PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK UNTUK SEGMENTASI CITRA DENGAN TRANSFORMASI WATERSHED

TUGAS AKHIR

Oleh :

WASKITHO WIBISONO

NRP. 2694.100.046

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1999

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK UNTUK SEGMENTASI CITRA DENGAN TRANSFORMASI WATERSHED

TUGAS AKHIR

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

Mengetahui / Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. HANDAYANI TJANDRASA, MSc.	NANIK SUCIATI, S.Kom., M.Kom
NIP. 130 532 048	NIP. 132 125 674

SURABAYA
Agustus, 1999

Kupersembahkan untuk orang-orang yang tercinta, alm.Ayahandaku, Ibundaku serta Adik-Adikku Tercinta.

ABSTRAK

Proses pengelompokan data sering dilakukan dalam analisis data. Dengan mengelompokkan data berdasarkan kriteria keseragaman tertentu, maka analisis data akan dapat dilakukan dengan relatif lebih mudah. Demikian pula untuk data citra.

Segmentasi citra adalah proses pembagian citra kedalam bagian-bagian ataupun kelompok-kelompok yang disebut obyek berdasarkan suatu kriteria keseragaman tertentu. Segmentasi adalah salah satu komponen terpenting dalam analisis citra secara otomatis, karena obyek-obyek yang terdapat dalam citra akan digunakan sebagai acuan untuk pengolahan data yang lebih lanjut seperti deskripsi dan interprestasi citra ataupun pengenalan pola.

Transformasi *watershed* memandang citra sebagai sebuah relief topografi dimana intensitas setiap pixel merepresentasikan ketinggian topografinya. Dalam sebuah permukaan topografi, apabila air hujan jatuh diatasnya, sesuai dengan hukum gravitasi maka air tersebut akan mengalir melewati jalur yang lebih rendah sampai ia mencapai ketinggian yang paling rendah atau *minima* dimana ia tidak dapat mengalir kemana-mana lagi.

Himpunan titik-titik pada permukaan topografi citra dimana aliran air yang melewatinya menuju ke *minima* tertentu yang sama, menjadi sebuah *catchment basin* (cekungan yang terisi air) yang berasosiasi dengan *minima* tersebut dan membentuk sebuah region citra.

Watershed terbentuk dilokasi dimana air dari kedua *catchment basin* yang berdekatan bertemu dan merupakan batas dari dua buah *catchment basin* tersebut. Keseluruhan *watershed* yang terbentuk menghasilkan seluruh kontur tertutup yang ada pada citra dan merepresentasikan obyek-obyek dalam citra yang telah tersegmentasi.

KATA PENGANTAR

Bismillahirrohmanirrohiim

Alhamdulillahirobbil'alamiin

Segala puji syukur tertuju kehadirat Allah Swt karena hanya dengan rahmat serta hidayahnya penulis dapat menyelesaikan Tugas Akhir dengan judul :

Perancangan Dan Pembuatan Perangkat Lunak
Untuk Segmentasi Citra Dengan Transformasi Watershed

Tugas akhir ini merupakan sebagian persyaratan kelulusan di jurusan Teknik Informatika, Fakultas Teknologi, Institut Teknologi Sepuluh Nopember Surabaya. Semoga tugas akhir ini membawa manfaat khususnya dalam bidang ilmu pengolahan citra digital serta bidang ilmu informatika pada umumnya.

Pada kesempatan ini, penulis mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Dr.Ir Handayani Tjandrasa, Msc selaku pembimbing I tugas akhir yang telah banyak memberikan bimbingan dalam penyelesaian Tugas Akhir ini.
2. Nanik Suciati , S.Kom, M.Kom selaku pembimbing II tugas akhir yang telah banyak memberikan bimbingan dalam penyelesaian Tugas Akhir ini.
3. Dr.Ir Arif Djunaidy, Msc selaku ketua jurusan Teknik Informatika-ITS.
4. Ir. Aris Tjahyanto, M.Kom selaku dosen wali selama perkuliahan.
5. Seluruh Staff Pengajar dan Staff TU di jurusan Teknik Informatika-ITS.
6. Mas Kadir dan mbak Davi yang telah banyak membantu dan memudahkan penulis dalam mendapatkan referensi di ruang baca T.Informatika-ITS.
7. Almarhum Ayahanda Drs Soedarno yang selalu menjadi inspirasi dan motivasi bagi penulis walaupun telah dipangil menghadap Nya sebelum menyaksikan keberhasilan study putra-putrinya. Ibunda Supartini.Bchk atas dorongan semangat, kasih sayang, serta doa yang selalu menyertai. Adikku Mei, yang selalu membangkitkan semangat untuk berkompetisi dalam meraih prestasi serta meneruskan cita-cita penulis untuk kuliah di Fakultas Kedokteran.

8. Tita Karlita yang selalu memberikan perhatian yang tiada henti, pengertian yang begitu luar biasa, doa dan dukungan yang begitu tulus yang menjadikan segala kesulitan terasa mudah serta hari-hari menjadi begitu indah dan berkesan
9. Rekan-rekan seperjuangan dan sepermainan dan sahabat-sahabatku tercinta, yang selalu menjadi teman penulis dan menjadikan masa-masa kuliah sebagai bagian kehidupan yang sangat indah dalam penemuan jati diri yaitu: Beni M-ITS sahabat sejati, Setyohadi EL-ITS rekan seperjuangan sejak di STAN, , Anjeb konsultan kuliah yang telah mendahului lulus. Adik Boys rekan diskusi penulis untuk memahami kehidupan. Anton Jovi, konsultan dan rekan setia dalam diskusi baik ilmiah ataupun non ilmiah " life is always fair". Yetty , moderator TA dan rekan KP di ARCO. Jody Jabier, terimakasih atas "*triger*" mautnya untuk menyelesaikan TA. Deni "*viva MU* ", Firman "*fans setia Liverpool*", Firdaus, Hamsy , Pratomo , Heri, Riza , Seger, Rahmat Itong "*FIFAnya diganti Delphi Tong*"., Bambang, Herlambang,Mbah Noer,Tony, Wahyudi, Bang Ucok "*Jgn menyerah Bang, cepet lulus*" . Agus 'oedjoe' ulum, Yoyok "*ingat kampus Yok*". Shanti adiknya "*Beckham*", Intan '*Wek*", Yanuar Kusnadi, Hadi,Eddy, Sutama, Fajar, Arya ,Agus Bagiartha, Gusmang.
10. Segenap pengurus, penghuni, penggembira yang menjadi bagian dari keluarga besar laboratorium AJK T.Informatika ITS khususnya kepada perintis, pejuang dan pendiri kepengurusan lab AJK sejak tahun 1996-1999 yang menjadikan kehidupan di AJK begitu "*fantastis*" dan "*unforgetable*".
11. Segenap keluarga di Mojoklanggru Lor 63,Bu Kusno, Mas Boy, Mas Farid, Into, dan sekali lagi Adik Boys.
12. Seluruh rekan-rekan yang karena keterbatasan tempat belum tersebut disini.
13. Rekan-rekan kesebelasan C0A, tim terkuat yang pernah ada di TC.

Surabaya, 18 Agustus 1999

Penulis.

DAFTAR ISI

DAFTAR GAMBAR

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Penggunaan pengolahan citra digital berkembang pesat sejalan dengan berkembang dan memasyarakatnya teknologi komputer di berbagai bidang. Diantaranya di bidang kesehatan, bidang teknologi industri , pemetaan geografis dan bidang-bidang lainnya.

Bidang-bidang tersebut membutuhkan berbagai macam aplikasi pengolahan citra digital termasuk untuk aplikasi pembagian citra menjadi region-region yang sama untuk sehingga dapat dilakukan interprestasi terhadapnya. Proses tersebut dinamakan segmentasi citra.

Segmentasi citra adalah sebuah proses yang membagi sebuah citra kedalam region-region yang terpisah, dimana setiap region adalah homogen dan mengacu kepada sebuah kriteria keseragaman yang jelas. Secara lebih lengkap tujuan dari segmentasi citra ialah untuk memisahkan komponen dari suatu citra kedalam bagian-bagian yang berhubungan dengan obyek fisik dari gambar tersebut. Komponen komponen yang telah disegmentasi tersebut kemudian digunakan oleh proses yang lebih lanjut untuk interprestasi terhadap citra ataupun pengenalan pola.

Segmentasi yang dilakukan terhadap citra tersebut harus tepat agar informasi yang terkandung didalamnya dapat diterjemahkan dengan baik

Pemilihan metode yang digunakan untuk melakukan segmentasi sangat berpengaruh terhadap hasil yang didapatkan .

Transformasi Watershed merupakan metode yang efektif dan handal untuk melakukan segmentasi citra. karena tepi-tepi yang dihasilkan tipis dan tidak terputus-putus sehingga kontur yang didapatkan merupakan region tertutup. Hal ini didukung pula oleh kemampuannya untuk mendeteksi kontur-kontur yang kecil dan kompleks sehingga sangat tepat untuk menyelesaikan permasalahan diatas.

1.2 Tujuan

Tujuan dari penelitian yang dilakukan adalah membuat perangkat lunak yang mampu mendapatkan kontur tertutup dari sebuah citra yang merupakan representasi dari obyek-obyek yang ada dalam citra dengan metode Transformasi Watershed.

1.3 Perumusan Masalah

Untuk memecahkan permasalahan diatas perlu dilakukan pemecahan terhadap permasalahan-permasalahan sebagai berikut :

1. Proses reduksi *noise* untuk meminimalkan noise-noise yang ada di citra.
2. Penentuan keseluruhan *local minima* dari citra serta *gradien slope* dari pixel yang bukan merupakan *local minima.*

3. Penelusuran setiap *gradien slope* piksel untuk menghasilkan region terlabeli
4. Penggabungan region-region terlabeli dengan kriteria penggabungan region yang tepat.

1.4 Pembatasan Masalah

Input citra yang diproses merupakan citra digital dalam format file gambar BMP (*Windows Bitmap*). Citra yang diproses dalam program ini merupakan citra graylevel, dengan level keabuan 256.

Untuk citra berwarna (bukan graylevel) dapat juga diproses. Namun sebelumnya warna yang ada dikonversikan kedalam citra graylevel.

1.5 Metodologi Penelitian

Metode penelitian yang digunakan dalam penyusunan tugas akhir ini adalah sebagai berikut :

1. Studi Literatur
2. Perumusan masalah dan perumusan penyelesaiannya.
3. Perancangan perangkat lunak.
4. Pembuatan perangkat lunak per modul.
5. Pengujian secara keseluruhan.
6. Evaluasi dan revisi perangkat lunak.
7. Penulisan naskah Tugas Akhir.

1.6 Sistematika Pembahasan

Dalam naskah tugas akhir ini pembahasan dibagi menjadi beberapa bab sebagai berikut :

- Bab I merupakan pendahuluan yang menerangkan latar belakang dan tujuan pembuatan tugas akhir ini, permasalahan dan pembatasan permasalahannya serta metodologi yang dipakai untuk menyelesaikannya.
- Bab II menjelaskan teori dasar citra digital serta teori-teori lain yang mendukung pemecahan permasalahan.
- Bab III menguraikan tentang definisi trransformasi watershed , pendekatan-pendekatan yang ada dalam transfomasi watersed serta implementasinya untuk segmentasi citra.
- Bab IV menguraikan desain perancangan dan pembuatan perangkat lunak yang merupakan implementasi dari bab-bab sebelumya.
- Bab V menjelaskan hasil pengujian yang telah dilakukan terhadap citra sampel yang ada.
- Bab VI menguraikan kesimpulan dan saran dari tugas akhir yang telah dibuat.

BAB II

TEORI PENUNJANG

BAB II

TEORI PENUNJANG

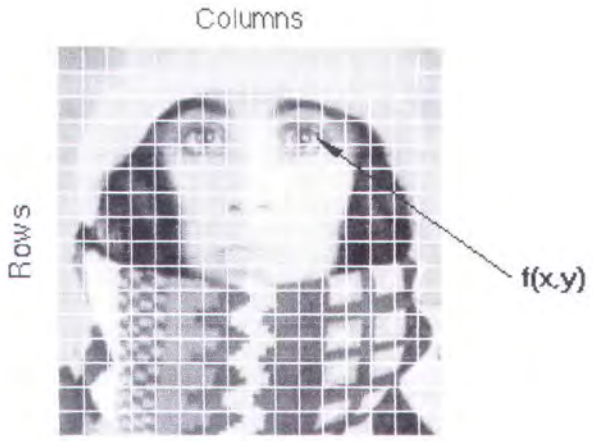
Pembahasan pada bab ini menguraikan beberapa konsep dan teori-teori dasar yang digunakan untuk menunjang ataupun berhubungan erat dengan teori dan implementasi dalam tugas akhir ini. Hal ini perlu diketahui terlebih dahulu sebelum membahas bab-bab selanjutnya.

2.1 Pemodelan Citra

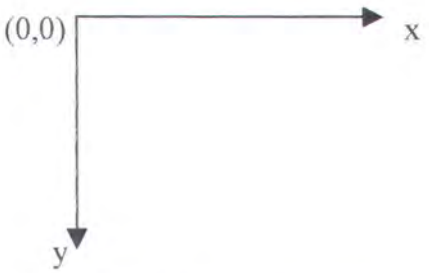
Kata citra (*image*) bisa diartikan sebagai "suatu produksi atau bentuk imitasi dari sesuatu objek". Dapat juga citra dinyatakan sebagai kumpulan atau koleksi dari sinyal-sinyal, seperti sinyal dua dimensi (2-D) yang menunjukkan warna dari citra.

Dalam hal ini penulis mendefinisikan kata citra sebagai suatu fungsi dua dimensi $f(x,y)$, dimana $f(x,y)$ adalah fungsi yang bernilai *real* yang menyatakan intensitas atau tingkat kecerahan. Jika variabel x dan y kontinyu , citra tersebut adalah citra *analog* atau kontinyu. Jika x dan y disampling, fungsi menjadi diskrit, dimana x dan y menyatakan index dari ruang koordinat spasial (*spatial coordinates* atau disebut juga ruang koordinat kartesian). Jika $f(x,y)$ dikuantisasi (dalam hal ini hanya himpunan berhingga dari beberapa nilai), citra akan dikatakan digital.

Ilustrasi mengenai sebuah citra digambarkan dalam gambar 2.1, dan konvensi arah koordinat ditunjukkan dalam gambar 2.2.



Gambar 2.1 Representasi Citra Digital.



Gambar 2.2 Konvensi arah koordinat.

Citra yang kita lihat atau kita amati dalam keseharian penglihatan kita, adalah merupakan pemantulan cahaya dari obyek. Pada dasarnya *f(x,y)* dapat dikarakteristikan menjadi dua komponen. Satu komponen merupakan jumlah atau intensitas cahaya yang dikenakan pada obyek, sementara komponen yang lain merupakan jumlah cahaya yang dipantulkan oleh obyek tersebut.

Citra digital (*digital image*) adalah citra kontinyu *f(x,y)* yang didiskritkan baik koordinat spasial maupun tingkat kecerahannya. Kata kontinyu disini menjelaskan bahwa indeks *x, y* dapat bernilai *real* sedangkan kata diskrit

menjelaskan bahwa indeks x,y hanya bernilai integer. Untuk dapat diproses dengan komputer, fungsi citra *f(x,y)* harus didigitasi atau didiskritkan baik koordinat spasial maupun tingkat kecerahannya. Digitisasi terhadap koordinat spasial (x,y) disebut sebagai image sampling sedangkan digitisasi terhadap tingkat kecerahan atau *amplitudo* disebut *sebagai gray-level quantization.* Kita dapat menganggap citra digital (citra yang sudah didigitisasi, yang untuk seterusnya disingkat dengan citra) sebagai matriks dengan ukuran M X N yang basis dan kolomnya menunjukkan titik-titiknya, sedangkan nilai dari elemen matriks sebagai tingkat keabuan atau warna dari titik tersebut. Elemen dari matriks dua dimensi disebut dengan elemen citra (*image element*), elemen gambar (*picture element*), pixel atau pel. Dua istilah terakhir mengacu pada *picture element.*

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & \dots & f(0,N) \\ f(1,0) & f(1,1) & \dots & \dots & f(1,N) \\ \vdots & & & & \\ f(M,0) & f(M,1) & \dots & \dots & f(M,N) \end{bmatrix}$$

Gambar 2.3 Contoh matriks citra

2.2 Hubungan Antar Pixel

Hubungan antar pixel pada citra digital merupakan hal yang sangat penting karena pixel-pixel tersebut merupakan dasar utama dalam pengolahan citra digital.

Dalam bagian sebelumnya telah disebutkan bahwa sebuah citra akan dinotasikan dengan *f(x,y)*. Sedang untuk pixel tertentu yang spesifik akan dinotasikan dengan huruf kecil, seperti *p* atau *q*, dan subset dari pixel-pixel akan dinotasikan dengan huruf besar yang tebal, seperti **S** atau **V**.

2.2.1 Pixel Tetangga

Suatu pixel *p* pada koordinat *(x,y)* mempunyai 4 tetangga pada arah horisontal dan vertikal dimana koordinatnya adalah sebagai berikut

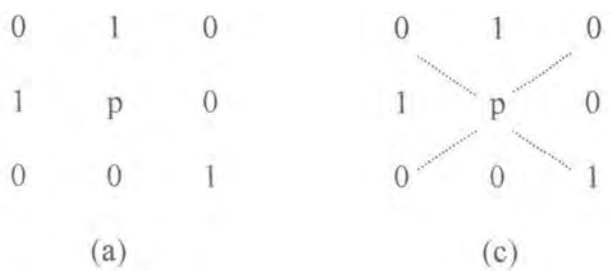
$$(x + 1, y), (x - 1, y), (x, y + 1), (x, y-1) \qquad (2.1)$$

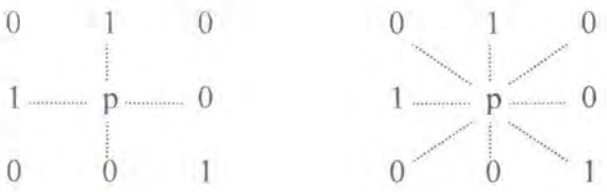
kumpulan dari pixel-pixel ini disebut " 4-tetangga" dari *p*, yang dinotasikan dengan $N_4(p)$.

Selain tetangga horisontal pixel *p* juga mempunyai empat tetangga diagonal sebut dengan "diagonal tetangga" yang koordinat digambarkan sebagai berikut :

$$(x + 1, y + 1), (x + 1, y - 1), (x - 1, y + 1), (x - 1, y - 1) \qquad (2.2)$$

yang dinotasikan dengan $N_8(p)$. Jika pixel "4-tetangga" dan "diagonal tetangga" digabungkan, maka disebut dengan "8-tetangga" dari *p*. Dinotasikan dengan $N_8(p)$.





(b) (d)

Gambar 2.4

(a) matrik dari sebuah citra.

(b) $N_4(p)$ untuk tetangga horisontal dan vertikal.

(c) $N_4(p)$ untuk tetangga diagonal.

(d) $N_8(p)$.

2.2.2. Keterhubungan (*connectivity*)

Konsep keterhubungan antara pixel-pixel adalah penting, digunakan untuk menentukan batas-batas dari obyek-obyek atau komponen-komponen pada gambar untuk menentukan apakah dua pixel saling terhubung, kita harus menentukan bagaimana cara pixel itu dalam berdekatan (misal berdekatan dengan "4-tetangga") dan jika graylevel dari pixel-pixel itu memenuhi kriteria "sama" tertentu (misal : jika gray-level sama persis). Jadi ada kemungkinan dalam suatu gambar nanti ada pixel yang berdekatan dengan "4-tetangga", tetapi mereka dikatakan tidak terhubung, karena tidak mempunyai graylevel yang sama.

Anggap **S** adalah kumpulan dari nilai graylevel yang digunakan untuk menentukan keterhubungan antar pixel, dan hanya pixel-pixel dengan nilai intensitas 7, 8, 9 dan 10 yang diperhatikan, maka **S** = {7, 8, 9, 10}.

Beberapa contoh jenis keterhubungan, antara lain :

(a) Keterhubungan - 4.

Dua pixel p dan q dengan nilai dengan nilai intensitas ada di **S** adalah termasuk jenis ketergantungan ini jika, q ada di set $N_4(p)$.

(b) Keterhubungan -8.

Dua pixel p dan q dengan nilai intensitas ada di **S** adalah termasuk jenis keterhubungan ini jika, q ada di set $N_8(p)$.

Pixel p adalah berdekatan/bersebelahan (*adjacent*) dengan pixel q jika mereka saling terhubung. Kita mungkin mendefinisikan keterhubungan-4, keterhubungan-8 , atau keterhubungan-c, kedekatan dari pixel, tergantung pada jenis keterhubungan yang ditentukan. Dua subset $\mathbf{S_1}$ dan $\mathbf{S_2}$ dari suatu gambar adalah berdekatan, jika ada beberapa pixel pada $\mathbf{S_1}$ yang berdekatan dengan beberapa pixel dari $\mathbf{S_2}$.

Sebuah jalur (*path*) dari pixel *p* dengan koordinat (x,y) ke pixel *q* dengan koordinat (s,t) adalah sebuah urutan dari pixel-pixel yang berbeda dengan koordinat

$$(x_0 y_0), (x_1 y_1), \ldots \ldots \ldots x_n y_n),$$

dimana $(x_0, y_0) = (x y)$ dan $(x_n, y_n) = (s,t)$, (x_i, y_i) bersebelahan dengan (x_{i-1}, y_{i-1}), untuk $1 \leq i \leq n$, dan n adalah panjang dari jalur. Kita bisa mendefinisikan 4-,8-, atau c-jalur, tergantung dari jenis keterhubungan yang dibutuhkan.

Jika *p* dan *q* adalah pixel pada gambar dengan subset **S**, maka *p* terhubung ke *q* dalam **S** jika ada jalur dari *p* ke *q* yang jalurnya terdiri dari semua pixel yang ada di **S**. Untuk setiap pixel *p* di **S**, kumpulan pixel di **S** yang terhubung ke *p*

disebut "komponen-terhubung" (*connected component*) dari S. Karena itu setiap dua pixel dari sebuah "komponen terhubung" adalah terhubung satu sama lainnya, dan "komponen-terhubung" yang berlainan adalah saling lepas (*disjoint*).

2.3 Deteksi Garis Tepi (*Edge Detection*)

Tepi (*edge*) merupakan dasar dari vitur (*feature*) citra. Tepi membawa informasi yang sangat berguna tentang batas-batas dari sebuah objek, yang sangat berguna dalam analisa sebuah citra (*image analysis*).

Garis tepi merupakan batas antara dua daerah yang memiliki tingkat keabuan yang berbeda. Diasumsikan bahwa kedua daerah tersebut memiliki sifat yang cukup sama sehingga transisi diantaranya dapat ditentukan berdasarkan *diskontinuitas* tingkat keabuan masing-masing daerah.

Pemisahan gambar menjadi obyek dan latarnya merupakan tahap utama dalam penafsiran citra. Pengenalan batas atau tepi antara beberapa bagian gambar diadopsi dari cara manusia mengenali obyek-obyek benda yang dilihat. Tepi umumnya ditemukan pada tempat-tempat pada citra yang mengalami perubahan intensitas yang menyolok. Dalam pandangan matematis, pada citra harus dilakukan operasi derivatif untuk mengidentifikasi diskontinyuitas dalam nilai-nilai intensitas. Proses ini menjadi kompleks karena dua alasan. Pertama, setiap perubahan intensitas pada titik-titik berdekatan bisa dinyatakan sebagai tepi karena tidak ada perbedaan yang tegas antara tepi dan perubahan tajam gradien. Oleh sebab itu, harus disediakan sebuah level threshold yang dipergunakan untuk

menentukan perubahan gradien mana yang benar-benar merupakan tepi. Kedua, citra dinyatakan dalam dua dimensi dan tepi-tepi selain memiliki magnitude juga memiliki arah. Detektor tepi harus bisa menangani tepi-tepi yang muncul dalam berbagai arah pada input citra.

Pada umumnya, ide yang mendasari kebanyakan aplikasi mengenai deteksi garis tepi adalah dengan perhitungan dari operator *local derivate*.

Derivasi kedua bernilai positif untuk bagian transisi yang berhubungan dengan bagian gelap dari garis tepi, negatif untuk bagian transisi yang berhubungan dengan bagian terang dari garis tepi, dan juga nol untuk daerah yang memiliki tingkat keabuan konstan.

Derivasi pertama untuk sembarang titik pada citra ditentukan dengan menggunakan *magnitude* dari *gradien* pada titik tersebut. Secara umum, *gradien* dari sebuah citra *f(x,y)* pada lokasi (*x,y*) adalah vektor yang dapat dinyatakan dalam persamaan derivatif spasial berorientasi arah sebagai berikut:.

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\delta f(x,y)}{\delta x} \\ \dfrac{\delta f(x,y)}{\delta y} \end{bmatrix} \tag{2.3}$$

Metode sederhana untuk menghitung derivasi persamaan (2.3) dapat menggunakan pendekatan beda depan (*forward difference*) antar dua titik , yaitu:

$$\frac{\delta f}{\delta x}(x,y) \approx \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x} \tag{2.4}$$

$$\frac{\delta f}{\delta y}(x,y) \approx \frac{f(x,y+\Delta y)-f(x,y)}{\Delta y} \qquad (2.5)$$

Fungsi persamaan (2.4) dan (2.5) dapat diimplementasikan dengan cara mengkonvolusikan input citra dengan mask sebagai berikut :

$$[\,-1 \quad 1\,] \qquad \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Gambar 2.5 mask gradien

Pendekatan ini hanya mengestimasi dengan baik *gradien* terpusat antar titik tapi tidak menghitung *gradien* terhadap titik-titik lainnya yang termasuk dalam area itu.

Sudah menjadi hal yang umum dalam analisis vektor bahwa vektor *gradien* menunjuk pada arah perubahan nilai tertinggi untuk f pada (x,y). Pada deteksi garis tepi, kuantitas yang penting adalah *magnitude* dari vektor ini, secara umum disebut dengan magnitude gradien (*gradient magnitude*) atau ∇f. Sehingga intensitas *gradien* atau ∇f untuk gambar 2.2 pada lokasi f_{11} diberikan oleh persamaan :

$$\forall f = mag(\forall f) = [G_x^{\,2} + G_y^{\,2}]^{1/2} \qquad (2.6)$$

Arah dari vektor *gradien* juga merupakan faktor yang penting. Misalkan $\alpha(x,y)$ merepresentasikan arah sudut untuk vektor ∇f pada (x,y). Kemudian, dari analisis vektor, didapatkan :

$$\alpha(x,y) = \tan^{-1}(\frac{G_y}{G_x}) \qquad (2.7)$$

dimana sudut tersebut dihitung terhadap sumbu x.

Dari penjelasan diatas didapatkan dua buah hasil dari proses deteksi garis tepi, yaitu : *magnitude* untuk *gradien* intensitas pada citra, dan arah sudut dari gradien tersebut.

2.4 Perbaikan Citra

Tujuan akhir dari teknik *enhancement* (*enhancement technique*) adalah untuk memproses gambar yang diberikan sehingga menghasilkan gambar yang lebih sesuai dari gambar aslinya untuk aplikasi tertentu [11],[13]. Kata tertentu ini mempunyai arti bahwa tidak semua gambar apabila diproses menghasilkan yang lebih sesuai dalam arti sesuai dengan keinginan.

Teknik *enhancement* pada dasarnya adalah prosedur *Heuristic* (Suatu metoda pemecahan masalah dengan mengevaluasi pengalaman-pengalaman atau percobaan-percobaan yang pernah dilakukan, lalu melangkah ke suatu pemecahan dengan mencoba-coba) yang dibuat untuk memanipulasi sebuah citra supaya menambah aspek psikologi dari sistem penglihatan manusia. Contohnya, merubah

contrast streching (tingkat kekontrasan), merubah *brighteness streching* (tingkat kecerahan) adalah termasuk teknik *enhacement* karena ini dasar awalnya adalah pada aspek kesenangan yang akan disajikan pada yang melihatnya, sementara menghilangkan kekaburan dari suatu citra dengan mengaplikasikan proses kebalikan dari penyebab terjadinya kekaburan itu termasuk pada permasalahan restorasi.

Dalam tugas akhir ini, perbaikan citra yang digunakan adalah dengan reduksi *noise* dengan metode pengurangan *noise* secara linier (*Linier Noise Cleaning*).

2.4.1 *Average Filter* (Filter Rata-Rata)

Reduksi *noise* citra yang digunakan dalam tugas akhir ini menggunakan filter *mean*, dikenal juga dengan nama filter *average* atau rata-rata. Ide dasar dari filter *mean* adalah sederhana, dengan mengganti setiap intensitas *pixel* citra dengan rata-rata (*average*) dari nilai *pixel* tetangganya, termasuk *pixel* itu sendiri. Metode ini memiliki efek yang mengeliminasi *pixel-pixel* disekitarnya yang intensitas *pixelnya* tidak representatif. Filter *mean* digunakan sebagai filter konvolusi. Seperti filter-filter konvolusi lainnya, filter ini berbasis pada sebuah *kernel* yang merepresentasikan ukuran dan bentuk tetangganya sebagai sampel dalam penghitungan mean.

Filter yang sering digunakan adalah 3×3 seperti yang ditunjukkan dalam gambar 2.6. Walaupun kernel yang besar (mis: 5x5) dapat digunakan juga untuk

mendapatkan hasil yang lebih halus. Filter yang kecil dapat digunakan lebih dari sekali untuk menghasilkan ukuran kernel yang sama dengan kernel yang lebih besar, akan tetapi efeknya tidak akan sama bila dibandingkan dengan menggunakan kernel yang besar.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Gambar 2.6 Filter Mean

Contoh hasil proses reduksi noise dengan menggunakan filter mean ditunjukkan dalam gambar 2.7 berikut

(a)

(b)

Gambar 2.7 (a) Citra yang bernoise , (b) Hasil reduksi noise dengan filter mean

2.5 Matematika Morphologi Citra

Matematika morphologi merupakan metode yang sangat handal dalam menganalisa dan mendeskripsikan bentuk (*shape*) geometris. Mertode ini menggunakan pendekatan teori himpunan terhadap analisa citra [12]. Mengacu kepada prinsip-prinsip matematika morphologi citra, sebuah citra tidak memiliki informasi jika citra tersebut tidak diobservasi dengan sebuah observator. Observator ini melakukan pengujian terhadap obyek dan mengekstraksi informasi yang dibutuhkan untuk penganalisaan dan pendeskripsian struktur. Observator didapatkan dengan menggunakan himpunan sederhana (*mis : rhombus, square*) yang didefinisikan sebagai elemen penstrukturan (*structuring element*) [11,13]. Gambar 2.8 berikut menunjukkan contoh elemen penstrukturan

(a) rhombus (b) square

gambar 2.8 Elemen Penstrukturan

fungsi dari elemen penstrukturan ini dinyatakan dengan *g(x)*.

Pada mulanya matematika morphologi citra ini hanya diterapkan terhadap citra biner (*binary image*) . Penjelasan mengenai hal ini secara mendalam telah dilakukan oleh W.K. Pratt dalam buku Digital Image Processing [11]. Penjelasan

lebih lanjut dalam sub-bab ini adalah tentang teori matematika morphologi citra graylevel.

2.5.1 Matematika morphologi citra gray level

Matematika morphologi citra biner dapat dikembangkan menjadi matematika morphologi untuk citra graylevel. Dalam citra biner, efek dari proses pelebaran *(dilasi)* dan pengikisan (*erosi*) mengakibatkan citra menjadi membesar atau mengecil dalam bidang spasial [13]. Untuk citra graylevel hal ini tidaklah demikian. Pelebaran ataupun pengikisan terhadap citra tidak mengakibarkan pelebaran ataupun pengikisan terhadap citra graylevel. Penjelasan mengenai hal ini di jelaskan dalam sub-bab berikut.

2.5.1.1 Dilasi pada Citra Graylevel

Dilasi pada citra Graylevel dari fungsi *f(x)* oleh elemen penstrukturan *g(x)* dilambangkan dengan notasi $\oplus$ dan didefinisikan sebagai berikut

$$[f \oplus g](x) = \max_{z \in D, z-x \in G} \{f(z) + g(z-x)\} \tag{2.8}$$

2.5.1.2 Erosi pada Citra Graylevel

Erosi pada citra Graylevel dari fungsi *f(x)* oleh elemen penstrukturan *g(x)* dilambangkan dengan notasi Θ dan didefinisikan sebagai berikut

$$[f(x)] = \min_{z \in D, z-x \in G} \{f(z) - g(z-x)\} \tag{2.9}$$

2.5.2 Operasi Open

Operasi open merupakan operasi ganda yang terdiri dari proses erosi dan dilanjutkan dengan proses dilasi dan didefinisikan sebagai berikut

$$f_g(x) = [(f \Theta g) \oplus g](x) = [f(x) \Theta g(-x)] \oplus g(x) \qquad (2.10)$$

2.5.3 Operasi Close

Operasi Close merupakan operasi ganda yang terdiri dari proses dilasi yang dilanjutkan dengan proses erosi dan didefinisikan sebagai berikut

$$f^g(x) = [(f \oplus g) \Theta g](x) = [f(x) \oplus g(-x)] \Theta g(x) \qquad (2.11)$$

2.6 Segmentasi Citra

Definisi umum segmentasi adalah proses yang membagi suatu citra kedalam bagian-bagian atau kelompok-kelompok yang mempunyai persamaan sifat atau ciri-ciri. Ada dua pedekatan dalam proses segmentasi yaitu: Edge based dan Region based. Pada pendekatan yang pertama, dilakukan terlebih dahulu deteksi tepi lokal (lokal diskontinuitas), yang selanjutnya dari tepi-tepi lokal tersebut digabung untuk membentuk satu segmen penuh. Sedangkan pendekatan yang kedua didasarkan pada kesamaan antara region, contohnya adalah tresholding, region growing, region spliting dan merging.

2.7. Kernel Gaussian

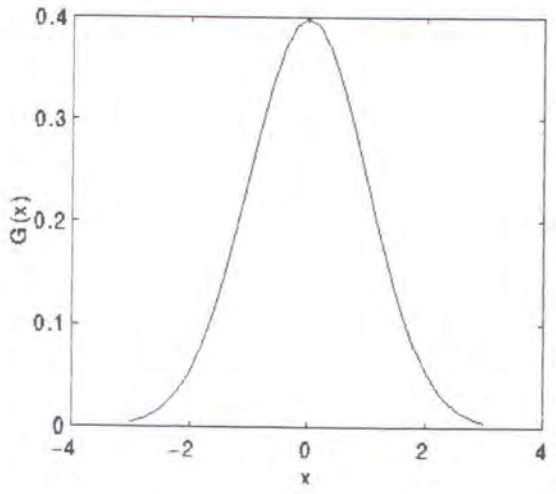
Operator *kernel Gaussian* adalah operator *konvolusi* 2-D yang digunakan untuk mengaburkan (*blur*) citra dan mengubah nilai-nilai intensitas pixel kedalam nilai-nilai *floating point*. Hal lainnya adalah kemampuannya untuk menghilangkan noise yang ada pada citra.

Distribusi gaussian 1 dimensi dihasilkan dengan persamaan sebagai berikut

$$F(x) = \frac{1}{\sqrt{2\Pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \qquad (2.12)$$

dimana σ merupakan *standar deviasi* dari distribusi. Distribusi gaussian mempunyai *mean* sama dengan 0 yang merupakan pusat (*center*) dari distribusi. Distribusi gaussian sering juga disebut dengan distribusi normal.

Ilustrasi terhadap distribusi gaussian 1-dimensi diilustrasikan dalam gambar 2.9 berikut :



Gambar 2.9 Distribusi Gaussian 1-D dengan mean 0 dan σ =1

Fungsi distribusi Gaussian 2-D didefinisikan dengan persamaan sebagai yang didefinisikan sebagai berikut

$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp^{\left(\frac{-\left(x^2+y^2\right)}{2\sigma^2}\right)} \tag{2.13}$$

Citra dikaburkan (blur) dengan menggunakan fungsi Gaussian dua dimensi (persamaan 2.13). Proses blur citra ini dilakukan dengan operasi konvolusi menggunakan mask dua dimensi yang diperoleh dengan mendiskritkan fungsi kontinyu Gaussian dua dimensi. Bila fungsi gambar input adalah $f(x,y)$ dan mask Gaussian dua dimensi adalah $m_G(i,j)$ maka output konvolusi yang dihasilkan adalah:

$$O_{G_{xy}}(x,y) = \sum_{i=-\frac{M}{2}}^{\frac{M}{2}} \sum_{j=-\frac{M}{2}}^{\frac{M}{2}} f(x-i,y-j)m_G(i,j) \tag{2.14}$$

dengan asumsi mask m mempunyai ukuran yang sama yaitu M pada masing-masing dimensi. Namun proses konvolusi menggunakan *kernel Gaussian* dua dimensi membutuhkan waktu yang lama, karena konvolusi ini membutuhkan M^2 perkalian, ditambah M^2-1 penjumlahan terhadap setiap pixel pada citra input.

Oleh karena itu dalam praktek biasanya konvolusi penghalusan gambar inimenggunakan dua buah kernel Gaussian satu dimensi, satu menurut arah horisontal:

$$O_{G_x}(x,y) = \sum_{i=-\frac{M}{2}}^{\frac{M}{2}} f(x-i,y)m_G(i) \tag{2.15}$$

dilanjutkan menurut arah vertikal:

$$O_{G_{xy}}(x,y) = \sum_{i=-\frac{M}{2}}^{\frac{M}{2}} f(x,y-i)\, m_G(i) \qquad (2.16)$$

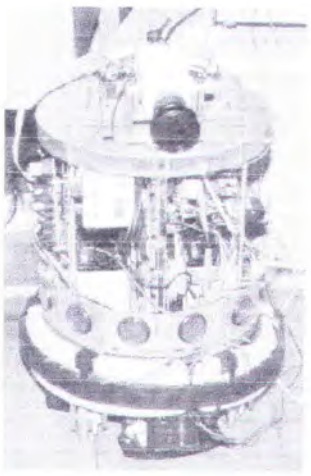
sehingga untuk setiap pixel pada citra input, operasi penghalusan gambar ini membutuhkan $2M$ perkalian, ditambah $2M$-2 penjumlahan, di mana m_G :

$$m_G(i) = \frac{1}{\sqrt{2\Pi}\sigma} \exp\left(\frac{-i^2}{2\sigma^2}\right) \qquad (2.17)$$

adalah mask hasil diskritisasi fungsi Gaussian satu dimensi:

$$G(x) = \frac{1}{\sqrt{2\Pi}\sigma} \exp\left(\frac{-x^2}{2\sigma^2}\right) \qquad (2.18)$$

Gambar 2.10 berikut menunjukkan efek dari konvolusi dengan kernel dengan σ =4 , ukuran kernel adalah 15x15



Gambar 2.10 (a) Citra asli (b) citra hasil *blurring*

BAB III

DEFINISI DASAR TRANSFORMASI WATERSHED

BAB III

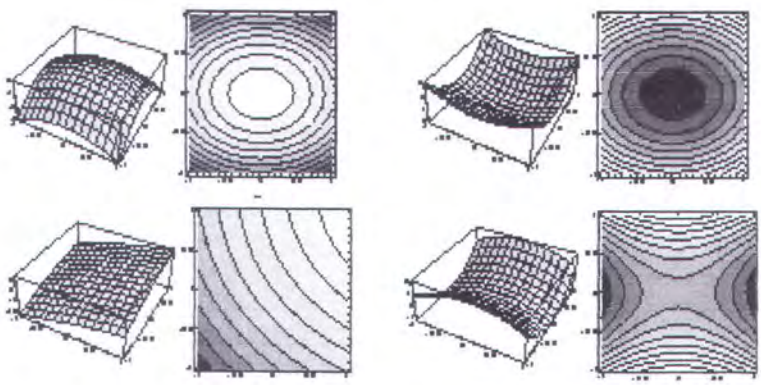
DEFINISI DASAR TRANSFORMASI WATERSHED

Pembahasan mengenai Watershed (aliran air) adalah salah satu permasalahan klasik dalam bidang topografi. Sebagai contoh nyata tentang hal tersebut adalah adanya dataran tinggi yang membagi Amerika kedalam dua bagian (*region*) yang berbeda. *Region* sebelah barat yang berbatasan langsung dengan samudera Atlantic serta *region* sebelah timur yang berbatasan dengan samudera Pasifik. Apabila terjadi hujan yang deras di kedua region tersebut, maka air hujan yang jatuh di region di sebelah barat akan mengalir turun mengikuti hukum gravitasi hingga mencapai samudera Atlantic. Sementara air hujan yang jatuh di region di sisi timur akan mencapai samudera Pasifik. Dalam definisi yang penulis jelaskan lebih lanjut di bab ini, dataran tinggi yang membagi Amerika ke dalam dua region tersebut didefinisikan sebagai w*atershed line* (*batas watershed*). Dua *region* yang terpisahkan oleh dataran tinggi tersebut didefinisikan sebagai *catchment basin* dari samudera Atlantic dan Samudera Pasifik. Kedua samudera diatas didefinisikan sebagai minima yang berasosiasi dengan catchment basinnya masing-masing. Lebih jelasnya, *watershed* yaitu suatu zona yang membatasai dua buah *catchment basin* (cekungan yang terisi air) yang berdekatan [5][6].

Dalam bidang pengolahan citra digital, fenomena *watershed* ini di bahas dan secara lebih khusus dalam sub-bab matematika morphologi citra.. Dalam pembahasannya, graph dari intensitas *graylevel* citra dapat dianggap sebagai relief

topografi, intensitas graylevel dari citra menunjukkan ketinggian topografinya [1],[2],[3],[5],[6]. Representasi ini digunakan untuk mengapresiasikan secara baik fenomena *watershed* dalam bidang topografi sebagai sebuah transformasi yang dilakukan terhadap citra.

Ilustrasi terhadap konsep topografi dalam morphologi citra tersebut ditunjukkan oleh gambar 3.1 dibawah ini.



Gambar 3.1. Representasi topografi citra.

Dalam gambar 3.1 diatas , digambarkan relief topografi dari 4 buah citra graylevel. Sebelah kiri menggambarkan relief topografi citra, sebelah kanannya menunjukkan intensitas gray levelnya, garis-garis tambahan ditambahkan untuk memperjelas perbedaan antar intesitas. Intensitas gray level citra di setiap lokasi pixel dalam citra tersebut dijadikan acuan untuk merepresentasikan relief topografinya.

Konsep watershed sebagai sebuah metode untuk melakukan segmentasi terhadap citra pertama kali diperkenalkan oleh C. Lantuejoul dan Sergei Beucher [16] dan dikembangkan oleh Jean Serra [15].

Segmentasi dengan menggunakan Transformasi Watershed sangat efektif dikarenakan tepi-tepi yang dihasilkan tipis ,tidak terputus-putus serta region yang dihasilkan *(close)* tertutup sehingga segmentasi yang dihasilkan sangat memuaskan [4],[14].

3.1 Transformasi Watershed

3.1.1 Definisi Dasar

Sebuah citra digital gray level adalah sebuah fungsi f : D $\rightarrow$ N dimana domain dari citra dinyatakan dinyatakan sebagai $D_I \subset Z^n$, n merupakan nilai integer postitif $f(p)$ menyatakan intensitas graylevel dari pixel p dimana $p \in D$.

$$D_f \subset Z^n . \rightarrow \{0,1 \ldots . ,N\} \qquad (3.1)$$

$$p \rightarrow f(p).$$

Dimisalkan G merupakan *grid digital* yang dapat berupa *grid square* dengan 4 atau 8 keterhubungan, atau *grid hexagonal* dengan 6 keterhubungan. $N_G(p)$ merupakan himpunan pixel pixel tetangga p yang mengacu ke G, untuk selanjutnya pixel pixel tetangga dari p tersebut dinyatakan sebagai $p' \in N_G(p)$.

3.1.2. Definisi Lower Slope .

Lower slope dalam topografi menunjukkan dataran yang paling rendah disekitar lokasi yang menunjukkan arah aliran air dari lokasi tersebut menuju minima terdekat [1],[4]. Dalam transformasi watershed, lower slope akan menghasilkan gradien dari tiap-tiap pixel yang menjadi acuan dari proses rainfalling untuk pixel tersebut . Rasio aksimal dari $I(p) - I(p')$ terhadap keseluruhan pixel tetangga p yang memiliki nilai gray level yang lebih rendah dari p yang dinyatakan sebagai $\Gamma(p)$ .Nilai maksimal *slope* dari fungsi f di lokasi $p \in D$ didefinisikan sebagai [1],[2],[3].

$$\mathrm{LS(p)} = \max_{\forall p' \in N(p)} \left(\frac{f(p) - f(p')}{dist(p, p')} \mid f(p') \leq f(p) \right) \qquad (3.2)$$

definisi diatas berlaku untuk $I(p') > I(p)$, $\forall p' \in N(p)$

3.1.3 Definisi Cost Lower Slope

Biaya untuk berjalan dari posisi p ke posisi tetangga q dalam permukaan topografi ditentukan dengan persamaan sebagai berikut [1][2][3].

$$Cost(p,q) = \begin{cases} LS(p) \rightarrow f(p) > f(q) \\ LS(q) \rightarrow f(q) > f(p) \\ \dfrac{LS(p) \rightarrow LS(q) \rightarrow f(p) = f(q)}{2} \end{cases} \qquad (3.3)$$

3.1.4 Definisi *path* (jalur) antar pixel

Sebuah *path* γ dengan antara dua pixel *p* dan *q* dalam D_f adalah tupel dari pixel-pixel *(p0,p1,p2,....,pt-1,pt)* dimana *p0=p, pt= q* . Jarak topografi (topografic distance) antara p dan q di D_f sepanjang jalur (*path*) γ didefinisikan oleh [2]

$$TD^{\gamma}(p,q) = \sum_{i=1}^{S} \cos t(p_{i-1}, p_i) \tag{3.4}$$

3.1.5 Jarak topografi *(topografic distance)*

Jarak topografi antara dua pixel p dan q adalah jarak topografi minimal antara pixel-pixel ini terhadap keseluruhan kemungkinan jalur yang menghubungkan pixel *p* dan *q* di D_f [2]. Dalam beberapa paper, jarak topografi seringkali juga dinyatakan sebagai jarak geodesi (*geodesic distance*) [5],[7], definisi terhadap hal ini dinyatakan sebagai berikut.

$$TD(p,q) = \min_{\gamma}\{TD^{\gamma}(p,q)\} \tag{3.5}$$

Ilustrasi terhadap definisi jarak topografi atau jarak geodesi antara titik x dan y dalam citra A, ditunjukkan dalam gambar 3.2 berikut.

gambar 3.2. JarakTopografi

3.1.6 Downstream (arus menurun)

Pixel q dinyatakan sebagai *downstream* [2][5] dari pixel p bila ada sebuah jalur $\gamma = \{p0, p1, p2, ..., ps = q\}$ antara pixel p dan q dimana $\forall i \in \{1,2,...,s\}$, $p_{i-1} \in \Gamma(pi)$ [2] . Jika dimisalkan p dan q merupakan dua buah pixel dimana $f(p) > f(q)$ didapatkan definisi jarak topografi untuk *downstream* p yaitu q sebagai berikut :

$$TD(p,q) = f(p) - f(q) \Leftrightarrow q \in \text{downstream dari } p. \quad (3.6)$$

3.1.7 Definisi Local Minima

Sebuah *Local minima* dalam relief topografi digambarkan sebagai dasar dari suatu lembah. Dalam transformasi watershed , sebuah elemen $m(i ,j)$ dinyatakan sebagai *local minima* dengan definisi sebagai berikut [4]

$$m(i,j) < p(i_o,j_o) , \forall (i_o,j_o) \in N(i,j) \quad (3.7)$$

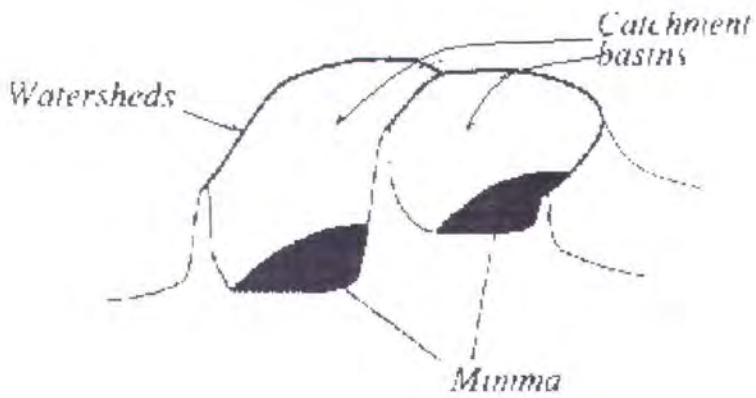
dimana N merepresentasikan tetangga dari elemen tersebut dalam keterhubungan 8.

Local minima ini merepresentasikan *catchment basin* dari image dan masing masing diberi label unik dan lebih besar dari nol [4],[8].

Jarak topografi antara pixel p dengan local minima m_i dimana p merupakan anggota *catchment basin* m_i dapat disamakan dengan $f(p) - f^*(mi)$, dimana $f^*(mi)$ *merupakan* intensitas graylevel dari local minima m_i . Lebih jauh m_i merupakan *downstream* dari keseluruhan p [2].

3.1.8 Catchment Basin.

Sebuah *catchment basin* dalam relief topografi dapat diibaratkan sebagai cekungan-cekungan yang terisi air. Untuk sebuah citra graylevel *I*, *catchment basin* yang berasosiasi dengan *local minima m* adalah himpunan dari pixel-pixel *p* dengan domain D_f dimana apabila terdapat air yang jatuh kepadanya akan mengalir mengikuti *downstream* p hingga akhirnya akan mencapai *m*. Ilustrasi untuk terhadap gambaran diatas terdapat dalam gambar 3.3.



Gambar 3.3. Watershed , catchment basin dan minima.

Dalam transformasi watershed, *Catchment Basin* dari local minima *mi* terdiri dari himpunan titik-titik $p \in D_f$ yang paling dekat dengan *mi* dibandingkan dengan local minima lainnya dan mengacu kepada definisi jarak topografi .dan didefinisikan sebagai berikut :

$$CB_f(m_i) = \{ p \in D_f \mid \forall j \in I \setminus \{i\} : f^*(mi) + TD(p,m_i) < f^*(mj) + TD(p, m_i)\} \quad (3.8)$$

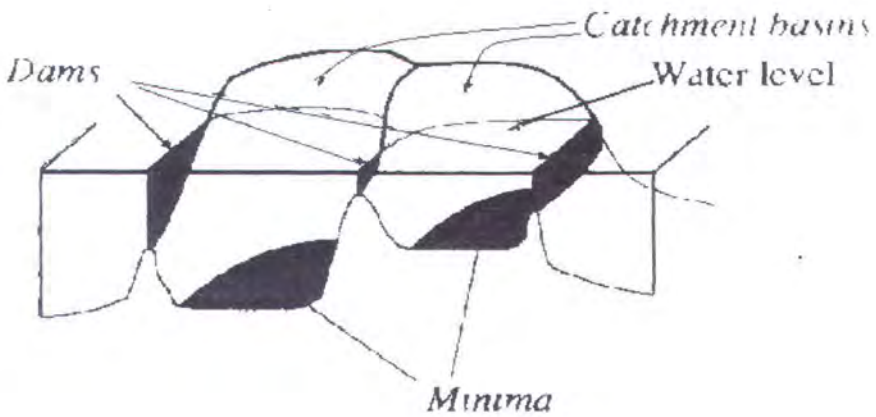
3.1.9 Watershed.

Dalam relief topografi, Watershed merupakan batas yang memisahkan danau-danau yang dalam hal ini merupakan *catchment basin*. Batas itu didapat apabila dua buah air dari dua buah *catchment* basin saling bertemu. Di titik dimana kedua air dari dua *catchment basin* yang berbeda bertemu disitulah kita membangun dam pembatas yang dalam hal ini disebut watershed .

Sebagai sebuah fungsi, watershed merupakan himpunan dari titik titik dari domain yang dinyatakan diatas, $p \in D_f$ dimana titik titik tersebut tidak tergabug kedalam catchment basin catchtment basin manapun [1].

$$Wsh(f) = D_f \cap (\cup_{i \in I} CB_f(m_i)))^c \qquad (3.9)$$

Gambar 3.4 berikut memberikan ilustrasi pembentukan *dam* di lokasi dimana dua air dari dua catchment basin yang berbeda bertemu.



Gambar 3.4. Pembangunan dam (wartershed) di lokasi dimana kedua air dari dua *catchment basin* bertemu.

Komputasi dari watershed diawali dari permasalahan perhitungan jalur yang mempunyai cost paling minimum antara pixel yang non *local minima* dan *local minima.* Jarak topografi antara antara 2 pixel p dan q , $f(q) < f(p)$ adalah minimum bila keduanya dihubungkan dengan *downstream* minimum [2], yakni $\gamma = \{p0, p1, p2, ..., ps = q\}$ dimana $\forall i \in \{1,2,3,\ldots,s\}, p_{i-1} \in \Gamma(pi)$. Mengacu pada persamaan 3.6, definisi *Cost Lower Slope* dan definisi *Jalur* (*path*) antar pixel. $\forall i \in \{1,2,3,\ldots,s\}$, $f(p_i) - f(p_{i-1}) = TD(p_{i-1}, pi) = cost\ (p_{i-1}, pi) = LS\ (pi)$, lebih lanjut dengan mengacu pada relasi dalam definisi Lower Slope menghasilkan $\forall p' \in N_G(p_i)$, $f(p_i) - f(p') \leq LS(p_i) = f(p_i) - f((p_{i-1}) \Rightarrow f((p_{i-1}) \leq f(p')\ \forall p' \in N_G(p)$.

Pada akhirnya dapat diambil kesimpulan , jika dimisalkan p adalah local minima, semua non local minima $p_i \in \gamma$ mendapatkan label dari tetangganya yang memiliki nilai paling rendah yaitu *Lower Slope* dari pixel tersebut.

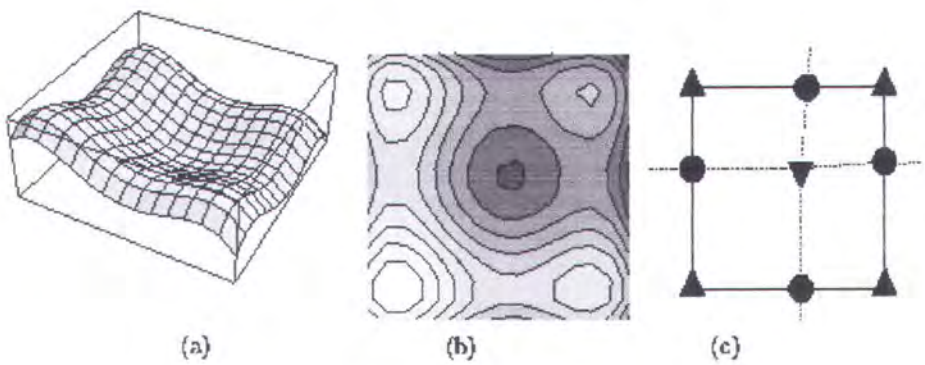
3.2 Segmentasi Citra dengan Transformasi Watershed

3.2.1 Definisi Umum

Segmentasi citra adalah sebuah proses yang membagi sebuah citra kedalam region-region yang terpisah, dimana setiap region adalah homogen dan mengacu kepada sebuah kriteria keseragaman yang jelas [2]. Secara lebih lengkap tujuan dari segmentasi citra ialah untuk memisahkan komponen dari suatu citra kedalam bagian-bagian yang berhubungan dengan obyek fisik dari gambar tersebut. Komponen

komponen yang telah disegmentasi tersebut kemudian digunakan oleh proses yang lebih lanjut untuk interprestasi terhadap citra ataupun pengenalan pola.

Transformasi Watershed merupakan metode yang efektif dan handal untuk melakukan segmentasi citra. Dalam metode ini , terdapat bagian yang memiliki kemiripan dengan region growing dalam hal pelabelan *connected components* [11],[12] (dalam transformasi watershed disebut *catchment basins*) dalam citra. Pendekatan mendasar yang digunakan dalam mendeskripsikan konstruksi dari watershed adalah dengan mengidentikkan citra dengan permukaan topografi. Ilustrasi dalam gambar 3.4 berikut menggambarkan konstruksi dari watershed .



Gambar 3.5 (a) topologi citra (b) citra grayscale (garis-garis ditambahkan untuk memperjelas beda intensitas) (c). maxima (daerah tertinggi),minima ,dan sadle (punggung bukit) yang digambarkan dengan segitiga keatas, lingkaran,dan segitiga kebawah.

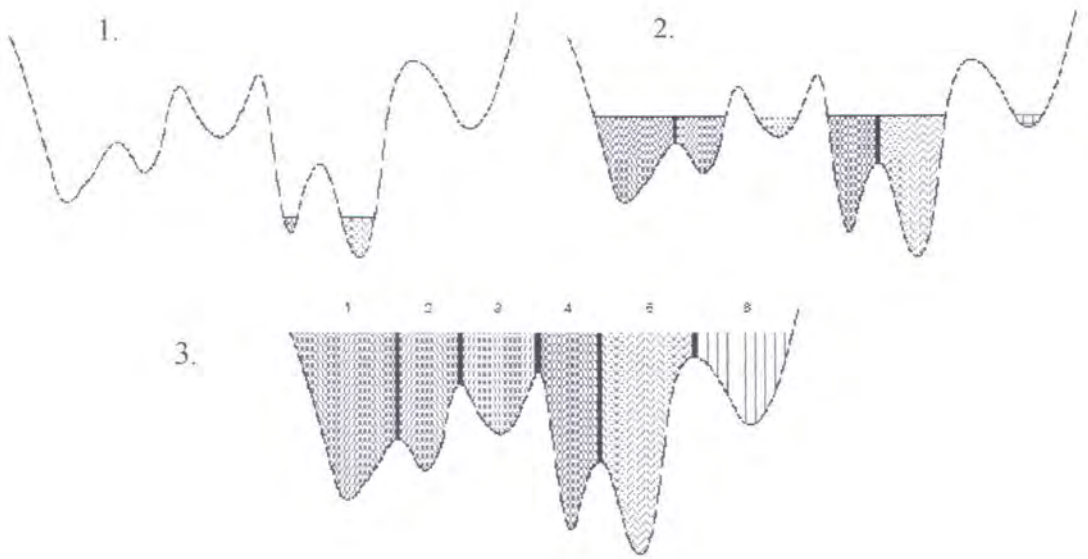
3.2.2 Metode Yang Ada di Transformasi Watershed.

Ada dua metode utama yang terdapat dalam transformasi watershed terhadap citra yang merupakan representasi ulang dari simulasi *watershed* dalam konsep topografi. Kedua metode tersebut mengacu kepada konsep *flooding* dalam topografi yang menggambarkan perilaku aliran air dalam suatu relief topografi [5]. Metode pertama disebut Watershed dengan simulasi perendaman (*immersion simulation*), metode kedua adalah dengan simulasi air hujan (*rainfalling simulation*).

3.2.2.1 Metode Transformasi Watershed dengan *Immersion Simulation.*

Pendekatan pertama dikenal sebagai *immersion* simulation (simulasi perendaman), .dimana pembentukan regionnya dengan metode *bottom up* [2],[5] . Dimisalkan kita merendami lembah-lembah dengan air dimulai dari dasar masing masing lembah yang disebut *minima.* Setelah ketinggian air di masing masing *catchment basin* atau lembah yang terendami dengan air mencapai ketinggian maksimal untuk masing masing *catchment basin*, maka air yang berasal dari catchment basin-catcment basin yang berdekatan akan menggabung. Ketika kedua air dari dua lembah yang berbeda akan menggabung, diposisi dimana kedua air dari dua buah catchment basin yang berbeda bertemu disitulah kita membangun dam/batas yang merepresentasikan watershed dari gambar. Setelah air telah mencapai ketinggian maksimal dari permukaan topografi maka keseluruhan watershed telah kita dapatkan.

Ilustrasi untuk konsep ini secara jelas ditunjukkan untuk mengapresiasikan secara lebih baik efek dari transformasi yang dilakukan terhadap citra.yang ditunjukkan oleh gambar 3.6 berikut.



Gambar 3.6 . Watershed dengan simulasi perendaman

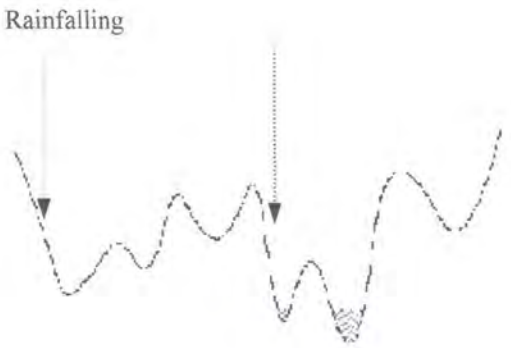
Dalam gambar 1, perendaman dimulai di minima yang paling rendah pada ketinggian h = 0. Selanjutnya secara bertahap air dinaikkan hingga mencapai ketinggian maksimal (h=maks) dari permukaan topografi citra.

3.2.2.2 Metode Transformasi Watershed dengan Rainfalling Simulation

Pendekatan kedua dikenal sebagai *rainfalling simulation*[2]. Metode ini juga disebut dengan *downhill watershed* [8] atau "*minimum following algorithm*" [9]. Penggambaran simulasi ini ini dalam topografi adalah sebagai berikut. Apabila air

hujan jatuh pada suatu permukaan topografi; sesuai dengan hukum gravitasi maka air tersebut akan mengalir melewati jalur yang lebih rendah sampai ia mencapai ketinggian yang paling rendah atau minima dimana ia tidak dapat mengalir kemana-mana lagi.

Ilustrasi terhadap konsep ini digambarkan dalam gambar 3.7 berikut



Gambar 3.7 Watershed dengan *rainfalling simulation*

Himpunan titik-titik pada permukaan topografi dimana aliran air yang melewatinya menuju ke *minima* tertentu yang sama menjadi sebuah *catchment basin atau connected component* yang berasosiasi dengan *minima* tersebut. Watershed adalah batas yang membagi dua buah catchment basin yang berdekatan.

Dalam pengolahan citra, transformasi watershed yang mengacu kepada konsep *rainfalling simulation* ini dimulai dengan melakukan adalah pencarian *local minima* yang merupakan dasar (*bottom*) dari sebuah *catchment basin. Local minima* ini selanjutnya diberi nomor yang *unik* yang merepresentasikan nomor dari *region*

yang akan dihasilkan [2],[4],[8]. Dalam proses ini sekaligus akan didapatkan *lower slope (gradien)* dari setiap pixel dalam citra yang bukan merupakan *local minima.* Langkah ini dilakukan dengan mencari *lower slope* dari setiap pixel dalam citra dengan mengacu pada persamaan 3.2. Pencarian *local minima* dan *gradien* dari setiap pixel dilakukan dengan mengacu pada konsep tetangga dalam keterhubungan (*connectivity*). Langkah ini dilanjutnya dengan mengikuti setiap gradien dari pixel citra ini hingga bertemu *local minima.* Ketika penelusuran dengan mengikuti arah *gradien* dari pixel tersebut telah menemui suatu *local minima* , maka pixel tersebut diberi label sesuai dengan label dari *local minima.* Untuk menghemat waktu semua pixel yang dilalui hingga mencapai local minima juga dilabeli dengan nomor yang sama. Proses pemberian label ini disebut dengan *connected component labelling* [11] yang akan menghasilkan *region-region* dari citra. Proses akhir dari metode ini adalah dengan mencari watershed yang didapat dengan melakukan proses terhadap *connected component* dari citra dimulai dari kiri ke kanan dan selanjutjnya dari atas ke bawah dengan mendeteksi setiap perubahan dari nomor region yang merepresentaskan tepi dari suatu *region* [4],[8].

3.2.3 Metode Segmentasi yang dipilih

Dalam tugas akhir ini metode yang penulis gunakan menggunakan metode *rainfalling watershed* dengan pertimbangan bahwa pendekatan *imersion watershed* membutuhkan waktu komputasi yang lebih lama karena sebelum proses perendaman terhadap permukaan topografi citra dilakukan, haruslah ditentukan dulu urutan-urutan

pixel-pixel dimana perendaman akan dimulai [5]. Urutan-urutan tersebut ditentukan dengan melakukan sorting terhadap ketinggian topografi setiap pixel dalam citra tersebut yang dalam hal ini ditentukan oleh intensitas gray levelnya.

Dengan metode pertama yang identik dengan proses *bottom up*, *pixel* yang paling rendah akan direndami terlebih dahulu. Proses sorting dilakukan kerhadap keseluruhan pixel-pixel dalam citra utnuk mendapatkan urutan perendaman menghabiskan waktu komputasi yang sebanding dengan luas dari citra. Sehingga banyak data yang sorting apabila ukuran citra adalah 512 x 512 adalah sebanyak 262144 data.

Proses sorting ini hanya akan menghasilkan urutan pixel berdasarkan ketinggianya, yang dalam hal ini adalah intensitas gray level masing masing pixel tersebut sehingga kemungkinan adanya *minima minima*_palsu jauh lebih besar dibandingkan *rainfalling* watershed. Hal ini memungkinkan adanya *over segmented* yang juga jauh lebih besar.

Diperlukan suatu struktur data tambahan dalam metode pertama yang berupa sebuah *queue* untuk mensimulasikan proses perendaman terhadap citra [5]. *Queue* yang digunakan ini mengacu kepada proses *first in first out* (FIFO). Pixel pixel yang masuk pertama kali sebagai anggota *queue* inilah yang pertama kali direndami[5].

Sebaliknya dalam pendekatan rainfalling watershed tidak diperlukan adanya sorting terhadap citra yang diproses. Proses *rainfalling* watershed yang mengacu pada metode *top down* dapat langsung dilakukan kepada masing masing pixel. Proses ini dilakukan dengan mengikuti *gradien* (*lower slope*) dari masing masing pixel tersebut

dimana akan didapatkan jalur terpendek untuk menuju minima terdekat. Proses awal yang dilakukan bukanlah melakukan sorting terhadap data, akan tetapi menentukan semua *local minima.* Dalam menentukan *local minima* ini, sekaligus akan didapatkan *lower slope* dari setiap pixel yang bukan merupakan *local minima.*

3.2.4 Transformasi Watershed terhadap Gradien Magnitude Citra

Transformasi Watershed secara langsung terhadap citra kurang memberikan hasil yang bagus . Kontur yang diharapkan dapat diekstraksi dengan mengaplikasikan transformasi watershed terhadap *gradient magnitude* citra [9],[10], suatu *diferensiasi non linear* terhadap citra (penjelasan mengenai gradien magnitude telah dijelaskan dalam bab II). .Pendekatan ini merupakan metode kunci untuk mendapatkan segmentasi yang bagus berbasis transformasi watershed. Dalam metode ini, *local minima* didapatkan dari perhitungan terhadap *gradient magnitude* citra. Untuk menghindari efek dari *noise* yang terdapat pada citra asli, dilakukan *preprosesing* yang bertujuan untuk menghilangkan noise yang terdapat pada citra sebelum *mencari gradien magnitude* dari citra.

Pemikiran yang mendasari metode ini adalah bahwa perubahan- perubahan yang terjadi pada setiap intensitas pixel dalam citra berkorespondensi secara langsung terhadap perubahan yang terjadi pada topografi citra. Tepi-tepi yang yang didapatkan dari proses *gradient magnitude* merepresentasikan "*mountain range*" [10] (pegunungan yang membatasi lembah) dan dalam proses transformasi dideteksi sebagai *watersheds* yang berkorespondensi dengan batas-batas dari obyek.

3.2.5 Tahapan-Tahapan Segmentasi Citra dengan Transformasi Watershed

Tahapan tahapan dalam segmentasi citra dengan transformasi watershed dibagi kedalam 5 tahap utama

1. Penghalusan Citra (*Image Smoothing*)

Penghalusan citra digunakan untuk menghilangkan *noise* yang ada pada citra. Terdapat beberapa metode yang dapat digunakan , diantaranya adalah dengan *filter average* ataupun dengan filter morphologi yaitu *open-close filter* . Efek dari penghalusan terhadap proses segmentasi sangat *significant* . Kemungkinan adanya *over segmented* sebagai akibat munculnya *region-region* palsu dapat diminimalkan.

2. Pengaburan citra dengan Kernel Gaussian (Gaussian Blurring)

Gambar dikaburkan dengan menggunakan *kernel gaussian* dua dimensi. Proses pengaburan (*blur*) ini dilakukan karena nilai *integer* (bulat) presisinya kurang baik untuk digunakan dalam mendeteksi *local minima* atau *lower slope* dari setiap pixel dalam citra [8]. Dengan proses *gaussian blurring* nilai -nilai integer pixel dikonversikan kedalam nilai-nilai pecahan (*floating point*) yang akan meningkatkan ketelitian dalam menghitung *local minima* ataupun *lower slope.* Konversi ini juga mengurangi efek *plateau* [8](efek dataran datar yang sangat luas sehingga arah aliran air sulit diprediksi). Selain itu , proses *blurring* dengan *kernel gaussian* juga mampu menghilangkan noise yang ada pada citra.

Proses pengaburan gambar ini dilakukan dengan operasi konvolusi menggunakan mask dua dimensi yang diperoleh dengan mendiskritkan fungsi

kontinyu *Gaussian* dua dimensi yang menghasilkan *kernel Gaussian*. Akan tetapi konvolusi dengan *kernel Gaussian* dua dimensi ini membutuhkan waktu komputasi yang lama sehingga digunakan dua buah *kernel Gaussian* satu dimensi kearah *vertical* dan ke arah *horizontal*. Penjelasan mengenai hal ini telah dijelaskan dalam bab II (Teori Penunjang).

3. Pencarian Gradien Magnitude citra

Dikarenakan transformasi watershed yang digunakan adalah transformasi watershed terhadap gradien Magnitude citra, maka dilakukan perhitungan untuk mencari gradien magnitude citra. Penjelasan mengenai perhitungan *gradien magnitude* citra telah dijelaskan dalam bab II (Teori Penunjang).

Dalam prakteknya pencarian gradien magnitude citra ini dapat secara mudah dan cepat dilakukan. Dimisalkan $\forall I$ merupakan gradien magnitude dari citra. Nilai $\forall I(i,j)$ didapatkan dengan perhitungan sebagai berikut [4]:

$$\forall I(i,j) = [I(i,j+1)-I(i,j)]^2 + [I(i,j) - I(i+1,j)]^{1/2}$$

4. Transformasi Watershed

Langkah pertama dalam transformasi watershed yang mengacu kepada metode rainfalling watershed adalah dengan mengidentifikasi *local minima* dari citra. Karena nilai intensitas citra telah dikonversikan dalam nilai *floating point* oleh proses *blurring* maka hasil local minima yang didapat lebih baik. Proses ini dilakukan

dengan membandingkan nilai intensitas pixel dengan nilai intensitas pixel tetangganya dalam keterhubungan 8. Apabila nilai intensitas pixel lebih kecil dari keseluruhan nilai tetangganya maka dia didefinisikan sebagai *local minima* (pers 3.8).

$$m(i,j) < p(i_o,j_o) , \forall(i_o,j_o) \in N(i,j).$$

Local minima ini diberi label dengan nilai integer positif yang merepresentasikan nomor *catchment basin* yang merepresentasikan nomor region.

Langkah selanjutnya adalah menghitung *lower slope/gradien* (arah aliran air) dari setiap pixel dalam citra. Proses ini dapat dilakukan sekaligus dengan proses pencarian local *minima*. Karena dalam proses pembandingan terhadap tetangga pixel untuk mencari *local minima* akan didapatkan *local minima* apabila nilai pixel pusat lebih kecil dari tetangganya. Akan tetapi apabila ada pixel tetangga yang lebih kecil intensitasnya dibandingkan dengan pixel pusat , maka pixel pusat bukanlah *local minima* dan nilai pixel tetangganya tersebut dijadikan nilai *lower slope (gradien).*

Proses selanjutnya adalah mengikuti setiap *gradien* dari pixel citra ini hingga bertemu *local minima*. Ketika penelusuran dengan mengikuti arah *gradien* dari pixel tersebut telah menemui suatu *local minima* , maka pixel tersebut diberi label sesuai dengan label dari *local minima*. Untuk menghemat waktu semua pixel yang dilalui hingga mencapai *local minima* juga dilabeli dengan nilai yang sama dengan label pada local minima yang menunjjukan bahwa pixel tersebut juga tergabung kedalam *catchment basins* yang sama.. Proses pemberian label ini disebut dengan *connected component labelling* [12] yang akan menghasilkan *region-region* dari citra. Proses

akhir dari metode ini adalah dengan mencari watershed yang didapat dengan melakukan proses terhadap *connected component* dari citra dimulai dari kiri ke kanan dan selanjutjnya dari atas ke bawah dengan mendeteksi setiap perubahan dari nomor region yang merepresentaskan tepi dari suatu *region* [4],[8].

3.2.6 Penggabungan Region

Teknik penggabungan region perlu ditambahkan karena walaupun tahapan tahapan diatas telah dilalui dengan baik , masih dimungkinkan terdapat region region palsu (*over segmented*) [4]. Terdapat beberapa teknik untuk melakukan penggabungan region, misalnya merging dengan dasar *mean* dari setiap region ataupun dengan *variance* dari setiap region.

Untuk itu dalam prakteknya informasi dari setiap region perlu disimpan sebagai acuan dalam proses penggabungan region. Informasi ini antara lain adalah luas *region* dan *mean* dari setiap region yang didapatkan dari perhitungan total intensitas dari setiap pixel yang menjadi anggota region tersebut dibagi dengan luas region.

Selanjutnya dilakukan pengecekan untuk setiap region yang berdekatan (*adjacent region*) apakah selisih nilai meannya masing masing berada dibawah nilai treshold untuk proses merging. Apabila nilai selisihnya berada dibawah treshold, maka kedua region digabung dan dilakukan penghitungan baru untuk mendapatkan nilai mean dan luas region baru yang terbentuk.

Apabila sebuah region A telah digabung menjadi bagian dari sebuah region B, maka region B merupakan *parent* dari region A. Untuk selanjutnya seluruh elemen yang menjadi anggota region A akan diberi label yang sama dengan elemen anggota region B yang menunjukkan bahwa region A telah bergabung menjadi bagian dari region B. Luas dari region B berubah menjadi luas region gabungan yang didapatkan dengan menambahkan luas region A dengan luas region B. Nilai mean dari region baru yang terbentuk didapatkan dengan menghitung total intensitas baru dari region tersebut dan dibagi dengan luas region baru yang terbentuk.

BAB IV

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

BAB IV

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Bab ini akan membahas perancangan dan pembuatan perangkat lunak yang telah dibuat dalam tugas akhir secara garis besar. Perancangan dan pembuatan perangkat lunak dibuat untuk mengimplementasikan teori-teori yang telah diuraikan dalam bab-bab sebelumnya.

Pembahasan tentang perancangan perangkat lunak meliputi perancangan data, perancangan proses serta diagram aliran data. Sedangkan pembuatan perangkat lunak meliputi implementasi program yang meliputi implementasi data dan implementasi prosesnya.

Perangkat lunak tersebut dibuat dengan menggunakan bahasa pemrograman *Borland Delphi 3.0* dan berjalan dalam lingkungan sistem operasi *Windows98*. Digunakannya bahas pemrograman tersebut dikarenakan *Borland Delphi* merupakan bahasa pemrograman visual yang didukung oleh banyak antarmuka serta fasilitas pengaksesan data citra yang lengkap, sehingga dapat lebih mempercepat dalam pembuatan aplikasi. Hal tersebut juga didukung kemampuan *Delphi* yang mendukung pemrograman berorientasi object (OOP).

Digunakannya sistem operasi *Windows* adalah dengan pertimbangan fasilitas dan kemudahannya dalam mengakses data-data yang berupa citra dan grafik, kemampuan melakukan *multitasking* serta kemampuannya untuk memanajemen memori secara baik.

4.1 Perancangan Perangkat Lunak

Sub Bab ini membahas tentang perancangan perangkat lunak sebelum diimplementasikan kedalam program. Perancangan perangkat lunak ini meliputi perancangan data, perancangan proses serta diagram aliran data.

4.1.1 Perancangan Data

Pada setiap pembuatan program akan selalu ada dua hal yang menjadi inti utamanya, yaitu proses dan data. Setiap proses yang ada selalu diawali dengan memproses suatu data masukan dan akan menghasilkan data keluaran. Dalam perancangan data dalam tugas akhir ini secara garis besar dibagi kedalam tiga jenis, yaitu data masukan, data proses dan data keluaran.

4.1.1.1 Data Masukan

Data masukan (input) yang digunakan dalam perangkat lunak ini berupa file citra *graylevel* dengan level 256 dalam format *bitmap*, karena proses transformasi yang dilakukan adalah terhadap citra *graylevel.*

4.1.1.2 Data Proses

Data proses adalah data-data yang digunakan selama proses dijalankan. Pada saat perangkat lunak dijalankan dan melakukan interaksi masukan , data-data yang digunakan dan dihasilkan adalah sebagai berikut. :

1. DataImage, adalah data dari gambar yang akan diproses. Data ini merupakan array dinamis satu dimensi dengan elemen byte yang menyimpan nilai intensitas gray level dari gambar.
2. SmoothData, adalah data dari proses reduksi *noise* terhadap DataImage. Data ini merupakan array dinamis satu dimensi dengan elemen *real*.
3. GaussData , adalah SmoothData yang telah dilakukan proses *blurring* dengan gaussian filter.dua dimensi. Data ini merupakan arary dinamis satu dimensi yang menyimpan nilai-nilai floating point hasil proses blurring terhadap DataImage.
4. MagValue , adalah data dari proses Gradien Magnitude terhadap GaussData.. Data ini merupakan array array dinamis satu dimensi yang menyimpan nilai-nilai gradien magnitude dari GaussData. Isi array ini juga bertipe floating point agar data yang didapat lebih teliti.
5. WaterValue adalah data yang menyimpan hasil proses Watershed yang menyimpan informasi nomor region dari setiap pixel dalam image. Array ini merupakan array dinamis satu dimensi dengan elemen bertipe integer.
6. Region merupakan array yang berisi informasi dari setiap region yang didapat. Data ini merupakan array dinamis satu dimensi dengan elemen array merupakan record RegionRec yang menampung informasi dari setiap region yang terbentuk. Array ini di buat setelah informasi mengenai jumlah seluruh region yang terbentuk didapatkan dari proses transformasi watershed.
7. Adj_Info menyimpan seluruh informasi tentang dua buah region berdekatan yang akan digabung dan digunakan dalam proses *merging region.* Data ini

merupakan array dinamis satu dimensi dengan elemen array merupakan record InfoRec yang menampung informasi dua region yang akan digabung.

4.1.1.3 Data Keluaran

Perangkat lunak yang dibuat ini bertujuan untuk melakukan segmentasi terhadap citra yang mendeteksi adanya kontur-kontur tertutup dari sebuah citra. Jadi output yang dihasilkan dari proses ini adalah berupa citra kontur yang didapatkan dari proses transformasi watershed terhadap citra masukan.

4.1.2 Perancangan Proses

Perancangan proses dalam perangkat lunak ini dibagi dalam enam tahap.Tahapan-tahapan tersebut terdiri dari tahapan sebelum transformasi watershed dijalankan. yaitu proses reduksi noise, proses blur dengan kernel gaussian dan pencarian gradien magnitude citra. Proses transformasi kemudian dijalankan terhadap *gradien magnitude* citra dan dilanjutkan dengan proses *merging region*. Proses terakhir adalah menentukan *kontur-kontur* tertutup dari region-region yang didapatkan yang merupakan hasil akhir dari keseluruhan proses..Tahapan-tahapan proses tersebut adalah sebagai berikut :

1. Reduksi *Noise*

Proses Reduksi *Noise* digunakan untuk mengurangi/menghilangkan *noise* yang ada pada citra. Terdapat beberapa metode yang dapat digunakan, diantaranya adalah dengan *filter average* ataupun dengan filter morphologi yaitu *open-close filter*. Efek dari reduksi noise terhadap proses segmentasi sangat *significant*.

Kemungkinan adanya *over segmented* sebagai akibat munculnya *region-region* palsu dapat diminimalkan.

2. Pengaburan citra dengan *Kernel Gaussian* (*Gaussian Blurring*)

Gambar dikaburkan dengan menggunakan fungsi Gaussian dua dimensi. Proses pengaburan (*blur*) ini dilakukan karena nilai *integer* (bulat) presisinya kurang baik untuk digunakan dalam mendeteksi *local minima* atau *lower slope* dari setiap pixel dalam citra [8]. Dengan proses *gaussian blurring* nilai -nilai integer pixel dikonversikan kedalam nilai-nilai pecahan (*floating point*) yang akan meningkatkan ketelitian dalam menghitung *local minima* ataupun *lower slope.* Konversi ini juga mengurangi efek *plateau* [8] (efek dataran datar yang sangat luas sehingga arah aliran air sulit diprediksi). Selain itu , proses *blurring* dengan *kernel gaussian* juga mampu menghilangkan noise yang ada pada citra.

Proses pengaburan gambar ini dilakukan dengan operasi konvolusi menggunakan mask dua dimensi yang diperoleh dengan mendiskritkan fungsi kontinyu *Gaussian* dua dimensi yang menghasilkan *kernel Gaussian.* Akan tetapi konvolusi dengan *kernel Gaussian* dua dimensi ini membutuhkan waktu komputasi yang lama sehingga digunakan dua buah *kernel Gaussian* satu dimensi kearah *vertical* dan ke arah *horizontal.*

Penjelasan mengenai hal ini secara lengkap telah dijelaskan dalam bab sebelumya yaitu Bab II (Teori Penunjang).

3. Pencarian Gradien Magnitude citra

Tahapan pra-Transformasi terakhir yang dilakukan ialah mencari gradien Magnitude citra. Penjelasan mengenai perhitungan *gradien magnitude* citra telah dijelaskan secara mendalam dalam bab II (Teori Penunjang).

Dalam prakteknya pencarian *gradien magnitude* citra ini dapat secara mudah dan cepat dilakukan. Dimisalkan $\forall I$ merupakan gradien magnitude dari citra. Nilai $\forall I$(i,j) didapatkan dengan perhitungan sebagai berikut [3]:

$$\forall I(i,j) = [I(i,j+1)-I(i,j)]^2 + [I(i,j) - I(i+1,j)]^{1/2}$$

Perhitungan diatas diberlakukan terhadap keseluruhan pixel-pixel dalam citra.

4. Transformasi Watershed

Langkah pertama dalam transformasi watershed yang mengacu kepada metode *rainfalling* watershed adalah dengan mengidentifikasi *local minima* dari citra. Karena nilai intensitas citra telah dikonversikan dalam nilai *floating point* oleh proses *blurring* maka hasil local minima yang didapat lebih baik. Proses ini dilakukan dengan membandingkan nilai intensitas pixel dengan nilai intensitas pixel tetangganya dalam keterhubungan 8. Apabila nilai intensitas pixel lebih kecil dari keseluruhan nilai tetangganya maka dia didefinisikan sebagai *local minima* (pers 3.8).

$$m(i,j) < p(i_o,j_o) , \forall (i_o,j_o) \in N(i,j).$$

Local minima ini diberi label dengan nilai integer positif yang merepresentasikan nomor *catchment basin* yang merepresentasikan nomor region.

Langkah selanjutnya adalah menghitung *lower slope/gradien* (arah aliran air) dari setiap pixel dalam citra. Proses ini dapat dilakukan sekaligus dengan proses pencarian local *minima*. Karena dalam proses pembandingan terhadap tetangga pixel untuk mencari *local minima* akan didapatkan *local minima* apabila nilai pixel pusat lebih kecil dari tetangganya. Akan tetapi apabila ada pixel tetangga yang lebih kecil intensitasnya dibandingkan dengan pixel pusat , maka pixel pusat bukanlah *local minima* dan nilai pixel tetangganya tersebut dijadikan nilai *lower slope (gradien).*

Proses selanjutnya adalah mengikuti setiap *gradien* dari pixel citra ini hingga bertemu *local minima*. Ketika penelusuran dengan mengikuti arah *gradien* dari pixel tersebut telah menemui suatu *local minima* , maka pixel tersebut diberi label sesuai dengan label dari *local minima*. Untuk menghemat waktu semua pixel yang dilalui hingga mencapai *local minima* juga dilabeli dengan nilai yang sama dengan label pada local minima yang menunjjukan bahwa pixel tersebut juga tergabung kedalam *catchment basins* yang sama.. Proses pemberian label ini disebut dengan *connected component labelling* [13] yang akan menghasilkan *region-region* dari citra.

5. Proses Penggabungan Region (*Region Merging*).

Proses Ini merupkanan proses tambahan yang dilakukan untuk mengantisipasi kemungkinan adanya *over segmented* (hasil segmentasi berlebihan). Apabila region-region yang didapatkan telah benar, maka proses ini tidak melakukan penggabungan region. Akan tetapi apabila ada dua region yang

berdekatan yang harusnya merupakan satu region maka kedua region tersebut digabung.

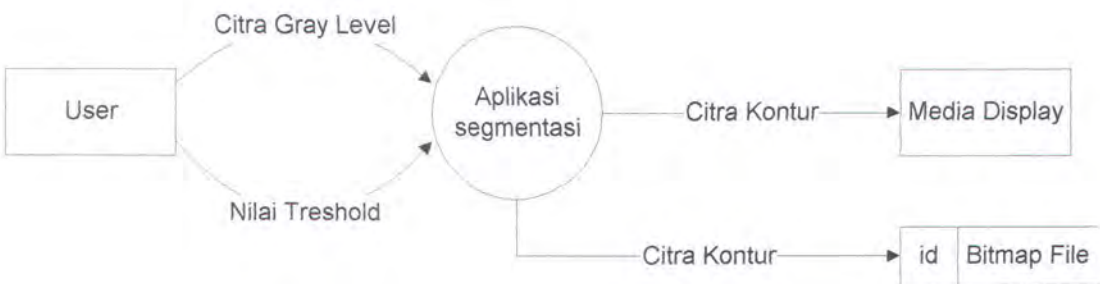
6. Penentuan tepi-tepi region (watershed)

Proses akhir dari metode ini adalah dengan mencari watershed yang didapat dengan melakukan proses *scanning* terhadap *connected component* dari citra dimulai dari kiri ke kanan dan selanjutnya dari atas ke bawah dengan mendeteksi setiap perubahan dari nomor region yang merepresentaskan tepi dari suatu *region* [3][8].

4.1.3 Diagram Alir Data (DAD)

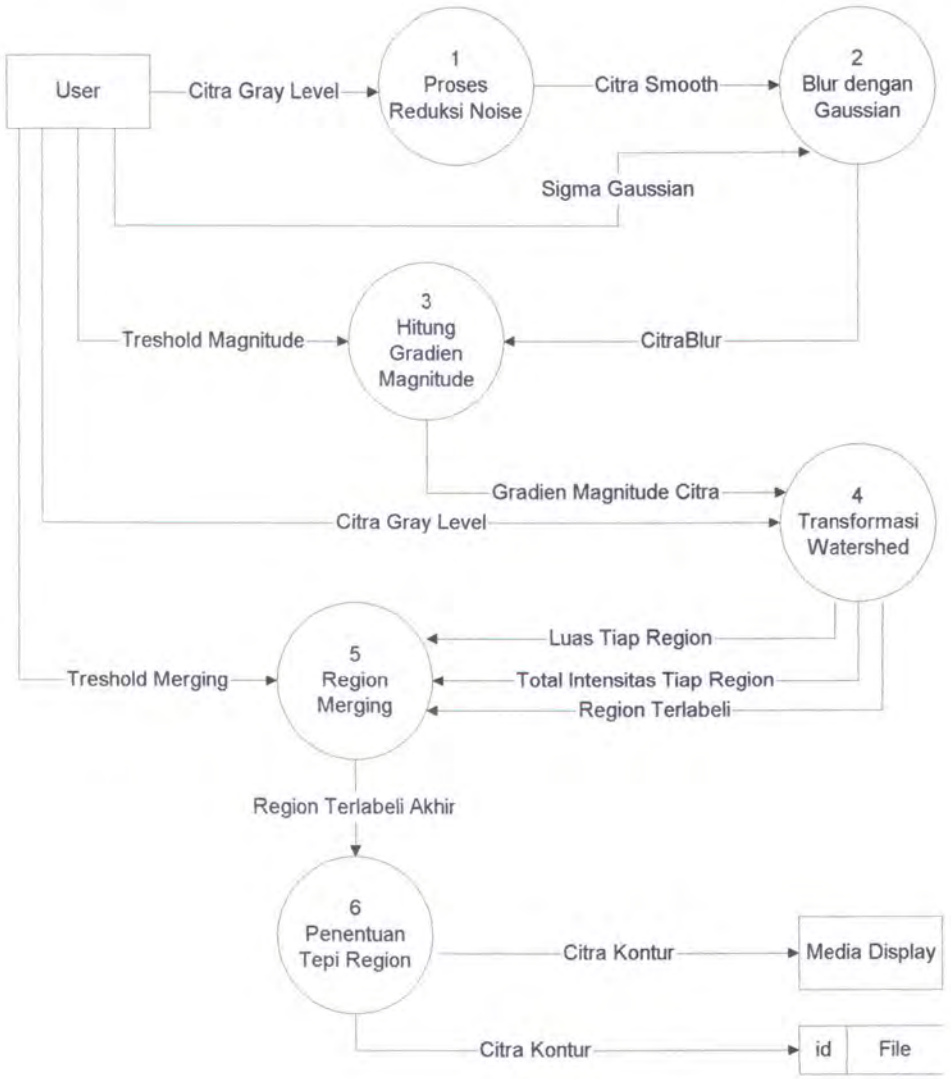
Diagram alir Data (DAD) merupakan salah satu cara untuk menggambarkan proses yang dapat memperjelas perancangan pembuatan suatu perangkat lunak. Dari DAD yang dibuat dapat diketahui data yang diolah dalam suatu proses dan data yang dihasilkan dalam proses tersebut. Dengan demikian aliran data mulai awal sampai akhir dapat diketahui dengan jelas.

Gambar berikut ini merupakan DAD level 0 dari proses Segmentasi Citra



Gambar 4.1 DAD level 0, Segmentasi Citra.Dengan Transformasi Watershed

Diagram Alir Data diatas terdiri dari enam proses yang secara detail digambarkan dalam gambar 4.2 yang merupakan Diagram Alir Data level 1



Gambar 4.2 Diagram 0 level 1, Segmentasi Citra dengan Transformasi Watershed.

Dalam DAD level 1 tersebut, data citra *graylevel* yang dimasukkan oleh user diolah dalam proses Reduksi *Noise* sehingga menghasilkan citra yang telah diperhalus (citra *smooth*).

Citra *Smooth* ini kemudian digunakan sebagai masukan dalam proses kedua yaitu *blur* dengan *Gaussian Kernel*. Dalam proses kedua ini terdapat masukan dari user yaitu nilai *sigma gaussian* yang digunakan untuk membuat *kernel gaussian*.Hasil yang didapatkan merupakan citra *blur* dimana dalam proses ini noise yang masih ada juga dereduksi serta didapatkan nilai-nilai floating point dari setiap intensitas pixel dalam citra.

Proses berikutnya adalah menghitung *gradien Magnitude* dari setiap pixel dalam citra dengan masukan citra *blur* yang didapatkan dari proses kedua. Dalam proses ketiga ini juga didapatkan masukan dari user berupa treshold *magnitude*. *Gradien magnitude* dari citra yang didapatkan kemudian dijadikan masukan dalam proses keempat yaitu transformasi watershed.

Dalam proses keempat data *gradien magnitude* citra digunakan untuk menghasilkan data region terlabeli. Dalam proses keempat ini juga dihasilkan informasi luas tiap region dan total intensitas tiap region yang didapat dengan mengkorelasikan data region terlabeli dan data citra graylevel asli yang akan digunakan dalam proses merging region.

Informasi Luas tiap region, intensitas total tiap region dan region terlabeli yang didapatkan dari proses transformasi watershed digunakan sebagai masukan dalam proses region merging. Dalam proses ini kriteria *merging* antara dua region yang berdekatan diuji dengan menggunakan nilai treshold merging yang

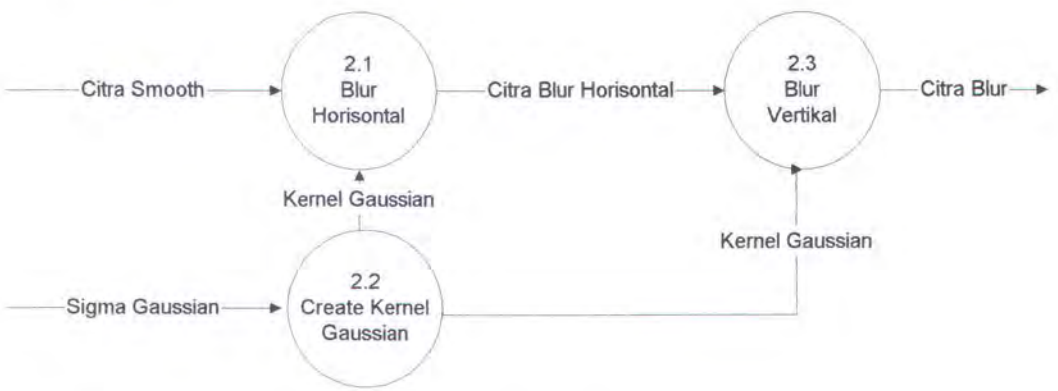
didapatkan dari user. Hasil yang didapatkan adalah region terlabeli akhir . Selanjutnya dalam proses keenam ditentukan tepi- tepi region dari region terlabeli akhir tersebut sehingga didapatkan hasil akhir berupa citra kontur.

DAD level 2 dari proses reduksi noise digambarkan dalam gambar 4.3

Gambar 4.3 Diagram 1 level 2, Proses Reduksi Noise

Dalam Diagram Aliran Data diatas, secara sederhana dapat digambarkan bahwa proses tersebut dilakukan dengan mengkonvolusi citra *graylevel* dengan filter *average* yang akan menghasilkan citra *smooth.*

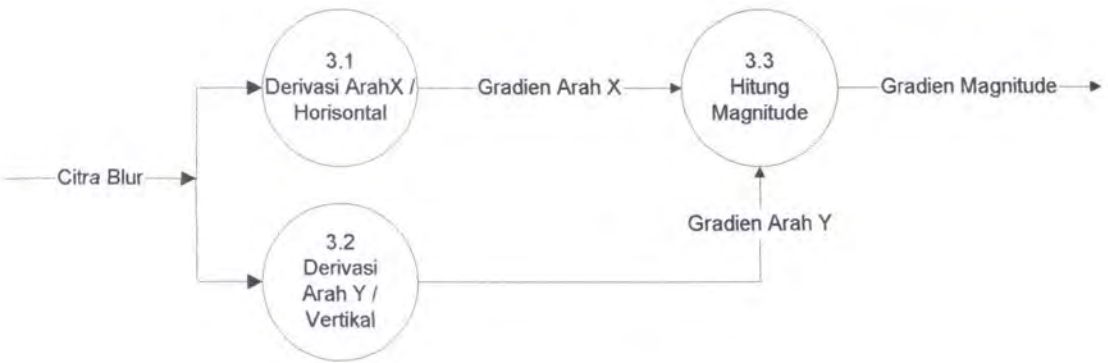
Diagram Aliran Data level 2 dari proses *blur* dengan Gaussian digambarkan dalam gambar 4.4 sebagai berikut.



Gambar 4.4 Diagram 2 level 2, Proses Blur dengan Gaussian

Dalam DAD diatas digambarkan proses konvolusi dengan *kernel Gaussian* yang akan menghasilkan citra *blur*. Input untuk *sigma gaussian* didapatkan dari input user. Nilai ini digunakan untuk membuat *kernel gaussian* yang digunakan untuk menghasilkan citra blur sebagai data masukan dalam proses perhitungan *gradien magnitude*.

DAD level 2 dari proses *gradien magnitude* digambarkan dalam gambar 4.5 sebagai berikut.

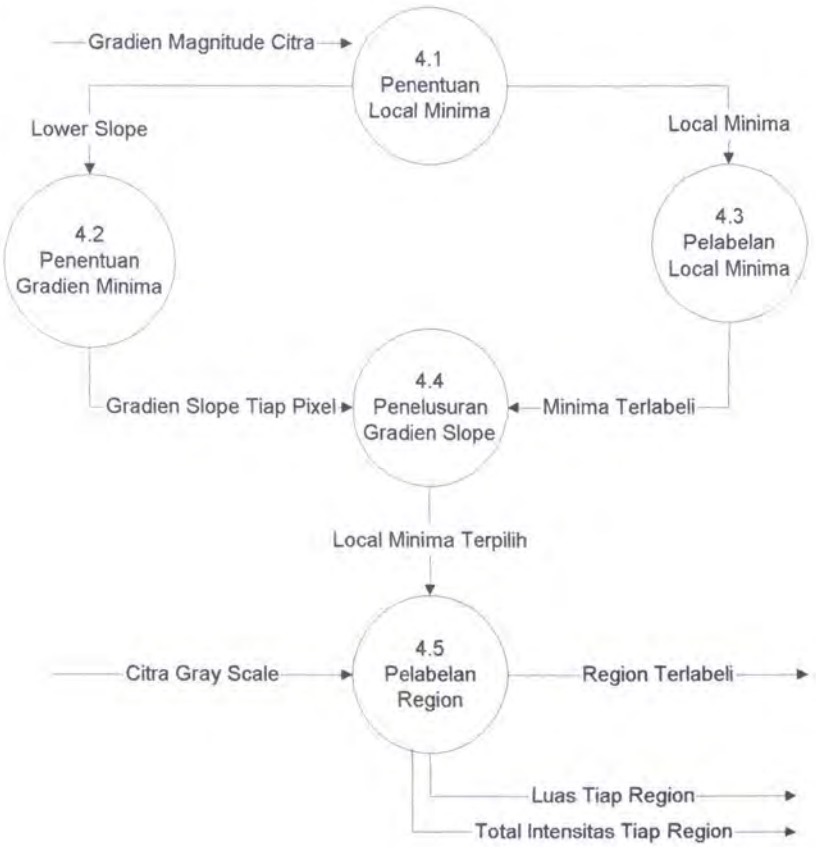


Gambar 4.5 Diagram 3 level 2 , Proses perhitungan Gradien Magnitude.

Proses perhitungan *gradien magnitude* citra digambarkan secara jelas dalam gambar diatas. Proses perhitungan tersebut dilakukan dengan melakukan derivasi horisontal terhadap citra *blur* dan derivasi vertikal terhadap citra *blur* . Hasilnya digunakan untuk menghasilkan gradien magnitude citra.

Sampai tahapan perhitungan gradien magnitude citra diatas, semua tahapan-tahapan pra-transformasi telah selesai dilaksanakan. Tahapan diatas menghasilkan nilai-nilai *gradien magnitude* citra *smooth* yang telah di *blur* dengan *kernel gaussian*.

Diagram Alir Data proses Transformasi Watershed digambarkan dalam gambar 4.6 sebagai berikut :



Gambar 4.6 Diagram 4 level 2 , Transformasi Watershed

Tahapan keempat yang DAD-nya digambarkan dalam gambar 4.6 yaitu proses transformasi Watershed.. Dalam proses transformasi tersebut langkah pertama adalah menentukan seluruh *local minima* dalam citra. Dalam proses ini dihasilkan dua data. Data yang pertama adalah pixel yang diidentifikasikan sebagai *local minima.* Data kedua adalah *lower slope* dari setiap pixel yang bukan merupakan *local minima.*

Tiap *local minima* yang didapat kemudian dalam proses pelabelan *local minima* diberi nomor yang *unik*. *Lower slope* yang didapatkan, dalam proses penentuan gradien slope ditentukan nilai arahnya (*gradien*). Sehingga didapatkan arah (*gradien*) *lower slope* yaitu lokasi pixel tetangga.yang diidentifikasikan sebagai *lower slope*.

Proses selanjutnya adalah proses penelusuran *gradien slope* yang digambarkan dalam diagram sebagai proses 4.4. Penelusuran ini dilakukan terhadap koefisien awal dari watershed yang berisi local minima dan nilai nilai gradien slope dari setiap pixel yang bukan local minima. Setiap pixel yang bukan *local minima* ditelusuri dengan mengacu kepada *gradien slopenya* untuk mencari *local minima* yang terdekat. Proses ini akan menghasilkan l*ocal minima* terpilih untuk tiap pixel yang bukan merupakan *local minima.*

Local minima terpilih tersebut digunakan dalam proses 4.5 yang digambarkan dalam gambar 4.7 yaitu pelabelan region. Proses tersebut dilakukan dengan melabeli pixel-pixel yang terhubung dengan *local minima* terpilih, dengan nilai unik pada *local minima* dan menghasilkan suatu *connected component* (region terhubung) hingga keseluruhan region teridentifikasi. Pada saat suatu pixel dilabeli sebagai anggota region dari local minima terpilih maka proses penghitungan terhadap informasi setiap region dilakukan dengan memanfaatkan informasi tentang citra graylevel yang asli. Perhitungan tersebut yaitu perhitungan total jumlah pixel yang masuk dalam suatu region, yang merepresentasikan luas suatu region dan serta total intensitas dari setiap pixel yang masuk kedalam region tersebut yang berguna dalam penghitungan *mean* tiap region.

Gambar 4.7 Diagram 4.5 level 3, proses pelabelan region.

Output akhir dari transformasi ini adalah seluruh region-region yang telah terlabeli dengan nomor yang *unik* dan informasi dari setiap region yang didapatkan yang nantinya digunakan dalam proses merging region..

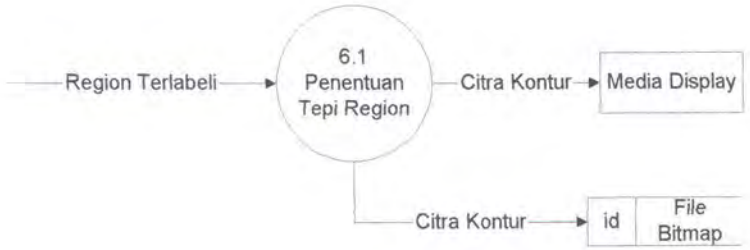
Dalam proses penggabungan region (*merging region*) informasi tentang dua region yang berdekatan dipergunakan oleh proses perbandingan mean yang memanfaatkan informasi luas region dan total intensitas region untuk menghitung mean dari masing masing region.

Selanjutnya dengan treshold merging dilakukan pengujian apakah kedua region tersebut memenuhi syarat untuk di merging. Semakin besar nlai treshold merging maka semakin besar pula probabilitas kedua buah region yang berdekatan untuk digabung menjadi sebuah region.

Gambar 4.8 menunjukkan Diagram Alir Data untuk proses penggabungan region

Gambar 4.8 Diagram 5 level 2, proses penggabungan region

Proses terakhir adalah dengan menentukan tepi tepi tiap region yang didapatkan yang merepresentasikan kontur dari citra asli.



Gambar 4.9 Diagram 6 level 2, proses penentuan tepi region

4.2 Pembuatan Perangkat lunak

Sub Bab ini membahas tentang pembuatan perangkat lunak yang diimplementasikan kedalam program. Program dibuat dengan mengunakan compiler Borland Delphi 3.0 yang menggunakan dasar bahasa pemrograman bahasa *pascal* . Pembuatan perangkat lunak ini meliputi implementasi data serta implementasi proses

4.2.1 Implementasi Data

Data citra dua dimensi adalah merupakan data matrik dua dimensi. Struktur data yang sesuai untuk data ini adalah sebuah *array* dua dimensi yang dapat dibangun dengan *array* satu dimensi dengan metode penyimpanan *row major*.

Dalam proses transformasi watershed , ukuran citra sebelum proses dan sesudah proses adalah sama, untuk itu struktur data yang dipilih adalah *array dinamis* dan bukan *linked list*, dimana pada saat proses berjalan , ukuran array ini tidak berubah. Ukuran array berubah ubah pada saat awal proses pembuatan array pertama kali. Untuk itu ukuran array sangat tergantung pada ukuran dari data gambar. Besar ukuran dari array adalah sebanding dengan ukuran dari citra,dimana jumlah elemen array adalah sama dengan banyak pixel yang terdapat dalam citra tersebut.

Tipe dari elemen array sangat berkait dengan tipe operasi yang menghasilkan data tersebut. Sebagai contoh konvolusi dengan *kernel gaussian* menghasilkan nilai-nilai *floating point*, sehingga elemen array untuk menampung data hasil operasi tersebut adalah *floating point*. Elemen array yang menampung citra grayscale adalah *byte* karena ukuran setiap elemen pixel dalam citra grayscale tersebut adalah byte.

Data data yang tergolong dalam array dinamis dengan elemen array *real* adalah morph_data, Magvalue dan Smoothedim. Array dengan elemen byte adalah dataImage yang menyimpan data citra gray level. Watervalue merupakan array dengan elemen integer yang menyimpan koefisien-koefisien dalam proses

transformasi hingga menjadi data region-region yang telah terlabeli. Array-array tersebut digenerate dengan ukuran yang sama dengan ukuran gambar sebagai berikut

Untuk array dengan elemen byte

*Getmem([nama data],(lebar gambar * tinggi gambar) * sizeof byte)*

Untuk array dengan elemen *integer*

*Getmem([nama data],(lebar gambar * tinggi gambar) * sizeof integer)*

Untuk array dengan elemen real

*Getmem([nama data],(lebar gambar * tinggi gambar) * sizeof real)*

Data data dari setiap tahapan proses tetap disimpan dan bukan data akhir proses transformasi dengan tujuan tahapan-tahapan proses secara *gradual* dapat dilihat hasilnya dengan menampilkan data masing-masing proses sebagai sebuah citra.

Region merupakan data yang menyimpan informasi dari setiap region yang terbentuk. Implementasi data ini merupakan array dinamis dengan elemen array merupakan record yang berisi informasi setiap region Elemen-elemen array yang menyimpan informasi setiap region yang terbentuk adalah sebuah record yang diim-plementasikan sebagai berikut:

```
type
RegionRec = record
        center : byte;
                size : integer;
                parent :integer;
        total_intensitas : integer;
end;
```

Array dinamis region dibangun pada saat proses transformasi watershed yang menghasilkan sejumlah *local minima* yang merepresentasikan jumlah region yang terbentuk. Proses pembangunan array tersebut adalah sebagai berikut.

```
RegionCount := MinCount; // jumlah local minima
GetMem (Region , regionCount * sizeof (regionRec) );
```

Adj_Info merupakan data array buffer untuk menyimpan informasi dua buah region berdekatan yang akan digabung dengan elemen array berupa record yang menyompan informasi dua buah region yang berdekatan. Record tersebut adalah sebagai berikut :

```
type
InfoRec= record
            R1:integer;
            R2:integer;
            Count:integer;
end;
```

Array dinamis Adj_Info dibangun setelah array dinamis region dibangun, karena ukurannya mengacu kepada ukuran region yang terbentuk. Proses pembangunannya adalah sebagai berikut :

```
InfoCount:=RegionCount *5+1000; // buffer cadangan
GetMem (Adj_Info , InfoCountt * sizeof (regionRec) );
```

4.2.2 Implementasi Proses

Perangkat lunak dalam tugas akhir ini diimplementasikan dalam enam tahapan proses yaitu reduksi *noise*, *blur* dengan *kernel gaussian, gradien magnitude, merging region* dan penentuan tepi region.

4.2.2.1 Reduksi Noise

Reduksi Noise dilakukan dengan filter *average.* Dalam prakteknya proses ini dilakukan dengan menjumlahkan seluruh intensitas pixel tetangga dalam keterhubungan delapan termasuk dirinya sendiri, dan dibagi dengan sembilan yang menghasilkan intensitas baru pixel tersebut

Implementasi proses untuk tahapan tersebut adalah sebagai berikut.

```
For I = 0 to tinggi citra –1
  For j = 0 to lebar citra –1
    Smooth_data^[y,x] = ( DataImage^[y-1,x-1]+ dataImage^[y-1,x]+ dataImage^[y-1,x+1]+
                          DataImage^[y,x-1]+ dataImage^[y,x]+ DataImage^[y,x+1]+
                          DataImage^[y+1,x-1]+ DataImage^[y+1,x]+ DataImage^[y+1,x+1] ) / 9;
```

4.2.2.2 Blur Dengan Kernel Gaussian

Tahapan dalam proses ini dibagai kedalam dua tahap. Tahap pertama adalah membangun *kernel gaussian* dengan memanfaatkan input *sigma gaussian* . Tahap selanjutnya adalah melakukan *konvolusi* citra dengan kernel gaussian kearah vertikal dan dilanjutkan dengan konvolusi kearah horisontal.

Implementasi proses untuk tahapan pembangunan kernel adalah sebagai berikut. Langkah pertama adalah menentukan ukuran kernel dan selanjutnya dilakukan pembangunan serta pengisian *kernel gaussian.*

```
// generate ukuran kernel dan create kernel
windowsize := trunc (1+ 2* ceil (2.5* g_sigma));
getMem(kerNel,(windowSize+1)*sizeof(real));
Center := trunc (windowSize /2) ; //
  for i:= 0 to WindowSize-1 do
   begin
      x:= i-center;
      fx:=power(2.71828, -0.5*x*x/ (g_sigma*g_sigma)) /
        (g_sigma * sqrt (6.2831852));
      kernel^[i]:=fx;
      sum:= sum+fx;
   end;
   for i:= 0 to windowsize-1 do
            kernel^[i] := kernel^[i] / sum;
```

Tahapan selanjutnya adalah *blur* kearah horisontal dengan *kernel gaussian* yang diimplementasikan kedalam proses sebagai berikut :

```
{ blur ke arah x / horisontal}
  for r:= 0 to tinggi_citra -1 do
    for c:= 0 to lebar_citra -1 do
      begin
        pos := r*lebar_citra + c; dot:=0.0;  sum := 0.0;
       for cc := -(center) to center do
        begin
          if (((c+cc) >= 0 ) and ((c+cc) < kolom)) then
          begin
            dot := dot + dataImage^[r*kolom+(c+cc)] * kernel^[center+cc];
            sum := sum + kernel^[center+cc];
          end;
        end;
        temporary^[pos]:= dot/sum; // buffer untuk blur arah horisontal
      end;
  end;
```

Tahapan terakhir dari proses ini adalah blur ke arah vertical yang diimplementasikan sebagai berikut.

```
{ blur ke arah y}
   for c:= 0 to lebar citra -1 do
      for  r:= 0 to tinggi citra -1 do
       begin
          pos:=r*lebar citra+ c;
          sum:=0.0;
          dot:=0.0;
          for rr:= (-center) to center do
          begin
             if (((r+rr) >= 0 ) and ((r+rr) < baris)) then
             begin
                dot:= dot+temporary^[(r+rr)*lebarcitra+c] * kernel^[center+rr];
                sum := sum+kernel^[center+rr];
             end;
          end;
          GaussData^[pos] :=  (dot/sum );
       end;
  end;
```

4.2.2.3 Perhitungan Gradien Magnitude Citra

Tahapan proses ketiga adalah perhitungan *gradient magnitude* keseluruhan pixel-pixel dalam citra hasil proses *blur* dengan *gaussian.*

```
for y = 0 to tinggiCitra - 1 do
  for x = 0 lebarCitra -1 do
  begin
        delta_x = Gauss^[y*lebarCitra+x+1]-Gauss^[y*lebarCitra+x];
        delta_y = Gauss^[y*lebarCitra+lebarCitra+x]-Gauss^[y*lebarCitra+x];
         magnitude = sqrt ((delta_x*delta_x)- (delta_y*delta_y))
  end;
```

4.2.2.4 Proses Transformasi Watershed

Proses keempat adalah transformasi watershed. Inti dari proses keempat adalah penentuan *local minima*, penentuan *gradien slope*, penelusuran *gradien slope* untuk mencari *local minima* ,serta pelabelan region.

Dalam prakteknya ,implementasi proses penentuan *local minima* sekaligus akan menghasilkan *gradien slope* yang bukan merupakan local minima. Nilai *gradien slope* adalah nilai negatif dari koefisien posisi *slope* yang digambarkan sebagai berikut.

0	-1	-2
-3	+	-5
-6	-7	-8

Gambar 4.10 gradien slope.

Nilai diatas merupakan nilai negative dari koefisien posisi tetangga pada saat proses perulangan (*looping*) terhadap keseluruhan tetangga. Sedangkan nilai local minima didapatkan apabila intensitas pixel itu sendiri (pixel pusat) paling kecil diantara tetangga-tetangganya dalam keterhubungan 8. Nilai *local minima* ini merupakan nilai integer positif yang dimulai dengan 1 yang akan langsung ditambah dengan satu apabila telah digunakan untuk melabeli sebuah *local minima*.

Implementasi proses untuk tahapan diatas dalam program diimplementasikan sebagai berikut.

```
// Inisialisasi terhadap tepi citra karena keterhubungan yang dipakai adalah keterhubungan 8
 mincount =1 ;
   for y := 0 to baris-1 do
   for x := 0 to kolom -1 do
   begin
        if (y = 0) or (x = 0) or ( y = baris -1 ) or ( x = kolom -1) then
           waterValue^[y*kolom+x]:=mincount // output watershed di set dengan 1
         else
          waterValue^[y*kolom+x]:=0;
    end;

// penentuan local minima dan gradien slope
for y = 1 to baris -2 do
 for x:= 1 to kolom –2 do
 begin
   minValue := magValue^[y*kolom+x]; // pixel yang di uji

// pemrosesan terhadap keseluruhan tetangga dalam keterhubungan 8
// dimulai dari tetangga kiri atas

   for ArahY :=0 to 2 do
    for arahX := 0 to 2 dojk
pencarian nilai minimal terhadap keseluruhan tetangga
   if (magValue^[(y+arahY-1)*kolom + (x+arahX-1)] <= minValue)  then
     begin
        minValue:=magValue^[(y+arahY-1)*kolom + (x+arahX-1)];
        MinIndex:= arahY*3 + arahX;
     end;

  // jika nilai paling kecil adalah dia sendiri maka.
  if (minIndex = 4 ) then
     begin
              minCount:=minCount+1; // increment terhadap indeks local minima
              waterValue^[y*kolom+x]:= minCount;
     end
    else
   // jika bukan maka tentukan gradien slope
     begin
          waterValue^[y*kolom+x]:= -MinIndex;
     end;
end;
```

Sebagai contoh apabila nilai slope didapatkan pada tetangga di posisi kiri atas maka nilai *gradien slope* dari pixel tersebut adalah –2. Sedangkan apabila nilai *gradien slope* didapatkan pada posisi tetangga tepat sebelah bawah maka nilai *gradien slope* pixel tersebut adalah –7. Apabila diantara pixel-pixel tetangga tersebut terdapat nilai-nilai yang lebih kecil dari pixel pusat dan intensitasnya sama serta jumlahnya lebih dari satu, maka nilai slope yang diambil adalah nilai slope yang terakhir. Sebagai contoh apabila nilai intensitas pixel pusat adalah 25, sedangkan terdapat pixel tetangga di posisi indek -1 dan -8 dengan intensitas 10, maka indeks yang dijadikan gradien slope adalah -8.

Tahapan diatas dimulai dari pixel dalam koordinate (1,1) karena mengacu kepada konsep keterhubungan 8. Untuk itu untuk seluruh pixel dalam koordinat (0,y) dan koordinat (lebar_citra,y), ataupun pixel dalam koordinat (x,0) dan koordinat (x,tinggi_citra) nilainya di set dengan nilai satu.

Tahapan selanjutnya adalah penelusuran *gradien slope* seluruh pixel hingga mencapai *local minima* dan diberi indeks yang sama dengan indeks *local minima.* Dalam penelusuran tersebut setiap pixel yang dilewati dicatat untuk kemudian apabila *local minima* telah didapatkan maka dalam proses pelabelan region dilakukan terhadap keseluruhan pixel-pixel dimana penelusuran dimulai serta keseluruhan pixel-pixel yang dilewati dalam penelusuran tersebut.

Hasil yang didapat dari proses diatas adalah *local minima* terpilih dan gradien slope pixel-pixel yang bukan merupakan *local minima.*

Implementasi proses untuk tahapan diatas adalah sebagai berikut.

```
// Proses penelusuran mengacu kepada ketrerhubungan 8 sehingga tepi citra tdk di proses
for y:= 1 to baris - 2 do
      for x:= 1 to kolom - 2 do
      begin
        arahX:=x;
        arahY:=y;
        count:=0;
      while (waterValue^[arahY*kolom+arahX] <= 0) do // selama pixel  bukan local minima
        begin
          case waterValue^[arahY*kolom+arahX] of  // nilai gradien slope diambil
           0: begin // kiri atas
                arahX:=arahX-1;
                arahY:= arahY-1;
              end;
           -1: begin  // atas
                arahY:=arahY - 1
              end;
           -2: begin  // kanan atas
                arahX:=arahX+1;
                arahY:=arahY - 1;
              end;
           -3: begin // kiri
                arahX:=arahX-1;
              end;
           -5: begin // kanan
                arahX:=arahX+1;
              end;
           -6: begin // kiri bawah
                arahX:=arahX-1;
                arahY:=arahY + 1;
              end;
           -7: begin // bawah
                arahY:=arahY + 1;
              end;
           -8: begin  // kanan bawah
                arahX:=arahX+1;
                arahY:=arahY + 1
              end;
          end ;
        path^[count]:=arahY*kolom+arahX; // pixel yg dilewati dicatat
        inc(count);
       end;
// label pixel diberi indeks sama dengan local minima terpilih.
    waterValue^[y*kolom+x]:=waterValue^[arahY*kolom+arahX];
```

Dalam tahapan diatas sekaligus dapat juga dilakukan pelabelan region dan penghitungan informasi region, dengan memanfaatkan informasi indeks region yang didapat dari local minima terpilih serta data citra *graylevel* asli.

Implementasi untuk tahapan diatas adalah sebagai berikut.

```
// pelabelan region

// pixel(x,y) diberi label dengan indeks local minima
waterValue^[y*kolom+x]:=waterValue^[arahY*kolom+arahX]; // indeks local minima

// nilai indeks dicatat
index:=waterValue^[arahY*kolom+arahX];

// array region , untuk menyimpan informasi tiap region diinisialisasi
region^[index].center :=dataImage^[arahY*kolom+arahX]; // center = local minima
region

// setiap ada pixel yang diberi label sebagai anggota region , size region diupdate.
 region^[index].size:= region^[index].size + 1;

// inisialisasi parent region
 region^[index].parent:=-1;

// total intensitas region dihitung dengan mengacu kepada intensitas citra graylevel asli
 // sumI1 = total intensitas di suatu region.
 region^[index].total_Intensitas:= region^[index].SumI1 + dataImage^[y*kolom+x];


// proses pelabelan region untuk pixel-pixel yang dilewati
for num := 0 to count-1 do
begin
    waterValue[path^[num]]:=waterValue^[arahY*kolom+arahX];
    region^[index].size:= region^[index].size + 1;
   region^[index].total_Intensitas:= region^[index].SumI1 + dataImage^[path^[num]];
 end;
```

Hasil yang didapat dari tahapan diatas adalah region-region yang terlabeli , serta informasi-informasi dari setiap region. Informasi-informasi ini digunakan sebagai acuan untuk melakukan tahapan berikutnya yaitu penggabungan region.

Tahapan berikutnya adalah tahapan pengabungan region yang dilakukan untuk mengantisipasi kemungkinan adanya *oversegmented* agar hasil akhir yang didapatkan baik. Tahapan ini diawali dengan menyimpan informasi dua buah region yang berdekatan (*adjacent region*) untuk selanjutnya dibandingkan nilai mean masing masing region dengan batasan treshold merging yang didapat dari user.

```
// Simpan informasi dua region yang berdekatan
for x:= 1 to kolom -2 do

for y:= 1 to baris -2  do
 begin
      if (waterValue^[y*kolom+x] <> waterValue^[(y-1)*kolom+x]) then
       saveReg(waterValue^[y*kolom +x],waterValue^[(y-1)*kolom+x]);
       if (waterValue^[y*kolom+x]<> WaterValue^[y*kolom+x-1]) then
       saveREg(waterValue^[y*kolom+x],waterValue^[y*kolom+x-1]);
  end;
 end

// proses penyimpanan region dalam array info

index1:= REGION1
info^[index1].R1 := REGION1;
info^[index1].R2 := REGION2;
Info^[index1].count:=1;

// proses pembandingan bandingkan mean dari masing masing region
// infocount merupakan jumlah dua region yang berdekatan yang akan digabung.

for i := 0 to infocount -1 do
    begin
        if ((info^[i].count > 0) and (parent(info^[i].R1) <> parent(info^[i].R2))) then
        if (meandiff(info^[i].R1,info^[i].R2)< threshold) then
          begin
             merge(info^[i].R1,info^[i].R2);
             inc(count);
          end;
    end;
```

Tahapan diatas dilanjutkan dengan proses merging terhadap region regon yang memenuhi kriteria dari treshold merging. Tahapan ini dilakukan dengan menggabungkan seluruh informasi dua buah region yang digabung. Penggabungan ini meliputi penggabungan luas masing-masing region menjadi luas region baru yang terbentuk. Penggabungan juga mencakup panggabungan total intensitas menjadi total intensitas baru serta penyamaan nilai intensitas *local minima.*

Implementasi proses dari tahapan diatas adalah sebagai berikut.

```
// proses merging region
// region1 & region2 merupakan nomor region
// Penggabungan dibawah ini dilakukan dengan menggabungkan region2 ke region1

   region^[region2].parent:= region1;
    region^[region2].center := region^[region1].center;
    region^[region1].size:=region^[region1].size+region^[region2].size;
    region^[region1].Suml1 := region^[region1].Suml1 + region^[region2].Suml1;
```

Tahapan terakhir adalah dengan melabeli region dengan nomor region baru sebagai hasil dari proses penggabungan region.

```
// renumber watershed region

for y:= 0 to baris -1 do
 for x:= 0 to kolom -1 do
  begin
     Region1:= waterValue^[y*kolom+x];
     while region^[region1].parent > 0 do
        region1 := region^[region1].parent;
        waterVAlue^[y*kolom+x]:= region1;
  end;
```

Hasil dari proses ini adalah region terlabeli akhir. Proses selanjutnya adalah menentukan tepi dari tiap region. Proses ini dapat dilakukan dengan mudah, salah satunya adalah dengan mengindentifikasi setiap perubahan nilai pixel dalam region terlabeli akhir tesebut sebagai tepi dari region. Implementasi tersebut dapat dilakukan secara mudah dengan proses gradien magnitude. Apabila magnitude lebih besar dari 0 maka merupakan informasi tepi region apabila tidak maka bukan tepi dari region (titik tersebut berada ditengah region).

BAB V

UJI COBA DAN EVALUASI

BAB V

UJI COBA DAN EVALUASI

Pada Bab ini akan dijelaskan mengenai uji coba dan hasil evaluasi perangkat lunak yang dibuat mulai dari input (citra asli) sampai dengan output (citra hasil pemrosesan) dari program untuk dianalisa.

Kecepatan program, banyaknya region yang terbentuk serta akurasi kontur yang terbentuk merupakan acuan dalam melakukan analisa uji coba yang dilakukan. Nilai-nilai treshold yang dimasukkan kedalam program diperlihatkan pengaruhnya terhadap hasil proses secara keseluruhan.

5.1 Perangkat Uji Coba

Uji coba dilakukan pada komputer dengan spesifikasi sebagai berikut :

- Prosessor Pentium 166
- Memori 32 MegaByte
- Hard Disk 1,7 GigaByte
- Memori VGA 4 MegaByte

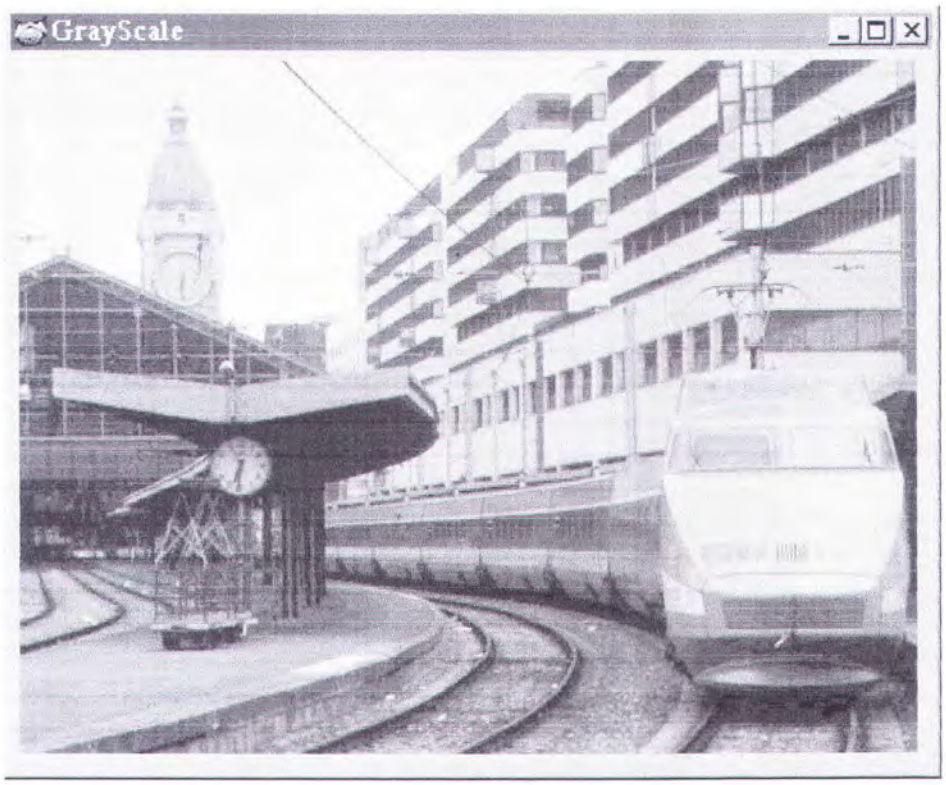
Hasil yang diperlihatkan adalah citra kontur , citra asli serta hasil *mapping* kontur kedalam citra sample. Hal ini dilakukan karena pengukuran akurasi kontur tidak dapat dilakukan secara kuantitatif, tetapi dapat dilihat secara visual Tidak ada satupun referensi yang dapat digunakan untuk menyatakan ukuran kuantitatif kontur hasil proses segmentasi.

5.2 Uji Coba terhadap Citra Sample.

Secara umum faktor utama yang berpengaruh terhadap kecepatan program adalah ukuran dari citra itu sendiri. Semakin besar ukuran citra maka semakin besar pula data yang harus diproses.

Nilai-nilai yang dimasukkan oleh user dalam program sebanyak tiga buah. Nilai pertama adalah *sigma gaussian* yang menentukan ukuran *kernel gaussian*. Nilai kedua adalah treshold magnitude yang membatasi nilai magnitude dari setiap pixel yang diterima. Nilai yang ketiga adalah nilai treshold merging yang menjadi kriteria penggabungan dua buah region yang berdekatan.

5.2.1 Uji Coba Kesatu dengan Citra Sample Train.bmp

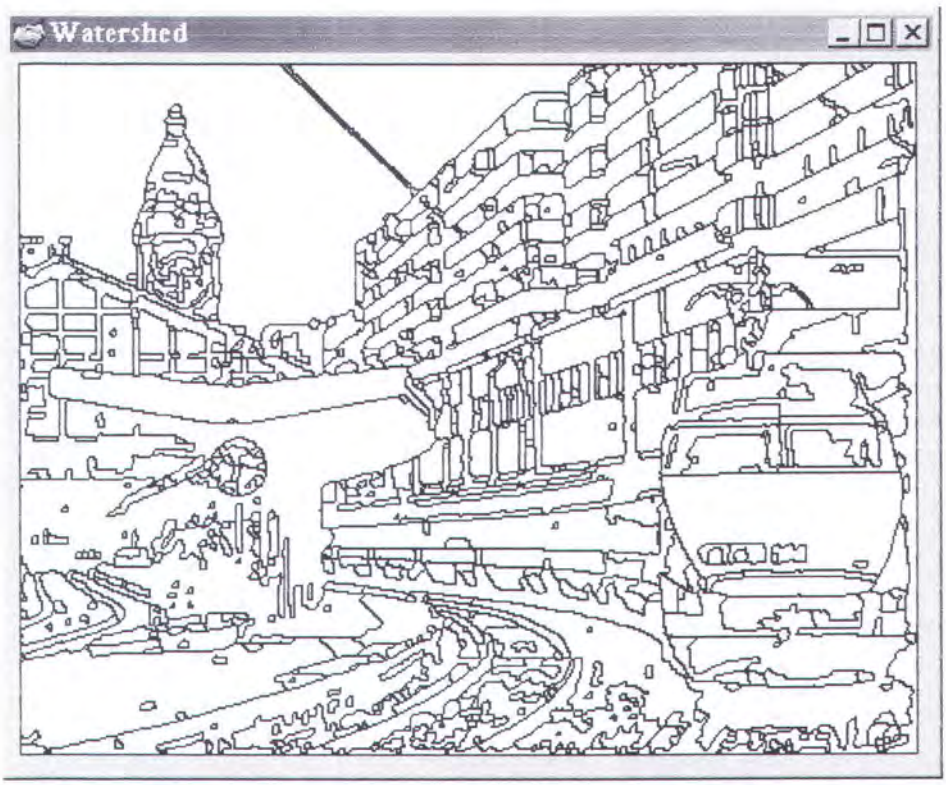


Gambar 5.1 Train.bmp

Untuk uji coba pertama ,nilai-nilai treshold serta sigma gaussian masukan adalah sebagai berikut:

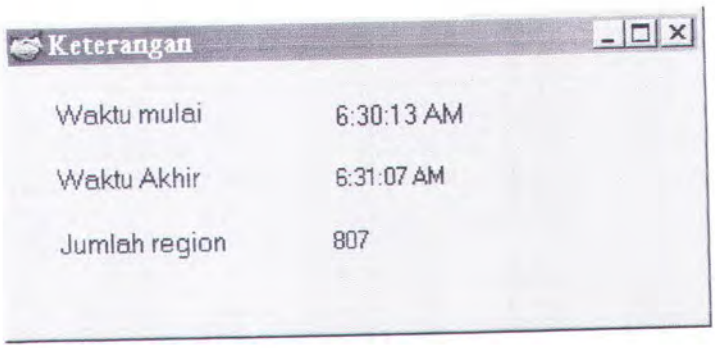
Ukuran Gambar = 450 x 343 ,
Sigma Gaussian = 0.35 ,
Treshold Magnitude = 2
Treshold Merging = 15

Citra Kontur yang didapatkan adalah sebagai berikut :



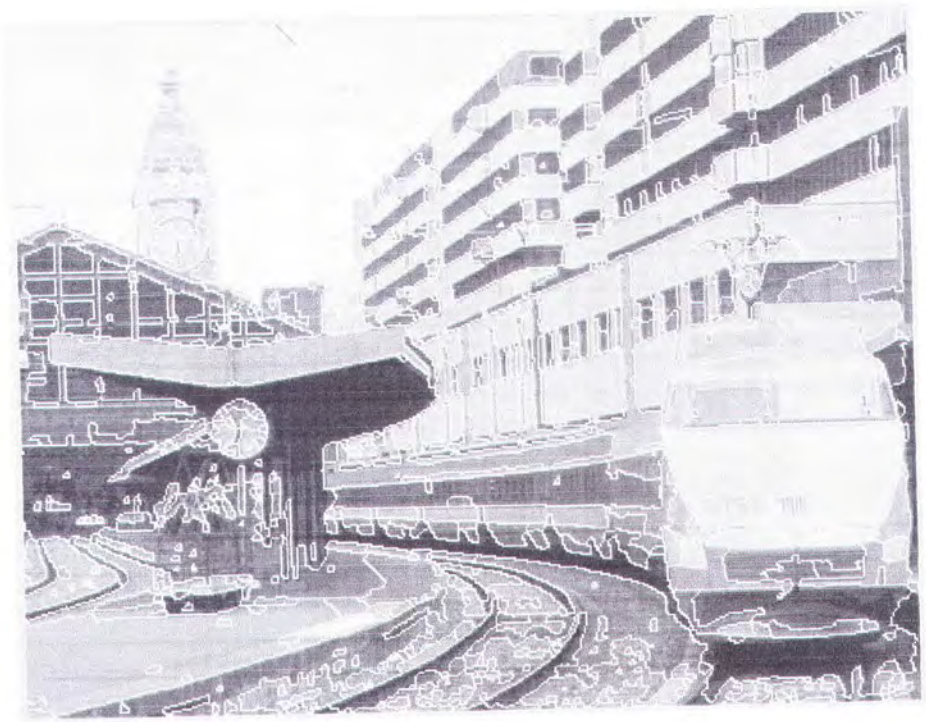
Gambar 5.2 Citra kontur untuk ujicoba pertama.

Informasi waktu proses serta region yang dihasilkan adalah sebagai berikut



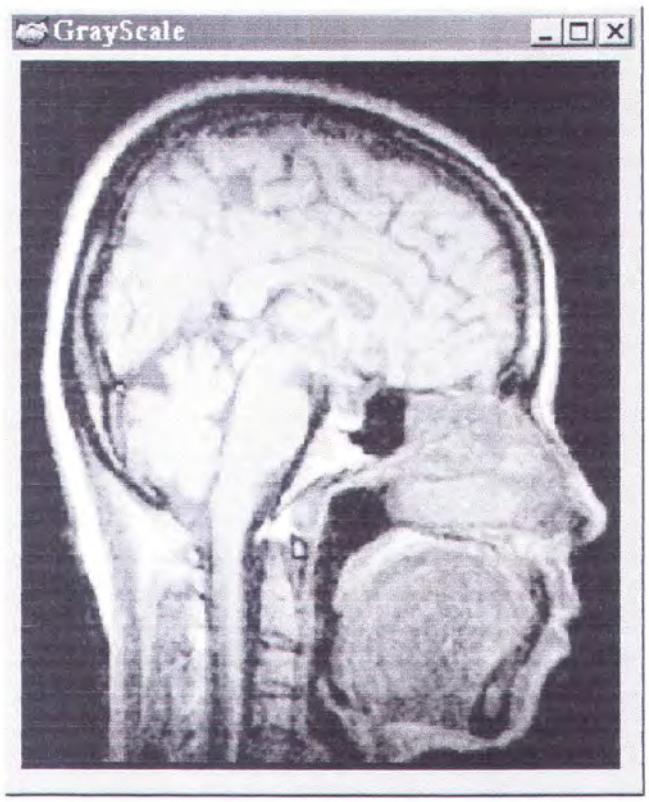
Gambar 5.3. Informasi ujicoba pertama

Hasil mapping kontur pada citra asli adalah sebagai berikut :



Gambar 5.4 Mapping kontur pada ujicoba pertama

5.2.2 Uji Coba Kedua dengan Citra Tengkorak.bmp

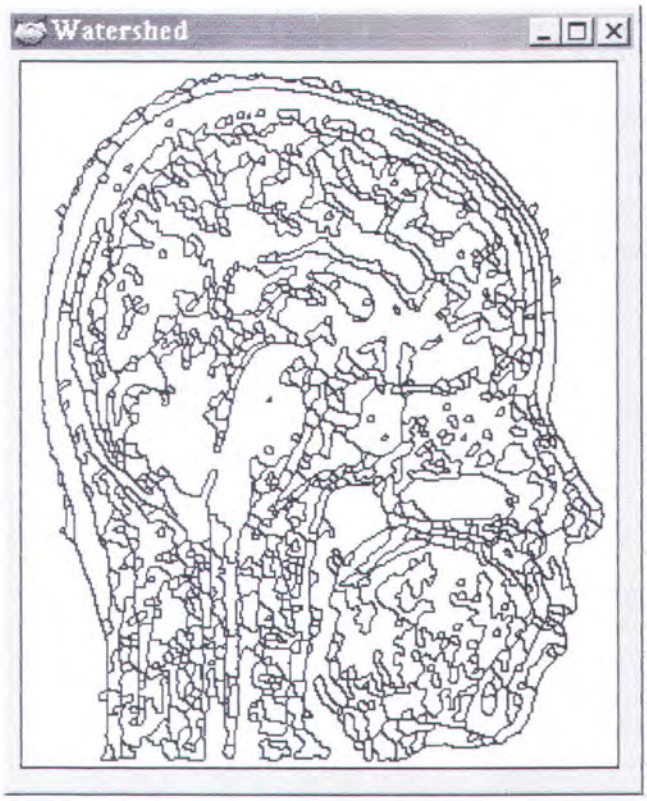


Gambar 5.5 Tengkorak.Bmp

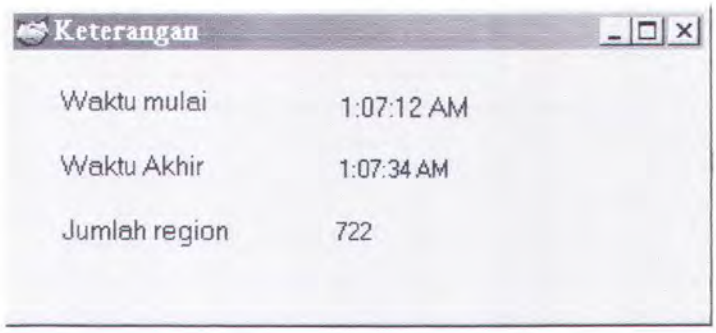
Untuk uji coba kedua, ukuran citra, nilai-nilai treshold serta sigma gaussian adalah sebagai berikut:

Ukuran Gambar = 300 x 350 ,
Sigma Gaussian = 0.35 ,
Treshold Magnitude = 3
Treshold Merging = 15

Citra Kontur yang didapatkan dari ujicoba kedua adalah sebagai berikut.



Gambar 5.6 Citra Kontur tengkorak.bmp



Gambar 5.7 Informasi hasil ujicoba kedua

Proses mapping kontur pada citra asli dilakukan dengan cara mengubah warna pixel-pixel pada citra asli yang diidentifikasikan sebagai sebuah kontur. Hasil mapping kontur pada citra asli untuk ujicoba kedua diilustrasikan dalam gambar 5.8 dibawah ini

Gambar 5.8 Mapping kontur pada ujicoba kedua.

Mapping kontur pada citra aslinya diatas dimaksudkan untuk melakukan pengecekan terhadap akurasi kontur yang didapatkan oleh perangkat lunak yang dibuat.

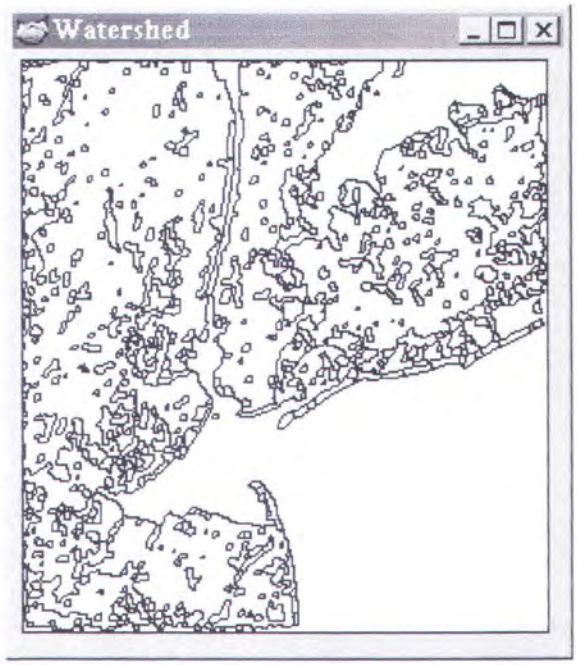
5.2.3 Uji coba ketiga dengan citra Newyork.Bmp

Gambar 5.9 Citra Newyork.Bmp.

Ukuran citra dan nilai nilai treshold masukan adalah sebagai berikut.

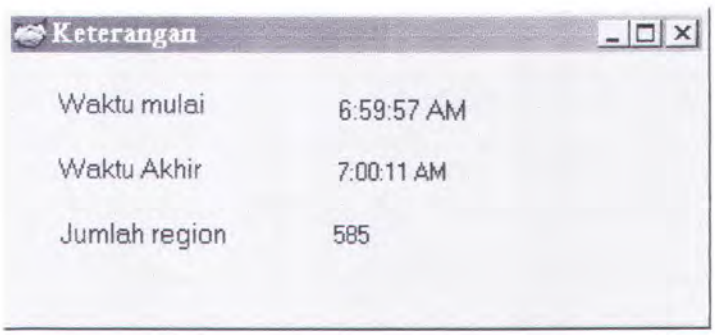
Ukuran Gambar = 264 x 284
Sigma Gaussian = 0.35 ,
Treshold Magnitude = 6
Treshold Merging = 16

Citra kontur yang didapatkan untuk uji coba ketiga diatas adalah sebagai berikut



Gambar 5.10 Citra kontur Newyork.bmp

Informasi untuk yang didapatkan untuk uji coba ketiga diatas adalah sebagai berikut.



Gambar 5.11 Informasi ujicoba ketiga.

Hasil mapping kontur pada citra asli untuk ujicoba keempat adalah sebagai berikut.

Gambar 5.12 Mapping kontur untuk ujicoba ketiga

5.3 Analisa Hasil.

Analisa yang dilakukan adalah dengan melakukan evaluasi terhadap citra kontur yang dihasilkan , lama waktu proses, jumlah region yang didapatkan serta pengaruh nilai- nilai treshold masukan.

5.3.1 Akurasi Kontur yang didapatkan.

Pengukuran akurasi kontur tidak dapat dilakukan secara kuantitatif, tetapi dapat dilihat secara visual. Hal ini karena tidak ada satupun referensi yang dapat

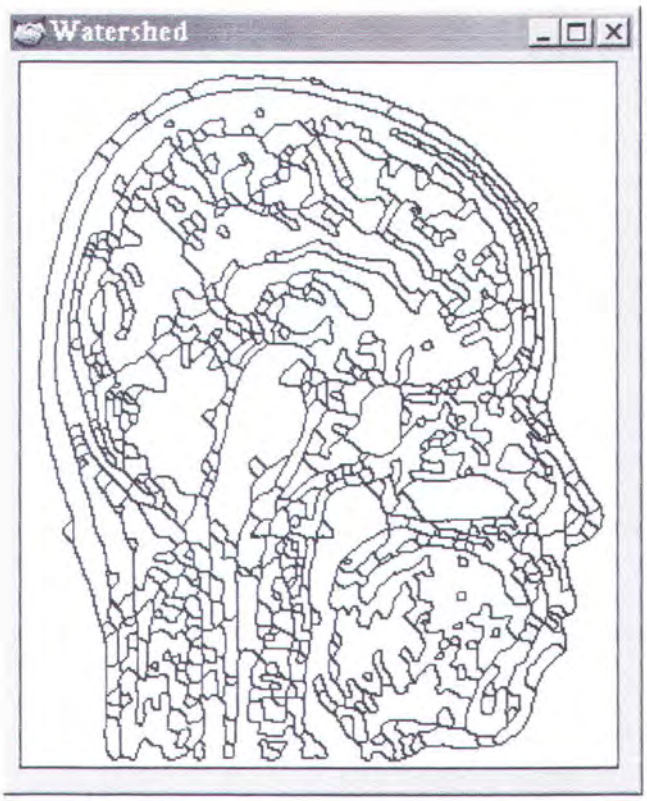
5.3.3 Pengaruh nilai-nilai treshold masukan .

5.3.3.1 *Sigma gaussian*

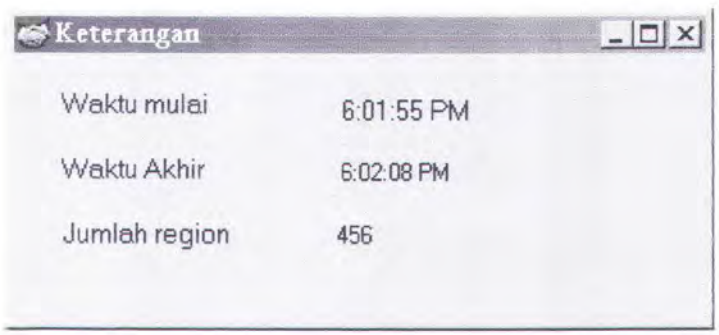
Sigma gaussian yang diinputkan berpengaruh secara langsung terhadap ukuran *kernel gaussian* yang digunakan untuk mengkonvolusi citra. Hal tersebut mengakibatkan citra menjadi semakin kabur . Efek yang diakibatkan dari citra semakin kabur adalah tepi tepi yang semakin kabur dan melebar bahkan cenderung hilang sehingga banyak kontur yang hilang, dengan sendirinya jumlah region yang didapatkan akan semakin sedikit dan waktu proses juga semakin cepat.

Gambar 5.13 Citra hasil blur dengan sigma = 1.3

Hasil segmentasi untuk citra yang semakin kabur diatas adalah sebagai berikut



Gambar 5.14 Kontur yang didapatkan untuk ujicoba keempat



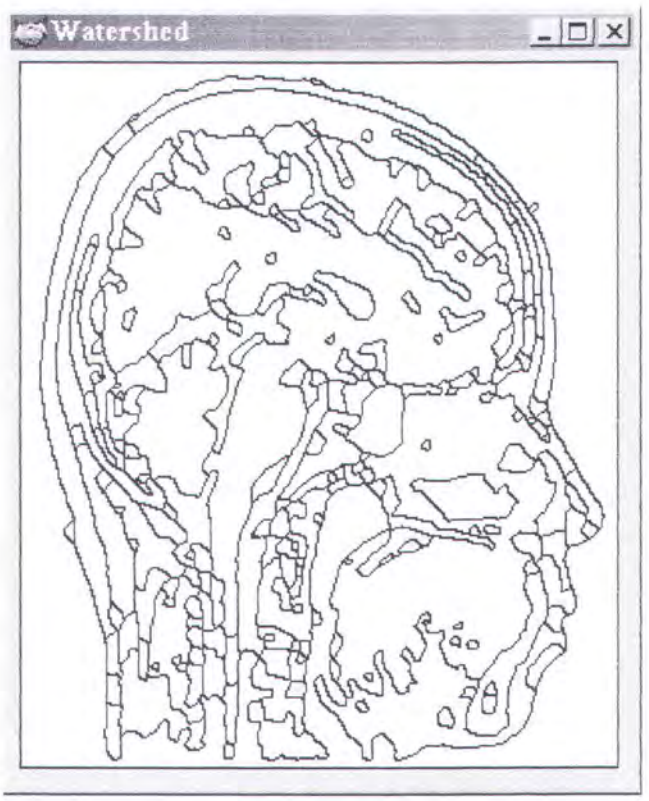
Gambar 5.15 Informasi ujicoba keempat.

Hasil diatas apabila dibandingkan dengan hasil dalam uji coba kedua secara visual diperlihatkan bahwa kontur-kontur yang didapatkan lebih lebar dan

region yang didapatkan jumlahnya lebih sedikit sehingga waktu proses yang juga semakin cepat.

5.3.3.2 Treshold Merging .

Treshold merging digunakan sebagai batasan untuk penggabungan dua buah region yang berdekatan . Nilai treshold merging yang semakin besar mengakibatkan kemungkinan dua buah region yang berdekatan untuk digabung menjadi lebih besar. Dalam gambar 5.16 ditunjukan perubahan kontur yang didapat dari ujicoba keempat dengan mengubah treshold merging menjadi 25.

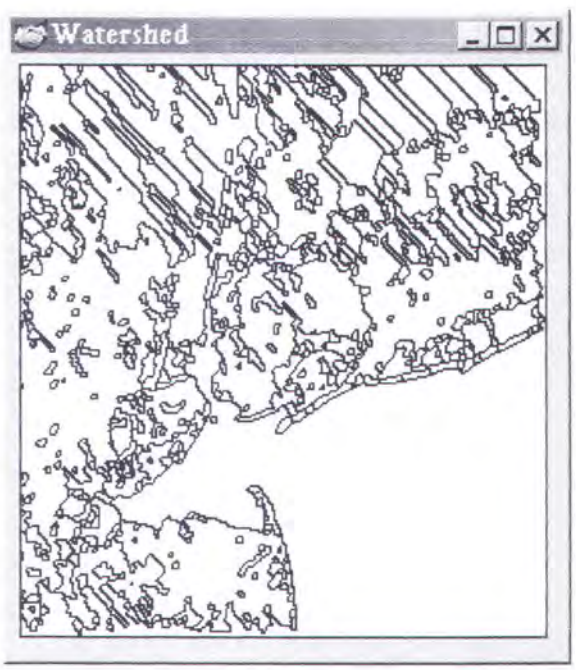


Gambar 5.16 Merging dengan treshold 25 untuk ujicoba keempat

5.3.3.3 Treshold Magnitude.

Nilai treshold magnitude diberikan untuk memberikan batasan-batasan kontur yang diproses. Tujuan yang diharapkan dari pemberian nilai treshold ini adalah agar noise pada citra dapat dikurangai serta kontur kontur yang sangat tipis tidak muncul sehingga mampu mengurangi terjadinya over segmented. Apabila treshold magnitude diberikan terlalu besar akan menyebabkan hasil segmentasi yang tidak memuaskan karena banyak kontur yang hilang sehingga struktur citra yang akan di proses dalam transformasi berubah.

Ilustrasi ujicoba kelima dalam gambar 5.17 di bawah ini menunjukkan pemberian nilai treshold magnitude yang terlalu besar untuk ujicoba ketiga (treshold = 9) sehingga hasil segmentasi tidak tepat.



gambar 5.17. Citra kontur untuk uji coba ketiga untuk treshold magnitude yang terlalu besar.

5. 4 Ujicoba Algoritma .

Dalam sub bab berikut akan dilakukan pengujian transformasi watershed untuk melakukan segmentasi terhadap nilai-nilai piksel dari citra yang berukuran 10 x 10. Penjelasan ini diperlukan untuk menunjukkan perubahan nilai-nilai dari setiap piksel yang terjadi dalam proses transformasi watershed. Citra masukan untuk ujicoba tersebut ditunjukkan dalam gambar 5.18 sebagai berikut

3	3	3	8	8	8	8	5	5	5
3	3	3	8	8	8	8	5	5	5
3	3	3	8	8	8	8	5	5	5
3	3	3	8	8	8	8	5	5	5
7	7	7	7	7	2	2	2	2	2
7	7	7	7	7	2	2	2	2	2
7	7	7	7	7	2	2	2	2	2
4	4	4	4	4	4	4	9	9	9
4	4	4	4	4	4	4	9	9	9
4	4	4	4	4	4	4	9	9	9

Gambar 5.18. Citra masukan.

Dengan menganggap bahwa citra diatas telah bebas dari *noise* maka proses selanjutnya adalah menghitung nilai *gradien magnitude* dari seluruh piksel yang ada dalam citra. Nilai *gradien magnitude* dari citra dari proses tersebut ditunjukkan dalam gambar 5.19 dibawah ini.

0	0	5	0	0	0	3	0	0	0
0	0	5	0	0	0	3	0	0	0
0	0	5	0	0	0	3	0	0	0
4	4	6.40	1	1	6	6.70	3	3	3
0	0	0	0	5	0	0	0	0	0
0	0	0	0	5	0	0	0	0	0
3	3	3	3	5.83	2	2	7	7	7
0	0	0	0	0	0	5	0	0	0
0	0	0	0	0	0	5	0	0	0
0	0	0	0	0	0	5	0	0	0

Gambar 5.19. Gradien Magnitude Citra Masukan.

Proses selanjutnya adalah transformasi watershed terhadap *gradien magnitude* citra yang telah didapatkan. Proses ini akan menghasilkan nilai-nilai *local minima* dan koefisien *gradien slope* dari piksel yang bukan *local minima.* Pemberian nilai koefisen tersebut mengacu pada indeks posisi tetangga yang merupakan *slope* dari sebuah piksel yang bukan *local minima.* Untuk piksel yang merupakan *local minima* diberikan nilai *integer* positif yang unik. Gambar 5.20 berikut menggambarkan koefisen *gradien slope* untuk proses transformasi watershed.

0	-1	-2
-3	+	-5
-6	-7	-8

Gambar 5.20. Koefisien Gradien Slope.

Nilai *local minima* dan *gradien slope* yang didapatkan dari proses transformasi watershed ditunjukkan dalam gambar 5.21 dibawah ini.

-8	-7	-8	-8	-8	-7	-8	-8	-8	-7
-8	-7	-8	-8	-8	-7	-8	-8	-8	-7
-5	1	-5	-5	-5	2	-5	-5	-5	3
-8	-8	-8	-7	-8	-8	-8	-8	-8	-7
-8	-8	-8	-7	-8	-8	-8	-8	-8	-7
-5	-5	-5	4	-5	-5	-5	-5	-5	5
-8	-8	-8	-8	-8	-7	-8	-8	-8	-7
-8	-8	-8	-8	-8	-7	-8	-8	-8	-7
-8	-8	-8	-8	-8	-7	-8	-8	-8	-7
-5	-5	-5	-5	-5	6	-5	-5	-5	7

Gambar 5.21. Gradien Slope dan Local Minima Citra Masukan

Proses penelusuran seluruh *gradien slope* terhadap *local minima* yang ada yang dilanjutkan dengan pelabelan region. Proses ini menghasilkan region-region terlabeli yang merepresentasikan region yang terdapat dalam citra masukan. Gambar 5.22 dibawah ini menunjukkan seluruh region yang didapat dari citra masukan.

1	1	2	2	2	2	3	3	3	3
1	1	2	2	2	2	3	3	3	3
1	1	2	2	2	2	3	3	3	3
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
6	6	6	6	6	6	7	7	7	7
6	6	6	6	6	6	7	7	7	7
6	6	6	6	6	6	7	7	7	7
6	6	6	6	6	6	7	7	7	7

Gambar 5.22 Region-Region dari Citra Masukan

BAB VI

KESIMPULAN DAN SARAN

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Setelah berbagai pembahasan mengenai teori dasar ,perancangan dan pembuatan perangkat lunak, uji coba dan evaluasi hasil yang telah dicoba , maka beberapa kesimpulan yang bisa diperoleh adalah :

- Kecepatan program dalam menemukan kontur sangat dipengaruhi oleh banyaknya kontur yang ada pada citra, sehingga walaupun ukuran citra dalam hal ini sama akan tetapi apabila kontur yang terdapat dalam citra tersebut lebih sedikit maka program akan berjalan dengan lebih cepat.
- Kecepatan program untuk mendapatkan kontur akan semakin bertambah apabila kernel gaussian untuk melakukan *blurring* terhadap citra semakin besar karena kontur yang didapatkan semakin sedikit. Efek negatif yang timbul oleh hal ini adalah kontur kontur yang ukurannya kecil akan hilang dan menggabung menjadi sebuah kontur yang lebih besar sehingga memungkinkan segmentasi terhadap obyek tersebut kurang tepat
- Pemberian nilai treshold magnitude yang terlalu besar mengakibatkan feature citra menjadi rusak sehingga hasil segmentasi tidak tepat.

Sebaliknya pemberian treshold magnitude yang terlalu kecil mengakibatkan hasil segmentasi yang didapatkan berlebih (*over segmented).*

- Transformasi watershed merupakan metode yang handal untuk melakukan segmentasi citra karena mampu melakukan ekstraksi kontur yang ada pada citra secara tepat dan terutama kemampuannya untuk mendeteksi kontur-kontur kecil dan rumit.
- Kemungkinan adanya hasil segmentasi yang berlebih (*over segmented*) pada transformasi watershed dapat diatasi dengan teknik penggabungan region untuk menghilangkan kontur-kontur yang tidak diharapkan.

6.2 Saran

- Pembentukan *catchment basin* serta penentuan batas *watershed* dalam transformasi watershed dengan metode *ranfalling* watershed ini dapat dikembangkan kedalam suatu proses paralel (*paralel process*). Hal ini sesuai dengan realita *rainfalling simulation* (simulasi hujan) dimana turun dan mengalirnya air hujan dalam suatu wilayah tertentu adalah simultan dan bukan satu-persatu.
- Segmentasi citra dengan transformasi watershed mampu menghasilkan kontur-kontur yang bagus walaupun kontur tersebut kecil dan rumit, untuk itu perlu dikembangkan suatu aplikasi interprestasi citra terhadap kontur

yang dihasilkan oleh transformasi ini untuk permasalahan-permasalahan dalam dunia nyata, misalnya interprestasi untuk citra brain (otak) manusia, ataupun interprestasi yang behubungan dengan pemetaan suatu wilayah.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

1. Arnold Meijster, Jos B.T.M. Roerdink, "A Disjoint Set Algorithm for The Watershed Transformation " , *Proc EUSIPCO '98, IX European Signal Processing Conference*, September 1998, Rhodes , Greece.
2. Andreas Bieniek, Alina Moga , " A Connected Component Approach to The Watershed Segmentation " , *In Mathematical Morphology and its Application to Image and Signal Processing*, Volume 12 of *Computational Imaging and Vision*, pages 215-222, Kluwer Academic Publisher, 1998.
3. Alina Moga , " *Paralel Watershed Algorithms for Image Segmentation* " Phd Thesis, Department of Information Technology, Tampere University of Technology, Finland , Februari 1997.
4. Anthony S.Wright, Scott T.Acton, "Watershed Pyramids for Edge Detection" ,Internal Report, The Oklahoma Imaging Laboratory, School of Electrical and Computer Engineering, Oklahoma State University Stillwater, OK 74078, USA.
5. Luc Vincent, Pierre Soille , " Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations ", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol 13, No. 6 , June 1991.
6. Luc Vincent, Pierre Soille, " Determining Watersheds in Digital Pictures Via Flooding Simulations " , *SPIE Vol 1360 Visual Communications and Image Processing* , 1990.

7. Laurent Najman, Michel Schmitt , "Geodesic Saliency of Watershed Contours and Hierarchical Segmentation " *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol 18, No. 12 , December 1996.
8. John M. Gauch , Stephen M. Pizer, " Multiresolution Analysis of Ridges and Valleys in Grey-Scale Images " , *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol 15, No. 6 , June 1993.
9. Ole Folgh Olsen , "*MultiScale Segmentation Of Grey-Scale Images*" , Master Thesis, Department of Computer Science, University of Copenhagen, September 1996.
10. Leila Shafarenko, Maria Petreou, "Automatic Watershed Segmentation of Randomly Textured Color Images", *IEEE Transaction on Image Processing*, Vol 6, No. 11 , November 1997.
11. William K. Prat, "*Digital Image Processing*", John Wiley and Sons, New York 1991.
12. Ioannnis Pitas, "*Digital Image Processing Algorithms*" , Prentice Hall Inc, 1992.
13. Rafael C. Gonzales, "*Digital Image Processing* ", Second Edition , Addison-Wesley Publishing Co, Tennesee, 1987.
14. Joseph Bosworth, Takashi Koshimizu, Scott T.Acton, " Automated Segmentation of Surface Soil Moisture From Landsat TM Data" , Internal Report, *The Oklahoma Imaging Laboratory*, School of Electrical and Computer Engineering, Oklahoma State University Stillwater, OK 74078, USA.

15. Jean Serra , "Image Analysis and Mathematical Morphology" , London Academic , 1982.

16. S.Beucher, C. Lantuejoul. "Use Watershed in Contur Detection" , *Proc .Int. Workshop Image Processing, Real-Time Edge and Motion Detection / Estimation*, Rennes , France , September ,1979