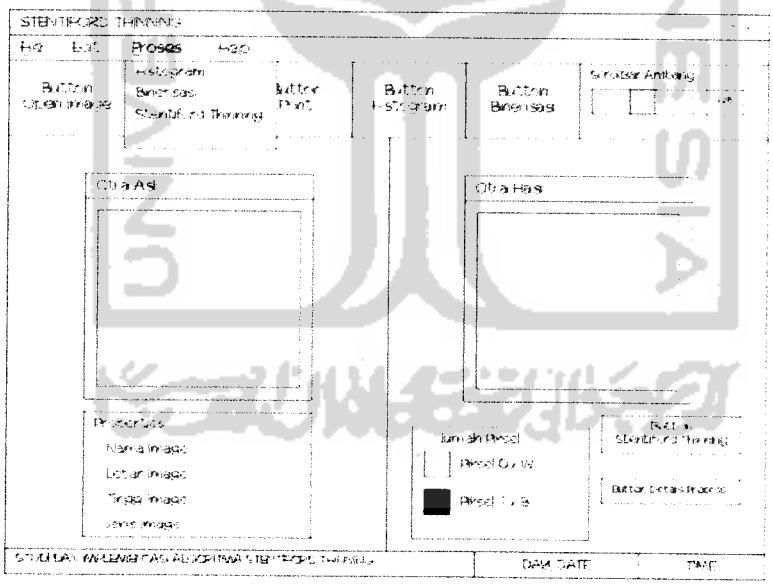
Gambar 13 : Rancangan Antarmuka Menu Utama



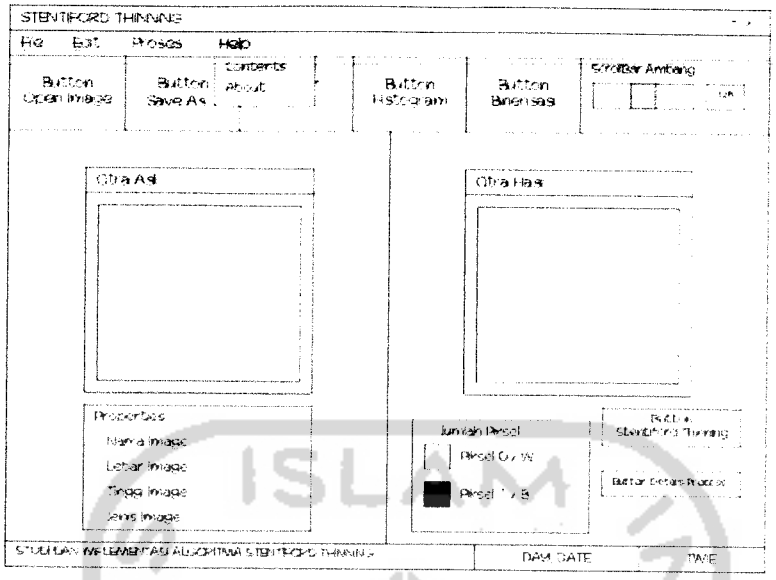
Gambar 14 : Rancangan Antarmuka Menu Utama File



Gambar 15 : Rancangan Antarmuka Menu Utama Edit



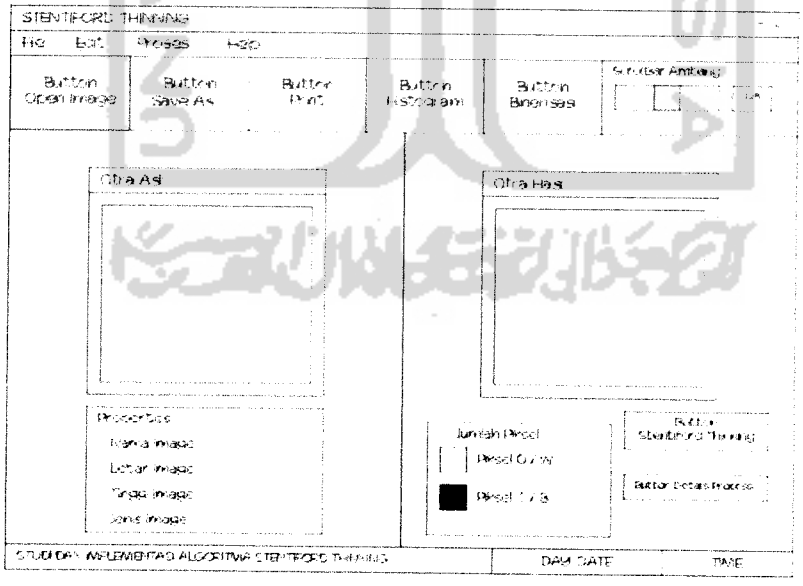
Gambar 16 : Rancangan Antarmuka Menu Utama Proses



Gambar 17 : Rancangan Antarmuka Menu Utama Help

3.2.3.3. Perancangan Antarmuka *Input* Citra

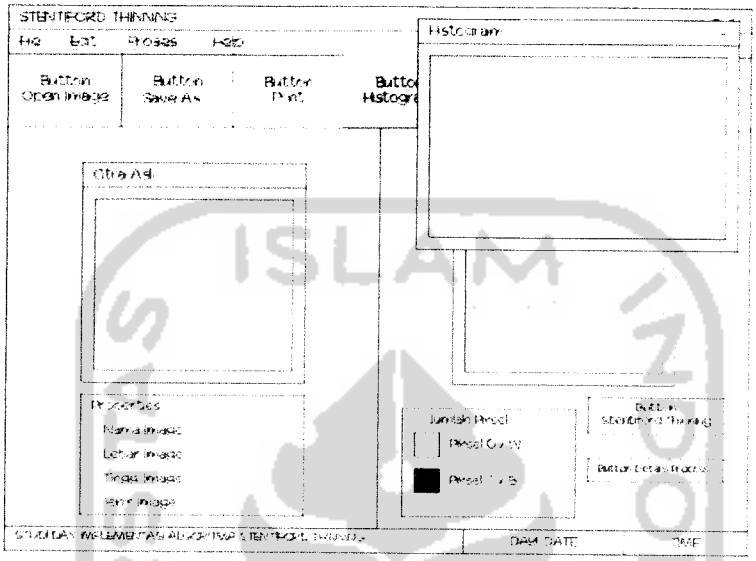
Antarmuka *input* citra ini digunakan untuk memasukkan/*input* sebuah citra yang akan diproses untuk proses selanjutnya. Rancangan antarmuka *input* citra dapat dilihat pada gambar sebagai berikut :



Gambar 18 : Gambar Rancangan Antarmuka *Input* Citra

3.2.3.4. Perancangan Antarmuka Proses Histogram

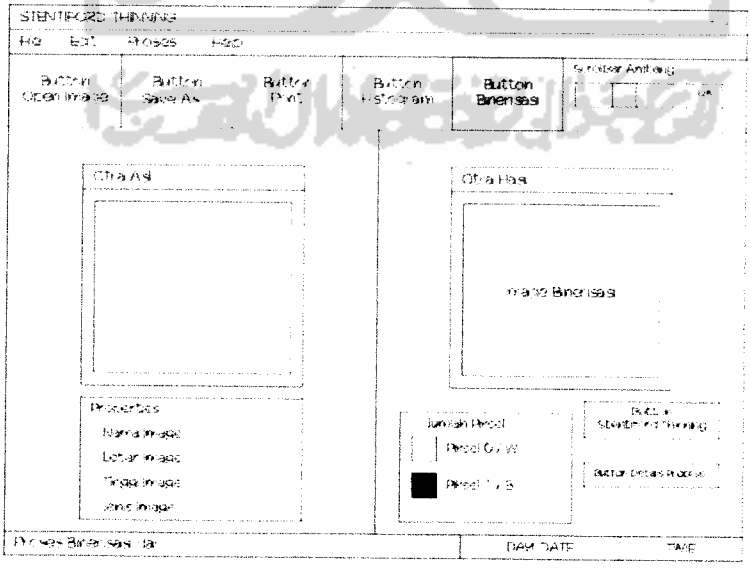
Antarmuka proses histogram ini digunakan untuk menunjukkan histogram dari sebuah citra. Rancangan antarmuka dari proses histogram dapat dilihat pada gambar sebagai berikut :



Gambar 19 : Gambar Rancangan Antarmuka Histogram

3.2.3.5. Perancangan Antarmuka Proses Binerisasi dan Perancangan

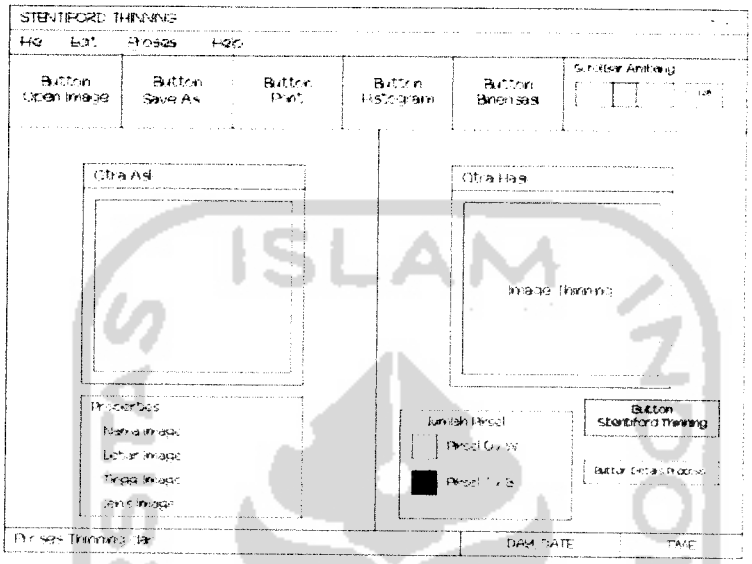
Antarmuka proses binerisasi ini digunakan untuk menunjukkan hasil binerisasi citra. Rancangan antarmuka proses binerisasi citra dapat dilihat pada gambar sebagai berikut :



Gambar 20 : Rancangan Antarmuka Proses Binerisasi

3.2.3.6. Perancangan Antarmuka Proses *Stentiford*

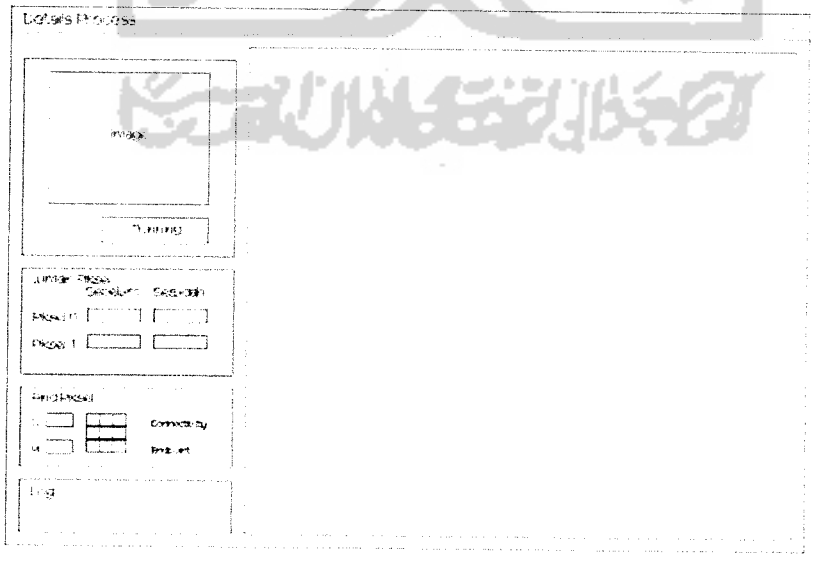
Antarmuka proses *Stentiford* ini digunakan untuk menunjukkan hasil *thinning* dari sebuah citra. Rancangan antarmuka proses *Stentiford* dapat dilihat pada gambar sebagai berikut :



Gambar 21 : Rancangan Antarmuka proses *Stentiford*

3.2.3.7. Perancangan Antarmuka *Details Process*

Antarmuka *Details Process* ini merupakan antarmuka yang menampilkan susunan dari piksel yang bernilai 0 dan 1 dari sebuah citra sehingga menyerupai citra yang sebenarnya.



Gambar 22 : Rancangan Antarmuka *Details Process*

BAB IV
HASIL DAN PEMBAHASAN

4.1. Implementasi Perangkat Lunak

Implementasi merupakan tahap di mana sistem siap dioperasikan pada tahap yang sebenarnya, sehingga akan diketahui apakah sistem yang telah dibuat benar-benar sesuai dengan yang direncanakan. Pada implementasi perangkat lunak ini akan dijelaskan bagaimana sistem ini bekerja, dengan memberikan tampilan form-form yang dibuat.

4.1.1. Batasan Implementasi

Aplikasi Implementasi Algoritma *Stentiford* ini memiliki batasan-batasan sebagai berikut :

1. Aplikasi hanya berjalan pada sistem operasi Windows.
2. Citra digital yang diolah adalah citra biner.
3. Citra digital yang akan diolah bukan merupakan citra abstrak dan rusak.
4. File grafik yang digunakan adalah file grafik berformat bitmap (*.bmp).
5. Ukuran citra yang akan diolah maksimal 290 piksel x 290 piksel.
6. Citra yang sudah diolah disimpan dalam format bitmap (*.bmp) dan dapat didokumentasikan dalam bentuk *printout*.

4.1.2. Implementasi Antarmuka

Implementasi dari aplikasi implementasi Algoritma *Stentiford* ini terdiri dari beberapa form yang memiliki fungsi sendiri-sendiri. Form-form tersebut akan tampil sesuai dengan pilihan proses yang diinginkan dan berurutan sesuai dengan urutan yang telah terprogram.

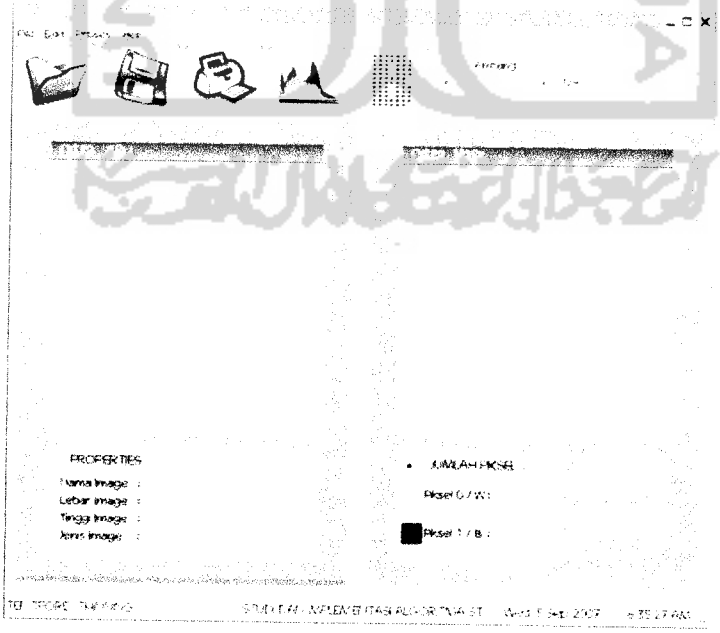
4.1.2.1. Implementasi Antarmuka *SpalshScreen*

Screen pembuka dari program utama dapat dilihat pada gambar berikut:

Gambar 23 : SplashScreen

4.1.2.2. Implementasi Antarmuka Menu Utama

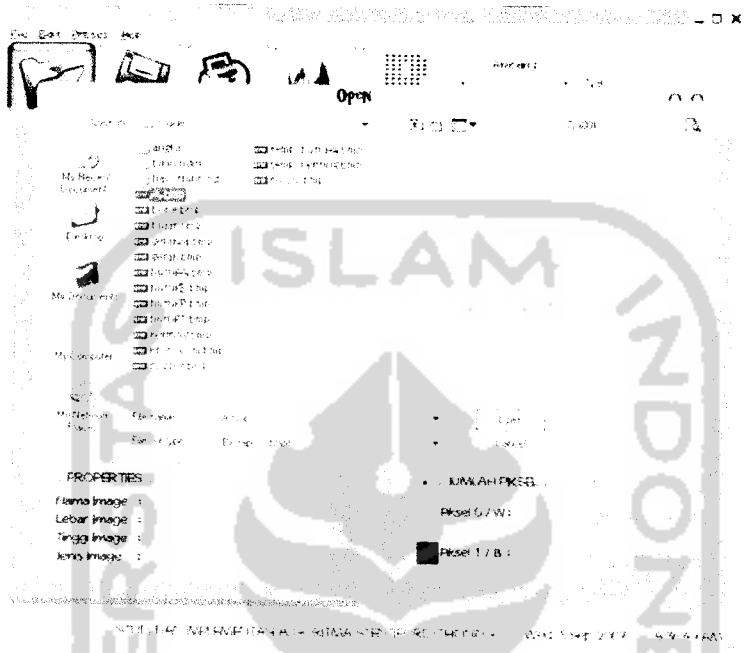
Menu utama ini merupakan menu awal dari aplikasi implementasi Algoritma *Stentiford* yang disajikan dalam form sendiri. Terdapat beberapa menu utama antara lain *File, Edit*, Proses, *Button Open, Button Save, Button Print*, *Button* Histogram, *Button* Binerisasi, dan *Button Stentiford.* Tampilan dari menu utama dapat dilihat pada gambar berikut :



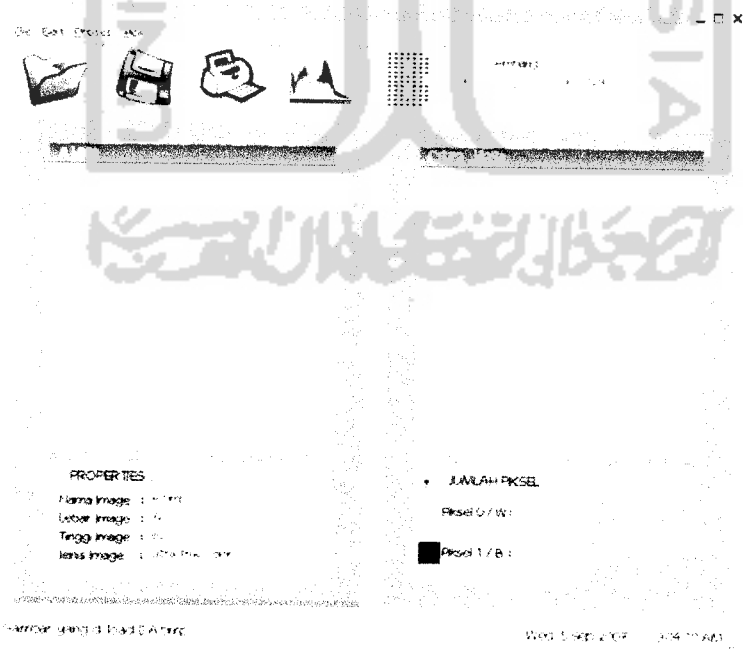
Gambar 24 : Menu Utama

4.1.2.3. Implementasi Antarmuka *Input* Citra

Antarmuka *input* citra ini digunakan untuk memasukkan citra yang akan diolah pada proses selanjutnya. Tampilan *input* citra dapat dilihat pada gambar:



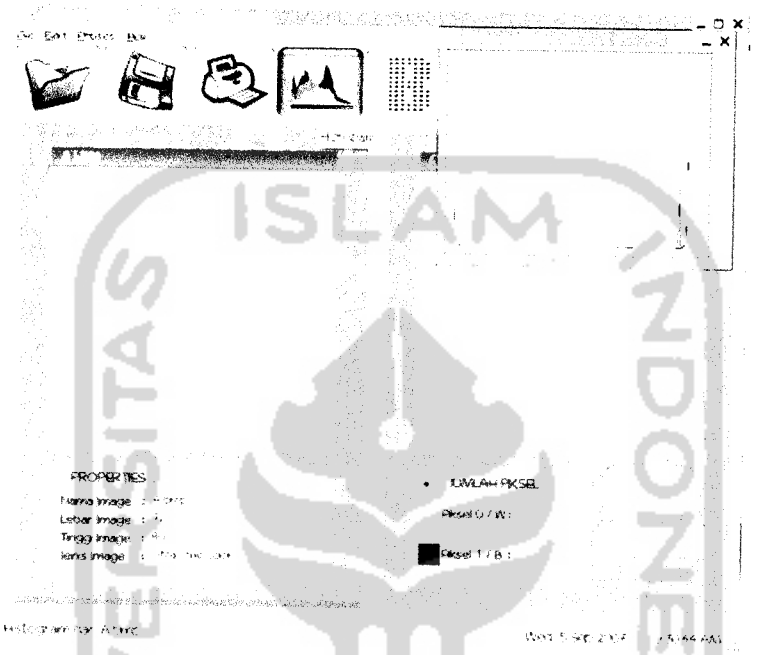
Gambar 25 : Proses *Input* Citra



Gambar 26 : Citra yang telah di*input*

4.1.2.4. Implementasi Antarmuka Proses Histogram

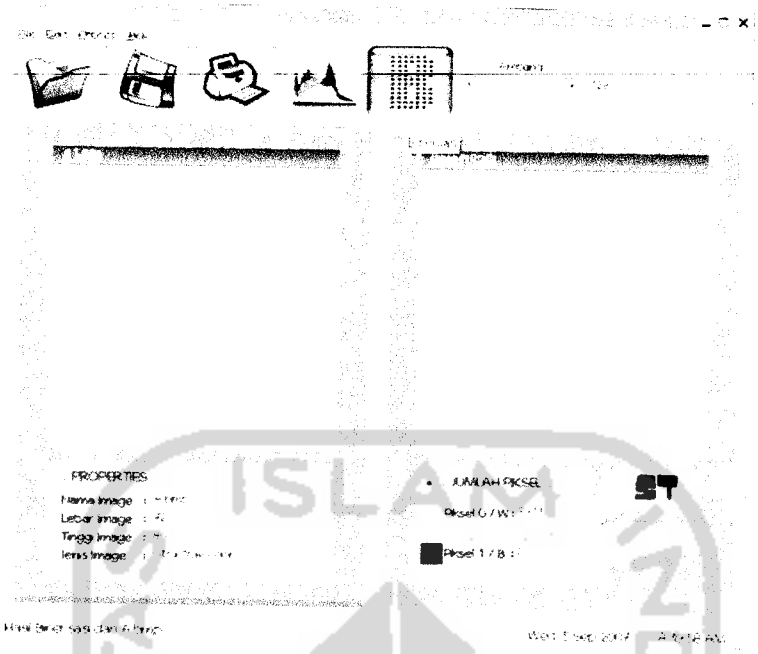
Antarmuka proses histogram ini digunakan untuk menunjukkan histogram dari sebuah citra. Implementasi antarmuka proses histogram dapat dilihat pada gambar :



Gambar 27 : Proses Histogram

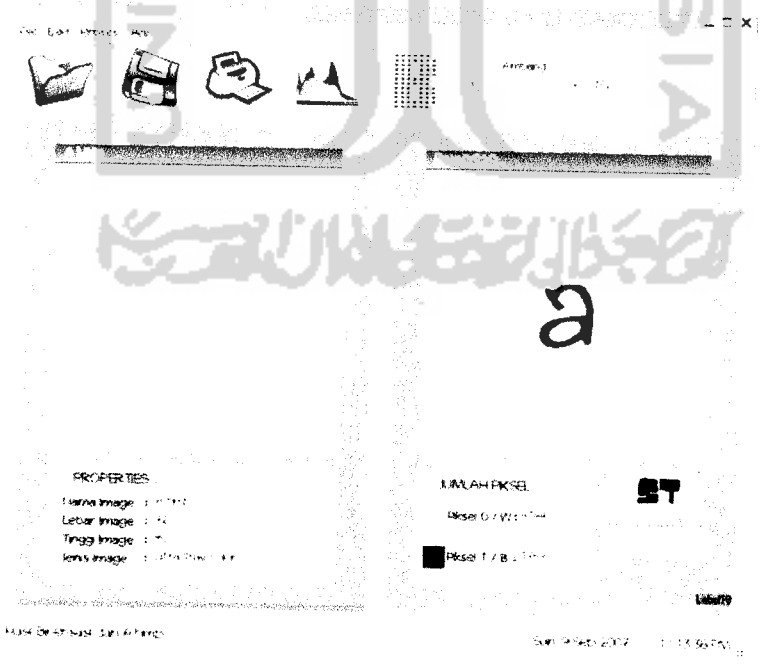
4.1.2.5. Implementasi Proses Binerisasi dan Pengambangan

Antarmuka proses binerisasi dan pengambangan ini digunakan untuk menunjukkan hasil dari binerisasi sebuah citra. Implementasi antarmuka proses binerisasi citra dapat dilihat pada gambar :



Gambar 28 : Proses Binerisasi

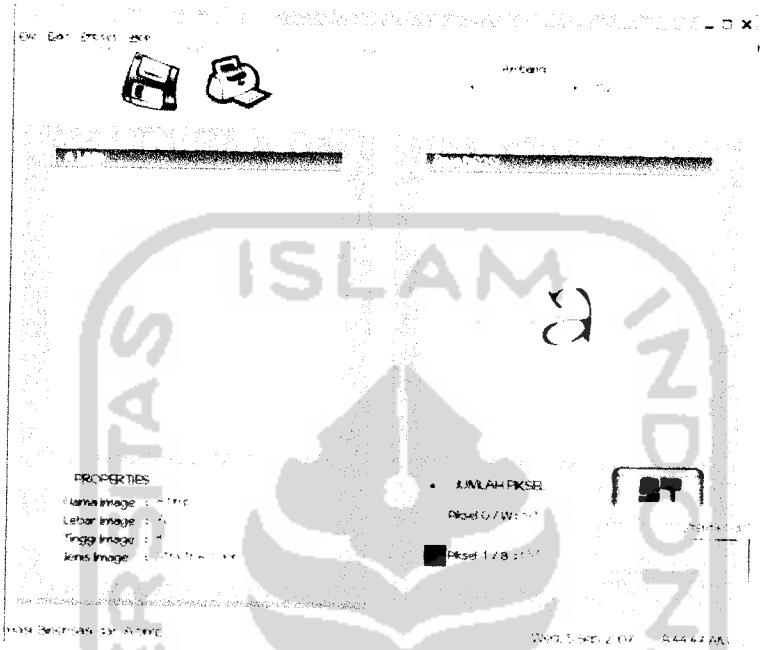
Setelah citra di*input*kan ke dalam form aplikasi, kemudian dilakukan proses binerisasi, jika citra hasil binerisasi belum baik, maka dilakukan proses pengambangan.



Gambar 29 : Proses Pengambangan

4.1.2.6. Implementasi Proses *Stentiford*

Antarmuka proses *Stentiford* ini digunakan untuk menunjukkan hasil dari proses *Stentiford* sebuah citra. Implementasi antarmuka proses *Stentiford* dapat dilihat pada gambar :

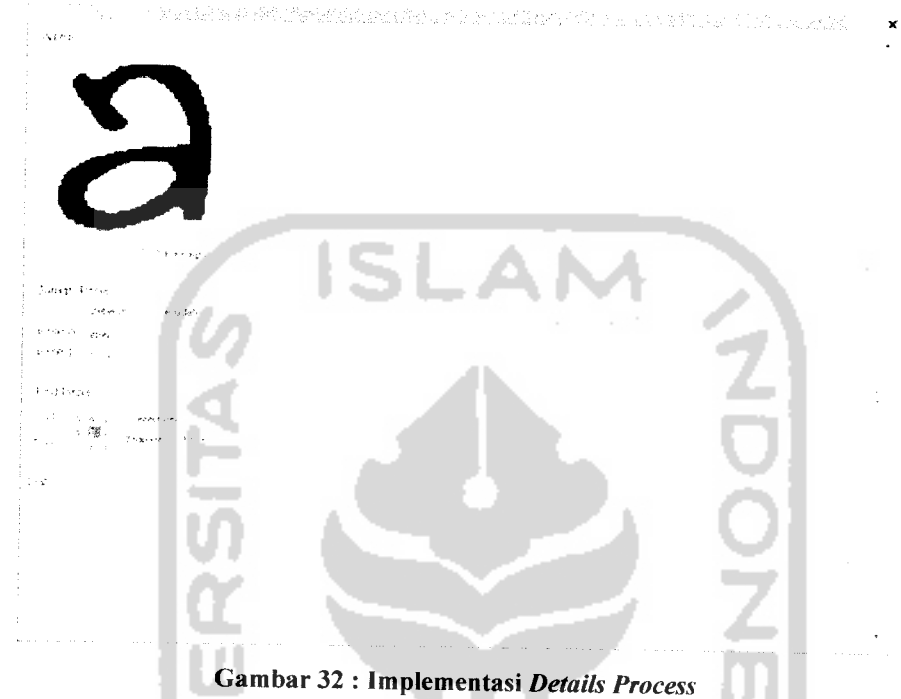


Gambar 30 : Proses *Stentiford*

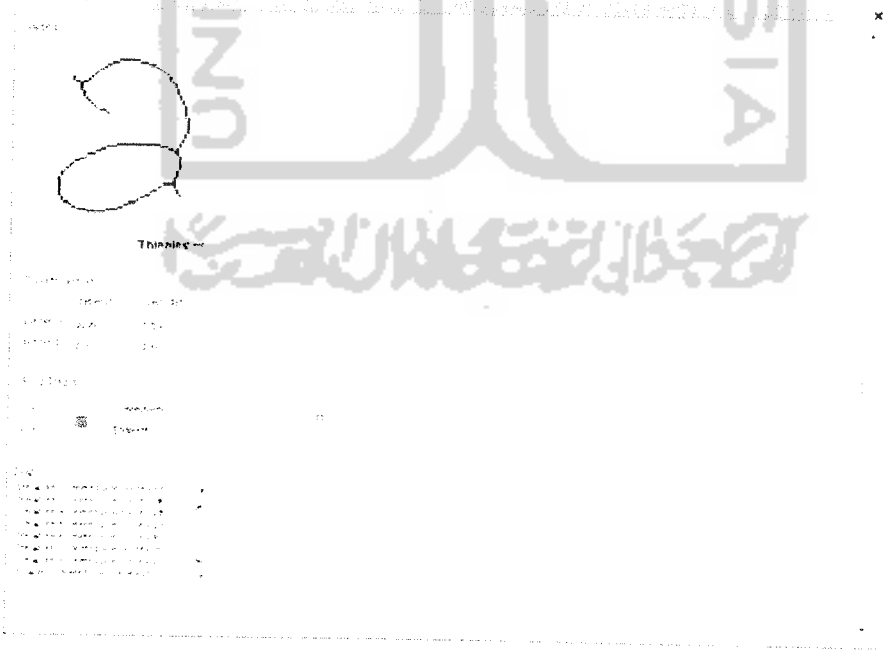
Gambar 31 : Waktu proses *thinning*

4.1.2.7. Implementasi Details Process

Antarmuka *Details process* ini merupakan antarmuka yang menampilkan susunan dari piksel yang bernilai 0 dan 1 dari sebuah citra sehingga membentuk citra.



Gambar 32 : Implementasi *Details Process*



Gambar 33 : Implementasi *Details Process Thinning*

4.2. Pengujian Program

Pada tahap analisis kinerja perangkat lunak dijelaskan tentang pengujian aplikasi dari implementasi algoritma *Stentiford*. Pengujian dilakukan dengan kompleks dan diharapkan dapat diketahui kekurangan-kekurangan dari sistem untuk kemudian diperbaiki sehingga kesalahan dari sistem dapat diminimalisasi. Pengujian sistem ini dilakukan untuk mendapatkan hasil yang akurat.

Pengujian sistem ini dapat dilakukan dengan meng*input*kan sebuah citra ke dalam form utama untuk kemudian dilakukan proses histogram, binerisasi, *Stentiford* dan details process.

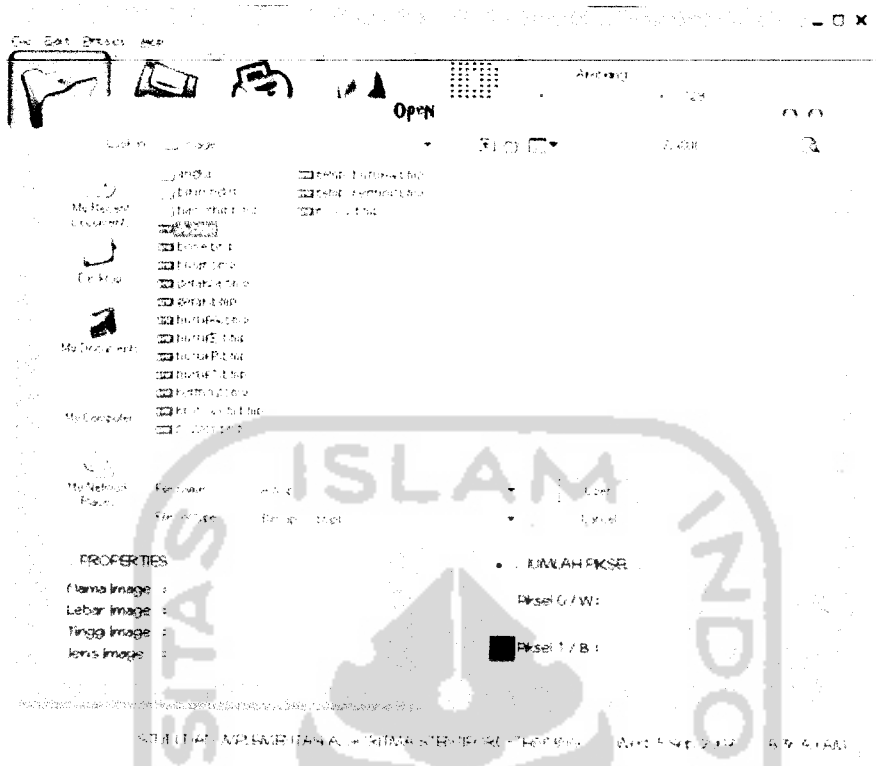
4.3. Analisis Kinerja Sistem

4.3.1. Pengujian dan Analisis

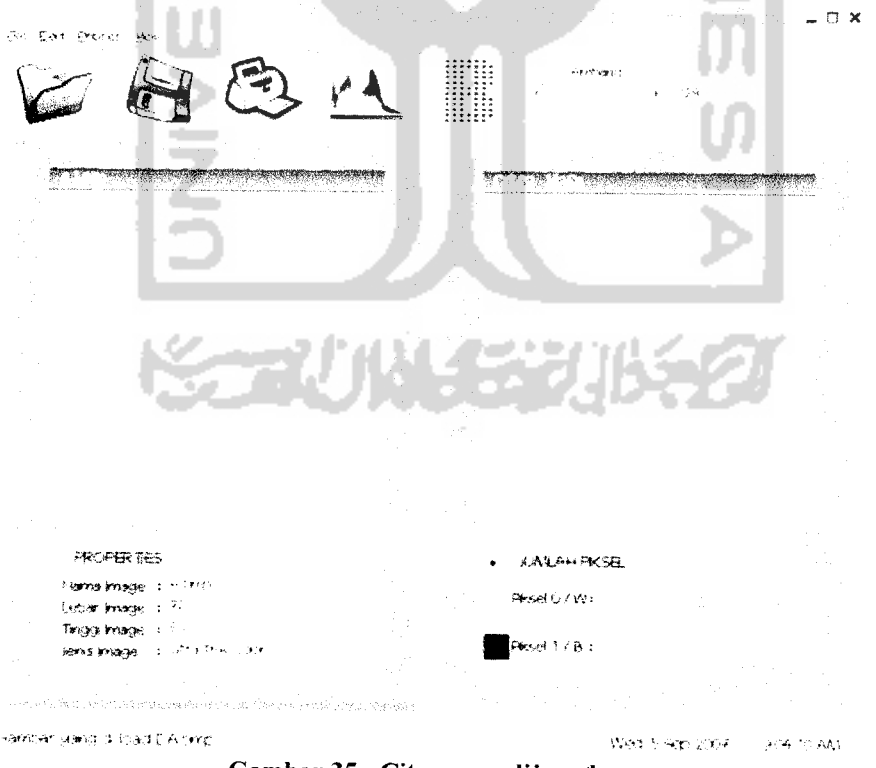
Pada tahap pengujian dan analisis program ini, dilakukan uji coba implementasi program dengan memberikan *input*an dan menganalisis proses yang terjadi untuk menghasilkan *output* yang sesuai, sebagaiamana alur perancangan yang sudah dibuat oleh penyusun sebelumnya.

4.3.1.1. Pengujian *Input* Citra

Tekan tombol *input* citra atau pilih menu *input* citra.



Gambar 34 : Peng*input*an Citra



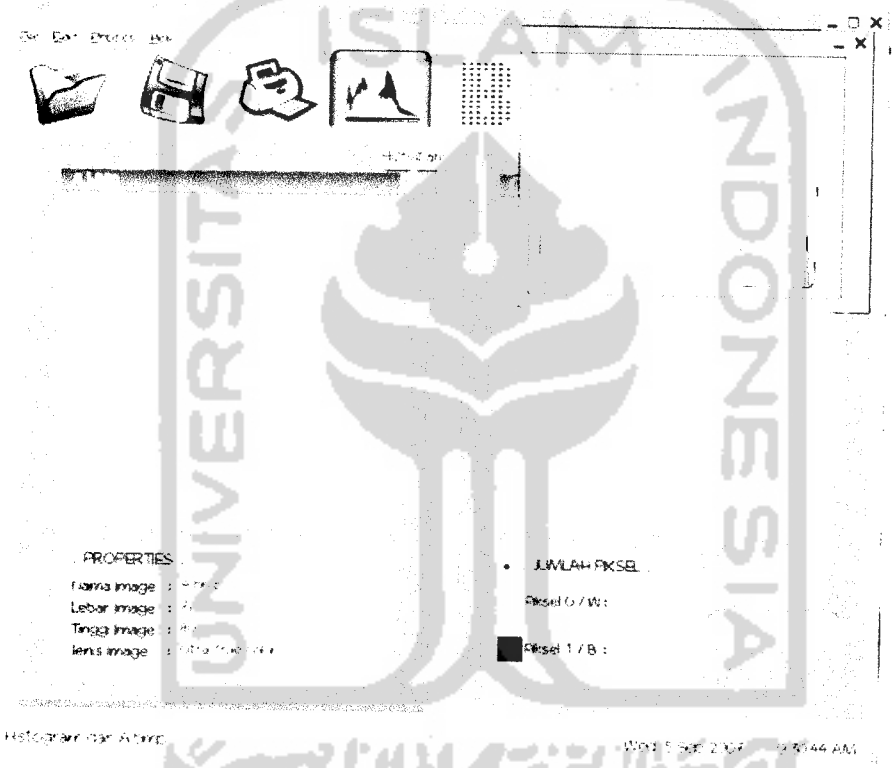
Gambar 35 : Citra yang di*input*kan

Keterangan :

- Nama *Image* : A.bmp
- Lebar *Image* : 72 piksel
- Tinggi *Image* : 80 piksel
- Jenis *Image* : Citra *True Color*

4.3.1.2. Pengujian Proses Histogram

Tekan tombol histogram atau pilih menu histogram.



Gambar 36 : Histogram *Image* A.bmp

4.3.1.3. Pengujian Proses Binerisasi

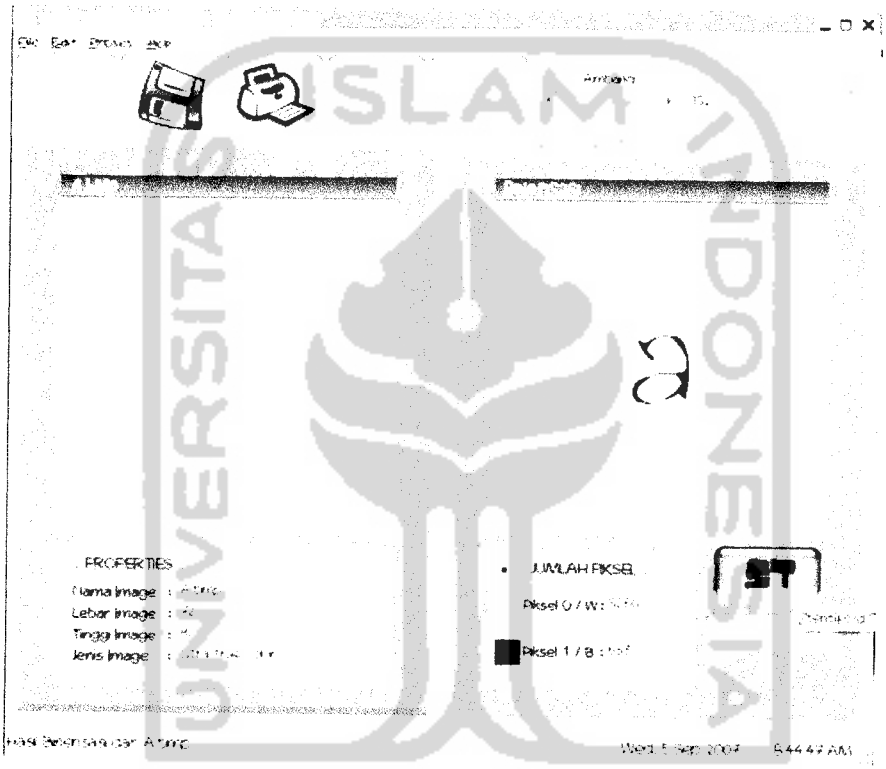
Tekan tombol binerisasi atau pilih menu binerisasi. Jika pada kolom citra hasil masih belum terlihat hasil *image* binerisasi, maka lakukan proses pengambangan dengan cara mengubah nilai ambang (menggeser *scrollbar* ambang) sampai didapatkan hasil *image* binerisasi yang baik.

Dari proses-proses di atas, kita mendapatkan informasi bahwa :

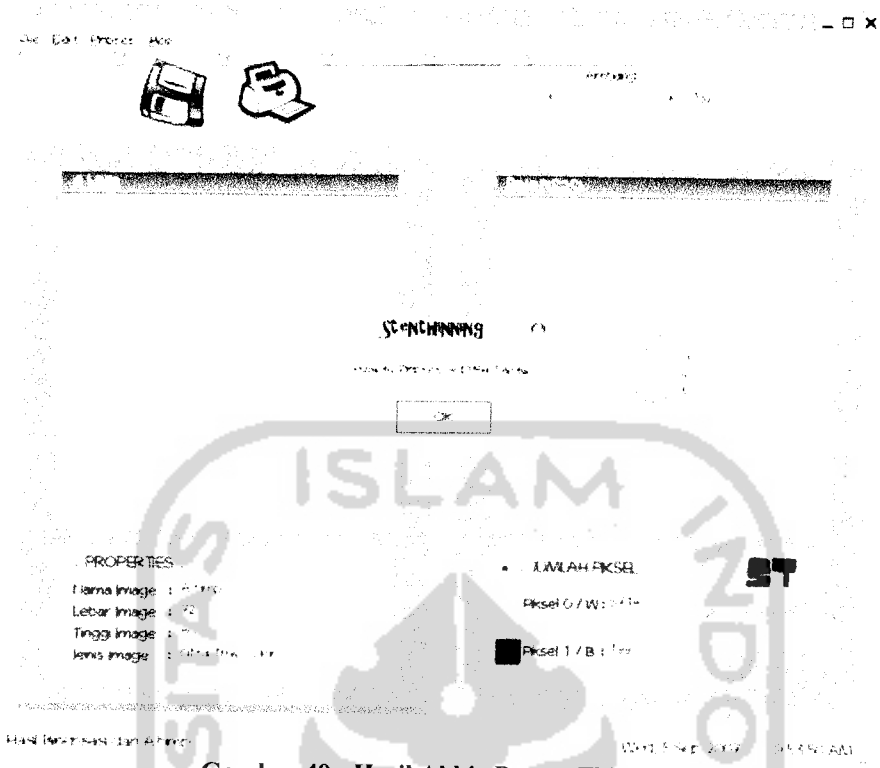
- Nilai ambang = 152
- Jumlah piksel 0 (berwarna putih) = 4144 piksel
- Jumlah piksel 1 (berwarna hitam) = 1769 piksel

4.3.1.4. Pengujian Proses *Stentiford*

Tekan tombol *Stentiford* atau pilih menu *Stentiford.*



Gambar 39 : Proses *Thinning* sedang berlangsung



Gambar 40 : Hasil Akhir Proses *Thinning*

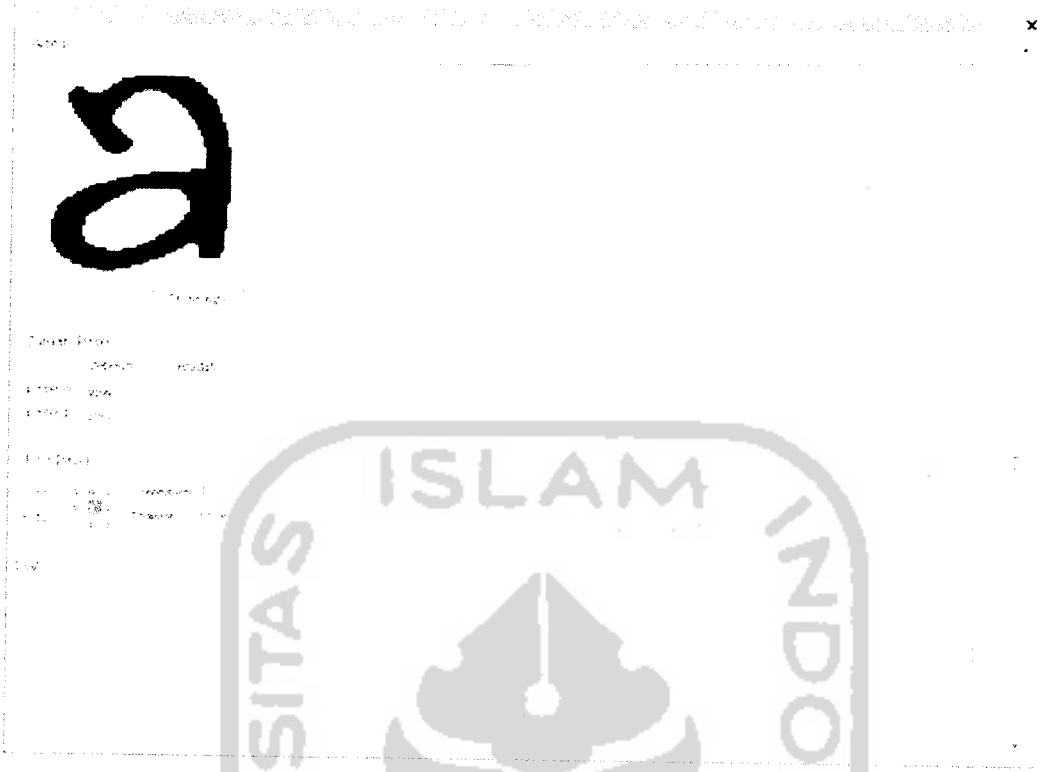
Informasi yang diperoleh dari hasil akhir proses *thinning* adalah :

- Jumlah piksel 0 (putih) = 5714 piksel
- Jumlah piksel 1 (hitam) = 199 piksel
- Waktu proses = 0,547 detik

4.3.1.5. Pengujian Details Process

Jika proses *thinning* telah selesai, kemudian pengguna bisa menekan tombol details process untuk melihat proses *thinning* secara detail.

Pada pengujian details process ini kita akan mendapatkan informasi mengenai berapa banyak iterasi yang terjadi dalam penghapusan piksel, berapa jumlah piksel yang dihapus pada tiap iterasinya, dan dapat melihat proses penghapusan piksel.



Gambar 41 : *Details Process Thinning*

Dari form details process ini, kita mendapatkan informasi :

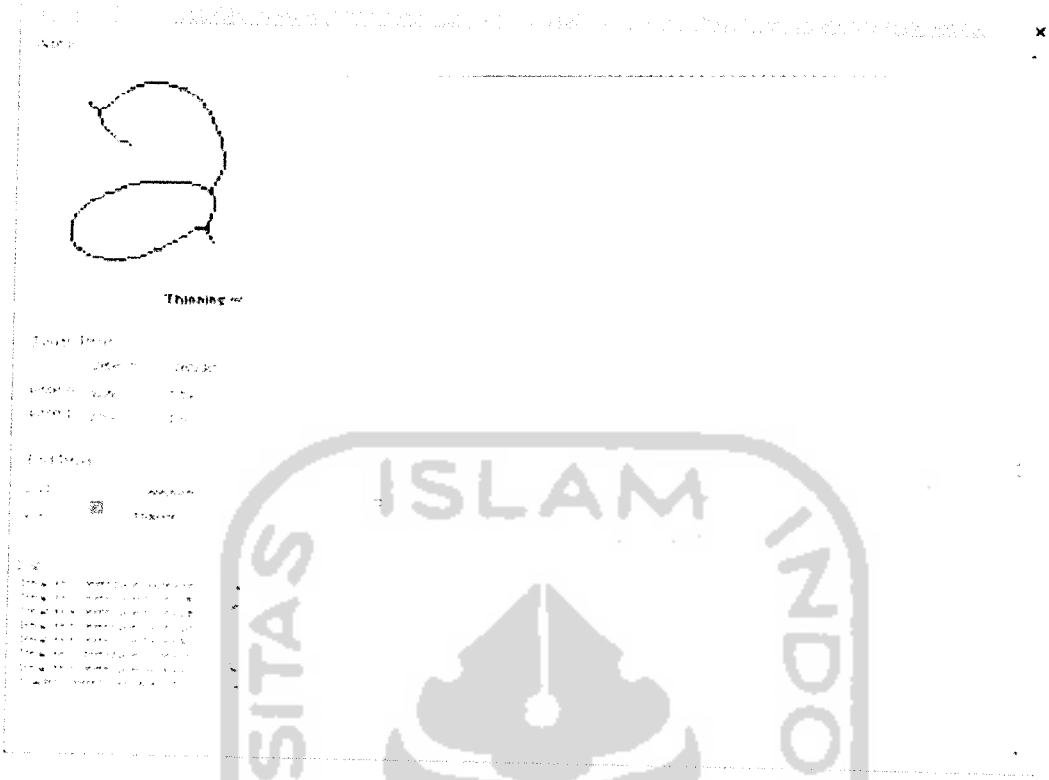
- Nama *Image* = A.bmp
- Jumlah piksel 0 (sebelum) = 4144 piksel
- Jumlah piksel 1 (sebelum) = 1769 piksel
- Posisi piksel yang dipilih

  X = 11

  Y = 52
- Tabel yang menunjukkan nilai piksel tetangga piksel [11,52]

0	0	1
0	1	1
1	1	1

- Nilai *Connectivity* = 1
- *Endpoint* = false



Gambar 42 : Hasil Details Process *Thinning*

Dari form details process ini, kita mendapatkan informasi :

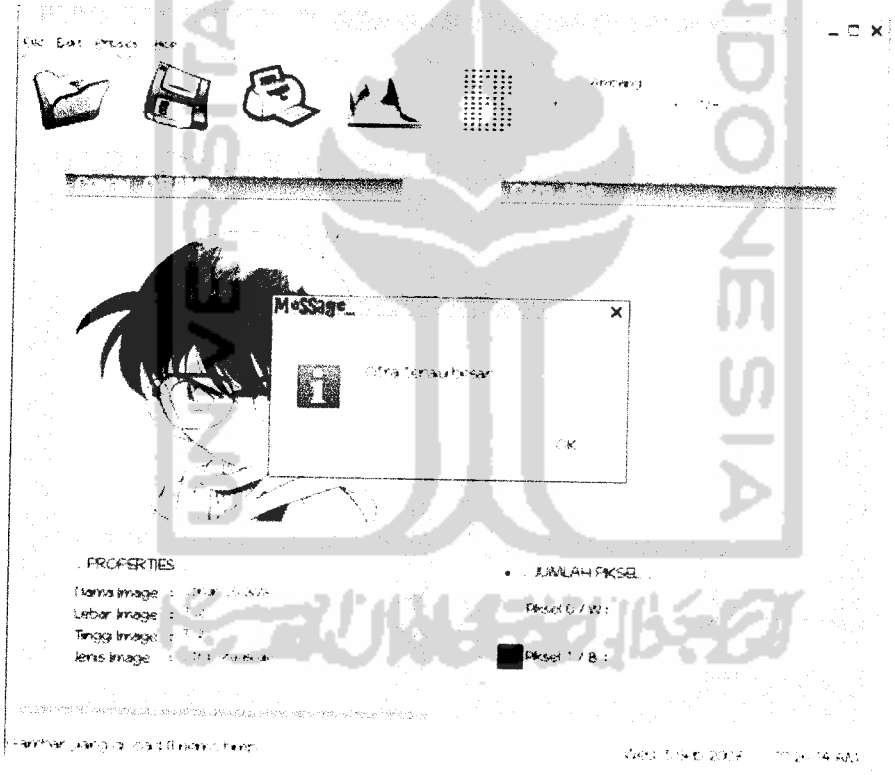
- Nama *Image* = A.bmp
- Jumlah piksel 0 (sebelum) = 3982
- Jumlah piksel 1 (sebelum) = 1931
- Jumlah piksel 0 (sesudah) = 5714
- Jumlah piksel 1 (sesudah) = 199
- Log = terjadi 8 iterasi
  - Iterasi ke-1, *deleted pixel count* = 391
  - Iterasi ke-2, *deleted pixel count* = 389
  - Iterasi ke-3, *deleted pixel count* = 336
  - Iterasi ke-4, *deleted pixel count* = 235
  - Iterasi ke-5, *deleted pixel count* = 133
  - Iterasi ke-6, *deleted pixel count* = 50
  - Iterasi ke-7, *deleted pixel count* = 17
  - Iterasi ke-8, *deleted pixel count* = 1
  - Final step, *deleted pixel count* = 18

4.3.2. Penanganan Kesalahan

Perangkat lunak ini dibuat dengan memperhatikan aspek interaksi manusia dan komputer , artinya aplikasi ini dibuat sedemikian rupa agar mudah dimengerti dan diterapkan oleh pengguna. Jika terdapat kesalahan-kesalahan pemasukan data, maka sistem akan memberikan tanggapan (*feedback*) kepada pengguna berupa informasi kesalahan yang terjadi. Ada beberapa tipe dari penanganan kesalahan antara lain :

4.3.2.1. Penanganan Kesalahan *Input* Ukuran Citra

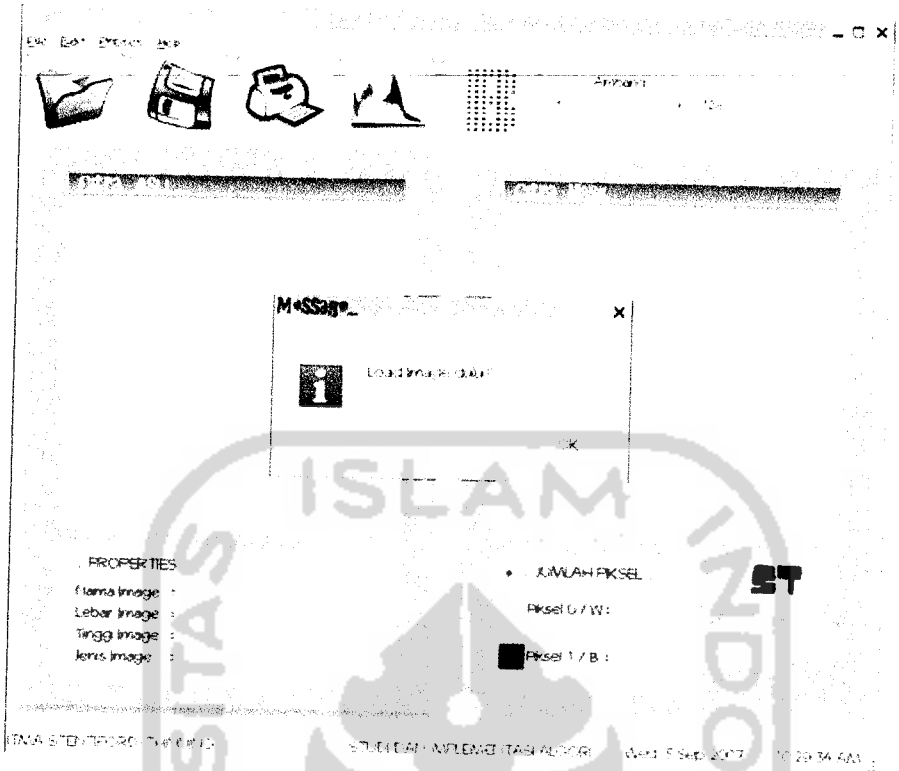
Penanganan kesalahan ini terjadi ketika pengguna memasukkan citra yang ukurannya lebih dari 290 piksel x 290 piksel.



Gambar 43 : Kesalahan *Input* Ukuran Citra

4.3.2.2. Penanganan Kesalahan Kolom Citra Masih Kosong

Penanganan kesalahan ini dilakukan jika kolom citra masih kosong dan pengguna menekan tombol *save*, tombol *print*, tombol histogram, tombol binerisasi, dan tombol *Stentiford*.



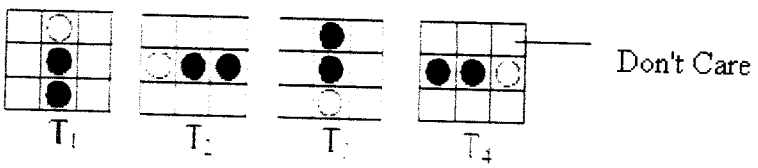
Gambar 44 : Kesalahan Citra masih kosong

4.4. Komparasi Algoritma *Stentiford Thinning* dengan Algoritma *Thinning* Lainnya

4.4.1. Algoritma *Stentiford*

Algoritma ini merupakan algoritma iteratif yang menggunakan *Template based mark-and-delete* untuk menandai piksel yang akan dihapus. Metode *thinning* jenis ini menggunakan template untuk dicocokkan dengan citra yang akan di-*thinning*, di mana *image* yang cocok dengan template, maka piksel tengahnya akan di-*delete*.

Empat buah template yang digunakan pada algoritma ini adalah :



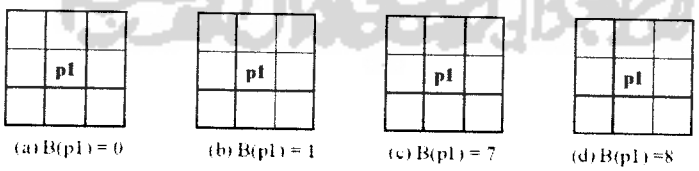
Selain template di atas, dalam pemrosesan *thinning* menggunakan algoritma *Stentiford* juga memperhitungkan nilai *connectivity* (merupakan suatu ukuran berapa banyak objek yang terhubung dengan piksel tertentu) dan *endpoint* (piksel yang merupakan batas akhir dan hanya terhubung dengan 1 piksel saja).

4.4.2. Algoritma Thinning-Hilditch

Algoritma ini merupakan algoritma iteratif yang menggunakan tetangga *Moore* untuk menandai piksel yang akan dihapus dari titik yang kita tinjau. Pada algoritma yang merupakan operasi terhubung-8 ini dilakukan beberapa kali iterasi pengikisan pada suatu obyek, di mana pada setiap pengikisan dilakukan pemeriksaan pada semua titik dalam citra dan melakukan pengubahan sebuah titik obyek menjadi titik latar apabila memenuhi keempat kriteria berikut ini:

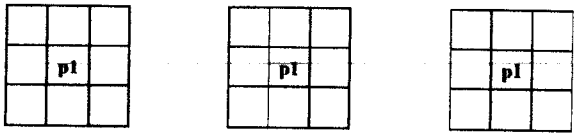
1) $2 \leq B(p1) \leq 6$
2) $A(p1) = 1$
3) p2, p4, atau p8 ada yang merupakan titik latar, atau $A(p2) \neq 1$
4) p2, p4, atau p6 ada yang merupakan titik latar, atau $A(p4) \neq 1$

Algoritma dihentikan apabila pada suatu iterasi tidak ada lagi titik yang diubah. Kriteria 1 menunjukkan bahwa tidak ada titik terisolasi, B(p1) = 0, ataupun titik ujung, B(p1) = 1, yang terkikis. Demikian juga titik yang ada di dalam obyek seperti pada (d), sedangkan titik pada (c) juga tidak boleh dihilangkan untuk mencegah pengecilan kerangka.



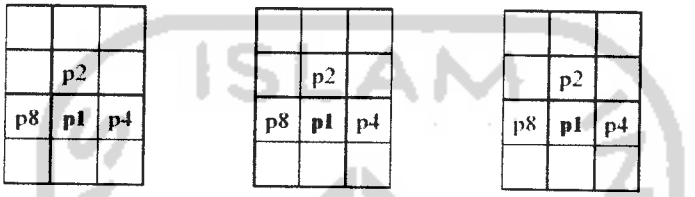
(a) B(p1) = 0 (b) B(p1) = 1 (c) B(p1) = 7 (d) B(p1) =8

Kriteria 2 menunjukkan sifat konektivitas, dimana jika kita menghilangkan suatu titik mempunyai nilai A lebih dari 1, maka pola atau kerangka akan menjadi terputus. Dengan demikian titik p1 pada contoh-contoh tersebut tidak boleh dihapus.



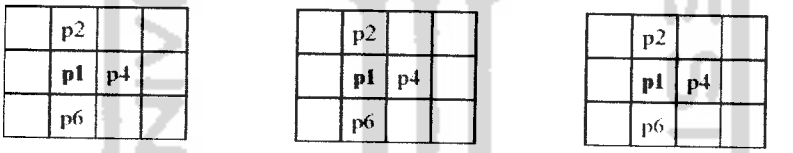
(a) $A(p1) = 2$ (b) $A(p1) = 2$ (c) $A(p1) = 3$

Kriteria 3 digunakan untuk menghindarkan terhapusnya garis horisontal dengan lebar 2 titik. Dalam gambar tersebut, titik obyek pada (a) tidak memenuhi kriteria 3 sehingga tidak boleh dihapus, sedangkan titik p1 pada (b) dan (c) memenuhi kriteria, sehingga mungkin akan dihapus.



(a) garis vertikal berlebar 2 titik (b) p2, p4, p8 ≠ latar $A(p2) \neq 1$ (c) p2, p4, atau p8 = latar $A(p2) = 1$

Kriteria 4 mirip dengan kriteria 3, namun digunakan untuk menghindarkan terhapusnya garis vertikal berlebar 2 titik. Titik p1 pada (a) tidak memenuhi kriteria tersebut sehingga tidak boleh dihapus, sedangkan pada (b) dan (d) memenuhi syarat, sehingga mungkin dapat dihapus.



(a) garis vertikal berlebar 2 titik (b) p2, p4, p6 ≠ latar $A(p4) \neq 1$ (c) p2, p4, atau p6 = latar $A(p4) = 1$

Algoritma dihentikan apabila pada suatu iterasi tidak ada lagi titik yang diubah.

4.4.3. Perbandingan Hasil *Thinning*

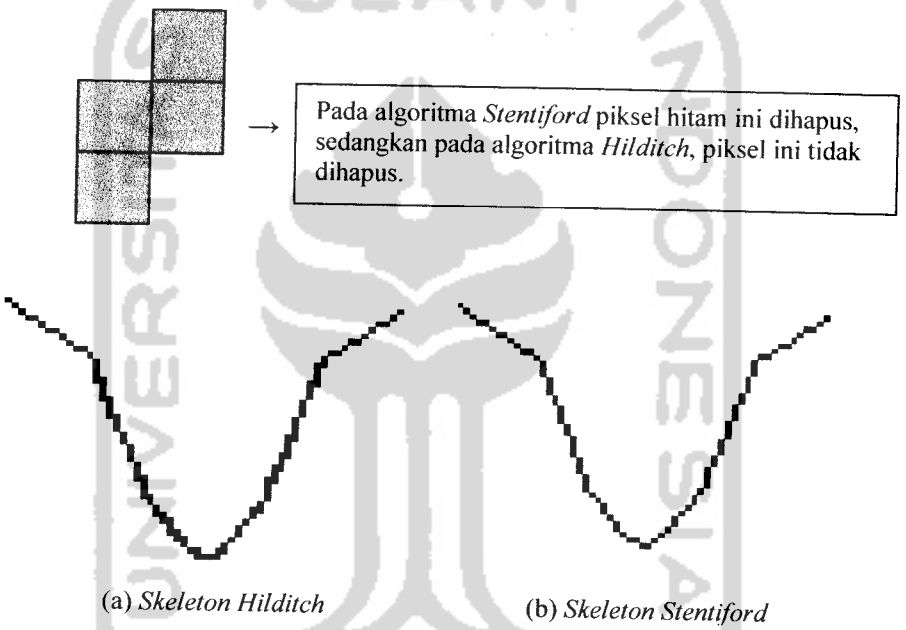
Tabel 1 : Tabel Komparasi Hasil *Thinning*

Citra			Algoritma *Stentiford*					Algoritma *Hilditch*				
Gambar	Jumlah Piksel		Gambar	Jumlah iterasi	Waktu proses	Jumlah Piksel		Gambar	Jumlah iterasi	Waktu proses	Jumlah Piksel	
	Hitam (1)	Putih (0)				Hitam (1)	Putih (0)				Hitam (1)	Putih (0)
A	1998	4538		10	0.734 s	152	6384		10	11 s	195	6341
E	2241	3865		10	0.578 s	158	5948		10	11 s	170	5936
P	1968	4138		10	0.578 s	141	5965		10	11 s	154	5952
T	1377	4729		10	0.5 s	94	6012		9	10 s	94	6012
	1945	14615		6	0.36 s	256	15452		6	7 s	318	15390

	1561	5000		4	0.344 s	249	6312		4	5 s	263	6298
	1242	5319		8	0.516 s	220	6341		7	8 s	265	6296
8	1825	4736		9	0.547 s	173	6388		8	9 s	206	6355
3	1832	4729		9	0.578 s	131	6430		9	10 s	160	6401
M	1870	4691		9	0.719 s	161	6400		8	9 s	191	6370
O	2214	4347		8	0.578 s	162	6399		7	8 s	205	6356

Berdasarkan tabel di atas, kita dapat melihat bahwa dibandingkan dengan algoritma *Hilditch* yang digunakan pada penelitian sebelumnya, algoritma *Stentiford* menunjukkan hasil (*skeleton*) yang lebih tipis dan waktu eksekusi yang lebih cepat. Kedua hal ini disebabkan karena :

1. Hasil akhir *image thinning* algoritma *Stentiford* menghasilkan jumlah piksel 1 (hitam) yang lebih sedikit dibandingkan dengan hasil yang diproses dengan algoritma *Hilditch*, karena algoritma *Hilditch* tidak dapat menyentuh sudut-sudut tertentu dari sebuah citra. Misal :



Gambar 45 : Perbandingan *Skeleton* Huruf V pada algoritma *Hilditch* dan *Stentiford*

2. Hasil dari proses *thinning* dengan Algoritma *thinning Stentiford* lebih sempurna (tipis) karena algoritma ini melakukan penghapusan secara tidak langsung (*mark-and-delete*), serta melakukan penghapusan secara memutar, penghapusan objek terluar dilakukan terlebih dahulu dengan menggunakan template 3x3 dan syarat-syarat lainnya.
3. Hasil yang lebih sempurna ini juga disebabkan karena algoritma *Stentiford* memberikan syarat tambahan untuk memperhalus citra yang dihasilkannya.

4. Proses penghapusan pada algoritma *Stentiford* dalam penelitian ini dilakukan dengan lebih cepat daripada penelitian sebelumnya yang menggunakan algoritma *Hilditch* dikarenakan perbedaan cara implementasi algoritma antara kedua penelitian. Pada penelitian sebelumnya, algoritma *Hilditch* diimplementasikan dengan proses penggambaran ulang citra secara keseluruhan, jadi, proses penghapusan dilakukan dalam *temporary array* dan penulis selalu menggambar ulang (*redraw*) citra pada tiap iterasi. Sedangkan pada implementasi algoritma *Stentiford* dalam penelitian ini, penulis tidak melakukan penggambaran ulang citra secara keseluruhan, hanya melakukan konversi nilai piksel pada piksel yang dihapus dari semula bernilai 1 (hitam) menjadi 0 (putih).
5. Perbedaan kecepatan proses juga disebabkan karena banyaknya iterasi (perulangan) yang dilakukan pada algoritma *Hilditch*. Algoritma *Stentiford* hanya melakukan iterasi dengan skala terbanyak bernilai 8 pada tiap pikselnya, dan pada tiap template yang diperbandingakan, penulis hanya melakukan proses sebanyak empat kali (perbandingan piksel utama dengan template, perhitungan nilai *connectivity*, pengecekan status *endpoint*, dan penulisan array).

BAB V
SIMPULAN DAN SARAN

5.1. Simpulan

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa :

1. Penulis menemukan beberapa permasalahan jika perhitungan nilai *connectivity* dilakukan dengan rumus

$$C_n = \sum_{k=1,3,5,7} (P_k - P_k \cdot P_{k+1} \cdot P_{k+2})$$

   Oleh karena itu penulis memodifikasi fungsi tersebut seperti pada bagian implementasi prosedural pada penelitian ini.
2. Dibandingkan dengan algoritma *Hilditch* yang digunakan pada penelitian sebelumnya, algoritma *Stentiford* menunjukkan hasil (*skeleton*) yang lebih tipis dan waktu eksekusi yang lebih cepat. Kedua hal ini disebabkan karena :
   a. Hasil dari proses *thinning* dengan Algoritma *thinning Stentiford* lebih sempurna (tipis) karena algoritma ini melakukan penghapusan secara tidak langsung (*mark-and-delete*), serta melakukan penghapusan secara memutar, penghapusan objek terluar dilakukan terlebih dahulu dengan menggunakan template 3x3 dan syarat-syarat lainnya.
   b. Hasil yang lebih sempurna ini juga disebabkan karena algoritma *Stentiford* memberikan syarat tambahan untuk memperhalus citra yang dihasilkannya.
   c. Proses penghapusan pada algoritma *Stentiford* dalam penelitian ini dilakukan dengan lebih cepat daripada penelitian sebelumnya yang menggunakan algoritma *Hilditch* dikarenakan perbedaan cara implementasi

algoritma antara kedua penelitian. Pada penelitian sebelumnya, algoritma *Hilditch* diimplementasikan dengan proses penggambaran ulang citra secara keseluruhan, jadi, proses penghapusan dilakukan dalam *temporary array* dan penulis selalu menggambar ulang (*redraw*) citra pada tiap iterasi. Sedangkan pada implementasi algoritma *Stentiford* dalam penelitian ini, penulis tidak melakukan penggambaran ulang citra secara keseluruhan, hanya melakukan konversi nilai piksel pada piksel yang dihapus dari semula bernilai 1 (hitam) menjadi 0 (putih).

d. Perbedaan kecepatan proses juga disebabkan karena banyaknya iterasi (perulangan) yang dilakukan pada algoritma *Hilditch*. Algoritma *Stentiford* hanya melakukan iterasi dengan skala terbanyak bernilai 8 pada tiap pikselnya, dan pada tiap template yang diperbandingakan, penulis hanya melakukan proses sebanyak empat kali (perbandingan piksel utama dengan template, perhitungan nilai *connectivity*, pengecekan status *endpoint*, dan penulisan array).

5.2. Saran

Mengingat berbagai keterbatasan yang dialami penyusun terutama masalah pemikiran, waktu dan *tools* pemrograman, maka penyusun menyarankan untuk pengembangan penelitian dimasa yang akan datang sebagai berikut :

1. Banyaknya algoritma *thinning* yang lain yang dapat digunakan untuk proses *thinning* dapat dijadikan alternatif metode lain dalam proses *thinning*.
2. Implementasi Algoritma *thinning Stentiford* dapat lebih maksimal jika perhitungan nilai *connectivity* tidak mengikuti aturan yang semestinya, namun dilakukan secara manual (mengecek nilai

piksel tetangga satu-per-satu) sesuai dengan fungsi yang diterapkan oleh penulis pada penelitian ini.

3. Pada penelitian berikutnya, efisiensi waktu dari proses *thinning* dapat lebih ditingkatkan lagi untuk citra yang berukuran besar dan citra berongga banyak. Keterbatasan pada algoritma *Stentiford* yang selalu melakukan penghitungan pada piksel terluar terlebih dahulu dan melakukan penghapusan piksel dengan cara *mark-and-delete* menyebabkan waktu yang diperlukan untuk proses *thinning* pada citra yang berukuran besar dan citra berongga banyak sangat lama.
4. *Skeleton* yang dihasilkan oleh metode ini tidak sehalus *skeleton* yang dihasilkan oleh metode *morphology MAT* (*Medial Axis Transform*) walaupun waktu pengerjaannya lebih cepat. Untuk itu, diperlukan syarat-syarat tambahan untuk lebih memperhalus *skeleton* yang dihasilkan, yang diharapkan dapat diteliti lebih lanjut pada penelitian selanjutnya.
5. Pada aplikasi implementasi algoritma yang penuyusun buat berbatas pada citra yang berformat bitmap (*.bmp), mungkin selanjutnya bisa dikembangkan untuk semua format citra.

DAFTAR PUSTAKA

[ACH05a] Achmad, Balza dan Kartika Firdausy, Teknik Pengolahan Citra Digital, Ardi Pubhlising, Yogyakarta, 2005.

[MUN04] Munir, Rinaldi, Pengolahan Citra Digital dengan Pendekatan Algoritmik, Penerbit Informatika, Bandung, 2004.

[PRA05] Prayudi, Yudi, Handout Mata Kuliah Pengolahan Citra, Yogyakarta : Teknik Informatika Universitas Isalam Indonesia, 2005.

[RUS06] Rusy Wibawaty, Aghna, 2006, Studi dan Implementasi Algoritma *Thinning* pada Citra Biner, *Tugas Akhir*, tidak diterbitkan. Yogyakarta : Teknik Informatika Universitas Islam Indonesia.

[ACH05b] Achmad, Balza, *Operasi Morphology* [online], Available at http://www.balzach.staff.ugm.ac.id/PengolahanCitra/*Morphology*.pdf, diakses pada tanggal 6 Juni 2007.

[MAR00] Martin, Alberto dan Sabri Tosunoglu, *Image Processing Tehnique for Machime Vision* [online], Available at http://www.eng.fiu.edu/me/robotics/elib/am_st_fiu_ppr_2000.pdf, diakses pada tanggal 6 Juni 2007.

[NN02] NN, *Thinning Algorithm (1)* [online], Available at http://www.massey.ac.nz/~mjjhonsho/notes/59731/presentations/*Thinning*%2520Agorithm.doc, diakses pada tanggal 26 Juni 2007.

[RUP00] Rupard, Jason, *Skeletonization* [online], Available at http://www.unf.edu/~rupj0001/presentations/*Skeletonization*.pdf, diakses pada tanggal 12 Juli 2007.

[HAR03] Harker, A.H. , *Skeletons In The Cupboard – First Steps In Image Analysis* [online], Available at http://www.cmmp.ucl.ac.uk/~ahh/teaching/PHAS2443/*imagesk*.pdf, diakses pada tanggal 29 Juli 2007.