PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK DETEKSI DAN KLASIFIKASI CITRA DENGAN DEFORMABLE CONTOUR

TUGAS AKHIR

Oleh :

JOKO WIJOSENO W. R.
NRP. 2694100051

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2000

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK DETEKSI DAN KLASIFIKASI CITRA DENGAN DEFORMABLE CONTOUR

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
Ir. ESTHER HANAYA, M.Sc.	**RULLY SOELAIMAN, S.Kom.**
NIP. 130 816 212	NIP. 132 085 802

SURABAYA
Februari, 2000

ABSTRAK

Kontur adalah bentuk tertentu dari sebuah obyek berupa garis tepi obyek tersebut. Melalui bentuk kontur dapat dijelaskan ciri dan karakter obyek yang bersangkutan. Kontur dapat dideteksi dan kemudian diklasifikasikan dengan memanfaatkan nilai energi kontur yang terdapat di dalam kontur tersebut yang dapat dihitung bila koordinat kontur dan citra pembentuk kontur diketahui.

Kemampuan kontur untuk dapat berubah bentuk dimungkinkan dengan penggunaan metode persamaan distribusi medan acak Markov (*Markov random field*) yang dialokasikan untuk kontur tersebut, perumusan energi internal kontur, dan energi eksternal yang merupakan korelasi kontur dengan citra. Dari hasil tersebut didapatkan model kontur aktif yang dikenal dengan istilah *generalized active contour model* atau *g-snake*.

Dalam tugas akhir ini dibahas tentang deteksi dan klasifikasi citra dengan memanfaatkan kontur yang dapat berubah bentuk. Untuk proses deteksi *deformable contour* digunakan transformasi Hough untuk mereposisikan kontur agar sesuai dengan obyek yang akan dideteksi. Sedangkan, untuk proses klasifikasi digunakan algoritma penjumlahan (*summation algorithm*) yang akan dijelaskan lebih lanjut dalam buku ini.

KATA PENGANTAR

Puji syukur kepada Alloh S.W.T., karena berkat taufik dan hidayah-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul :

PERANCANGAN DAN PEMBUATAN
PERANGKAT LUNAK DETEKSI DAN KLASIFIKASI
CITRA DENGAN DEFORMABLE CONTOUR

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

Dengan terselesaikannya Tugas Akhir ini, penulis menyampaikan penghargaan dan terima kasih yang sebesar-besarnya kepada mereka yang telah membantu penulis dalam mengatasi kendala-kendala baik teknis maupun psikologis, terutama kepada :

1. Ir. Esther Hanaya, MSc., selaku Dosen Pembimbing I yang telah memberikan bimbingan, arahan, dan kasih sayang selama perkuliahan di Teknik Informatika dan selama penyelesaian Tugas Akhir ini.
2. Rully Soelaiman, S.Kom., selaku Dosen Pembimbing II yang telah memberikan ide dasar, bimbingan, acuan dan materi-materi yang melandasi pembuatan Tugas Akhir ini.

3. Dr. Ir. Arif Djunaidy, MSc., selaku Ketua Jurusan Teknik Informatika ITS dimana penulis menuntut ilmu.
4. Ir. Aris Tjahyanto, M.Kom, selaku Dosen Wali yang telah memberikan perhatian dan saran-sarannya pada setiap awal semester.
5. Seluruh staf dosen dan karyawan Jurusan Teknik Informatika ITS atas bantuannya selama mengikuti perkuliahan.
6. Seluruh keluarga di Yogyakarta, Bapak, Ibu, 'Aik, serta teman-teman eks SMA 1 Yogya (*I Love You All*) yang telah memberikan dukungan baik moril maupun spirituil selama penulis menimba ilmu di ITS.
7. Teman-teman seangkatan terutama sesama penghuni LAB-AJK : Heri "Arjuno Solo", Hamsyi, Tom "SLAM", Dead "King of Fighter", Yushar "Pak YY", mbah Genthong, mbah Noer, Putut, PeWee, Paryono, Dudi, Jabier, Uppi yang kagak "keren", Yudhi P, Yance, Bilqis (*for the books*), Jimmy, Andy, Penyo(k) "Bukuku endy ?", Deni, Arthur "mrninja", Riza, Lambang, Bang Ucok, Iim, Ika, Shanty, Yetty, Maya, dan temen-temen lain yang penulis kenal.
8. Teman-teman penghuni GL-51 : Bronnie, Rio "Badak", Erry "Mammoe", Agung "Bison", Ezmir "Laga-disc", Ajrun "Tobil", Said "GAM", Arif "Juve", Ronny "Rong", Tiko, Wahyu & Wendy, atas kebersamaan dan dukungannya hingga penulis dapat seperti sekarang ini.
9. Teman-teman di ITS : Natsir (*for your cool stuff*), Indra, Zeta, Luthfi "Item", Kotrix, Mbes, Bobby "System Shock", Agatha, serta teman-teman sesama ITS lain yang tidak dapat penulis sebutkan satu per satu.

Sebagai penutup penulis mengharapkan tugas akhir ini dapat memberikan manfaat bagi rekan-rekan mahasiswa pada umumnya serta komunitas Grafika Komputer maupun Computing pada khususnya. Penulis juga mengharapkan agar tugas akhir ini dapat digunakan sebagai ide dasar untuk dapat dikembangkan lebih lanjut.

Surabaya, Januari 2000

Joko Wijoseno

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Deteksi dan klasifikasi citra dengan *deformable contour* dimaksudkan untuk melakukan pengenalan obyek dengan memanfaatkan bentuk kontur yang ada. Proses deteksi dilakukan untuk mengetahui keberadaan kontur sedangkan proses klasifikasi digunakan untuk mengelompokkan citra berdasarkan konturnya. Ada beberapa metode dalam mendeteksi dan mengklasifikasi citra menggunakan *deformable contour*, yang biasanya mengasumsikan bahwa kontur telah diekstraksi sebelumnya. Metode yang telah ada menyangkut deteksi dan klasifikasi citra dengan *deformable contour* langsung dari citra terdistorsi, sebagian besar menggunakan ide yang intuitif atau heuristik. Sebagai contoh pada proses pengenalan karakter hasil tulisan tangan digunakan kombinasi bobot (*weighted combination*) dari fungsi energi yang mengukur tingkat penyimpangan dari sebuah model yang ideal serta kemiripannya (*closeness of match*)[1]. Pada metoda ini tidak ada mekanisme formal untuk menentukan faktor bobot yang sangat signifikan dipengaruhi oleh unjuk kerja (*performance*). Contoh lain adalah pada proses pencocokan bentuk (*shape matching*) menggunakan jarak Hausdorff

[1] D.H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," Pattern Recognition, vol. 13, 1981, pp. 111-122.

yang memberikan toleransi dari kesalahan posisi yang kecil[2]. Metode ini sepertinya tidak berkutik karena adanya kemungkinan perubahan bentuk. Karena tidak ada pemodelan citra terdistorsi yang dilibatkan, algoritma ini harus menentukan secara heuristik bagian-bagian titik sudut untuk dibandingkan dengan model yang telah ada. Demikian juga pada pengenalan kurva yang menggunakan polinomial secara implisit dan biasanya mengacu pada aturan keputusan Bayes (*Bayes decision rule*)[3]. Tanpa mengesampingkan metode-metode yang telah ada, untuk mendapatkan kekuatan representasi yang cukup, biasanya dibutuhkan polinomial dengan derajat / pangkat yang tinggi. Akan tetapi dengan adanya gangguan (*noise*), koefisien-koefisien ini menjadi tidak stabil dan probabilitas gabungan mereka sulit dihitung.

Tugas akhir ini akan membahas strategi formal untuk mendeteksi dan mengklasifikasi citra dengan *deformable contour* langsung dari citra yang mengalami gangguan (*noisy images*). Formulasi yang dikembangkan didasarkan pada *generalized active contour model* atau *g-snake* yang melibatkan model deformasi (*deformation model*) dan model gangguan pada citra (*image noise model*). Tidak seperti *Fourier descriptor* atau *implicit polynomial*, representasi bentuk pada *g-snake* stabil. Sedikit deviasi pada bentuk (*shape*) akan mengakibatkan sedikit gangguan pada representasi bentuk (*shape representation*) ataupun sebaliknya. Lebih dari itu, representasi bentuk ini unik serta konstan

---

[2] D. P. Huttenlocher, G. A. Klanderman & W. J. Rucklidge, "Comparing Images Using the Hausdorff Distance," IEEE Trans. Pat. Anal. Mach. Intell., vol. PAMI-15, 1993, pp. 850 – 863.

[3] J. Subrahmonia, J. D. Karen & D. B. Cooper, "Bayesian Method for the use of Implicit Polynomials and Algebraic Invariants in Practical Computer Vision," Curves and Surfaces in Computer vision & Graphics 3, 1992, pp. 104-117.

(secara implisit) untuk memiliki relasi yang kuat dengan transformasi dan memiliki properti metrik (*metric properties*). Karakteristik ini membuat *g-snake* sangat cocok untuk tugas deteksi dan klasifikasi model *deformable contour*.

1.2 Maksud dan Tujuan

Manfaat dari perancangan dan pembuatan perangkat lunak ini adalah mempelajari dan memberikan pemikiran dasar dalam metode deteksi dan klasifikasi citra dengan memanfaatkan *deformable cotour* yang diimplementasikan menggunakan *g-snake*. Tujuan dari pembuatan perangkat lunak ini adalah untuk mengamati deteksi dan klasifikasi optimal pada citra dengan model *deformable contour* yang diimplementasikan menggunakan *g-snake*. Hasil dari deteksi dan klasifikasi ini berupa nilai skala kepercayaan (*confidence level*) dari beberapa model menurut *deformable contour* yang digunakan.

1.3 Permasalahan dan Batasan Masalah

Untuk mewujudkan hal tersebut di atas perlu dipecahkan beberapa permasalahan :

1. Representasi bentuk dengan menggunakan *deformable contour* yang telah diekstraksi ke dalam matriks bentuk (*shape matrix*).
2. Penentuan kontur kaku (*rigid contour*) dan kontur yang dapat berubah bentuk (*deformable contour*).

3. Penggabungan model kontur global dengan medan random Markov (*Markov random field*) untuk menghasilkan distribusi awal untuk setiap kontur yang ada untuk menentukan tingkat kecocokan pada tugas deteksi dan klasifikasi.
4. Pemodelan citra sebagai fungsi vektor.
5. Penggunaan model kontur dan model citra untuk mencari deteksi dan klasifikasi yang optimum.
6. Implementasi proses deteksi dan klasifikasi ke dalam perangkat lunak.

Dalam merancang dan membuat perangkat lunak ini ada beberapa batasan masalah yang perlu diperhatikan :

1. Citra yang digunakan dalam proses adalah citra *bitmap* dengan mode warna 24-bit.
2. Dalam pemrosesan citra digunakan citra yang menggunakan derajat tingkat keabuan (*graylevel image*) yang memiliki ukuran kurang dari 200 × 200 piksel.
3. Proses pemodelan dan ekstraksi *deformable contour* tidak termasuk dalam bahasan.
4. Model kontur yang digunakan dibatasi hanya pada bentuk benda yang tidak terlalu kompleks dan rumit.

1.4 Metodologi Penelitian

Metodologi penelitian yang digunakan dalam penyusunan Tugas Akhir ini adalah :

1. Studi Literatur.
2. Perancangan dan pembuatan sistem dan struktur data serta pengimplementasian kedalam perangkat lunak.
3. Proses pengumpulan data sampel.
4. Evaluasi dan revisi program.
5. Penulisan Naskah.

1.5 Sistematika Penulisan

Sistematika yang digunakan dalam tugas akhir ini dijelaskan berikut ini.

- Bab I Pendahuluan, menjelaskan tentang hal-hal yang mendasari serta memotivasi perancangan dan pembuatan perangkat lunak, maksud dan tujuan, permasalahan dan pembatasan masalah, serta sistematika penulisan.
- Bab II Pemrosesan Data Citra, membahas tentang proses–proses dasar yang digunakan dalam memanipulasi data citra antara lain proses *grayscaling* untuk mengabaikan informasi warna pada citra, proses *edge detection* untuk mendapatkan garis tepi citra, histogram ekualisasi dan juga proses korelasi citra yang digunakan dalam proses piramida citra.

- Bab III Pemodelan Kontur dan Citra, membahas tentang representasi bentuk, pemodelan kontur kaku dan kontur yang mampu berubah bentuk (*deformable contour*), pemodelan deformasi, karakteristik kontur dan pemodelan citra.
- Bab IV Deteksi dan Klasifikasi, membahas tentang strategi deteksi dan klasifikasi yang digunakan dalam tugas akhir ini.
- Bab V Perancangan dan Pembuatan Perangkat Lunak, menjelaskan tentang tahap perancangan, struktur data, obyek, dan implementasi pada perangkat lunak.
- Bab VI Uji Coba dan Evaluasi Perangkat Lunak, membahas hasil uji coba perangkat lunak yang sudah jadi dan menganalisa parameter-parameter yang digunakan dalam evaluasi hasil uji coba perangkat lunak.
- Bab VII Kesimpulan dan Saran, menguraikan kesimpulan dari bab-bab sebelumnya serta saran yang berkaitan dengan perancangan dan pembuatan perangkat lunak

BAB II

PENGOLAHAN DATA CITRA

BAB II

PENGOLAHAN DATA CITRA

2.1 Pendahuluan

Pengolahan data citra merupakan bagian yang tidak dapat dipisahkan dalam proses pembuatan model citra agar dapat dideteksi dan diklasifikasikan menurut konturnya dengan baik. Pengolahan data citra bertujuan untuk menambah, memperbaiki, dan meningkatkan kualitas citra sehingga kesalahan-kesalahan dalam proses deteksi dan klasifikasi citra dengan *deformable contour* dapat dikurangi. Selain itu, data citra yang digunakan dalam proses deteksi dan klasifikasi citra dengan *deformable contour* harus dapat memberikan interpretasi yang jelas sehingga hasil yang diinginkan sesuai dengan harapan.

Citra adalah representasi dua dimensi untuk bentuk fisik nyata tiga dimensi[4]. Citra terbentuk dari susunan piksel-piksel yang memiliki derajat warna tertentu yang disebut *palette*. *Palette* dapat terdiri dari kombinasi dua warna yaitu hitam dan putih yang bersama – sama membentuk *palette graylevel*. *Palette* dapat pula terdiri dari kombinasi tiga warna sekaligus yaitu merah, hijau, dan biru yang bersama – sama membentuk *palette* RGB (Red, Green, Blue). Ada pula *palette* yang terbentuk dari kombinasi empat warna sekaligus. *Palette* ini merupakan kombinasi dari warna sian (cyan), magenta, kuning serta hitam yang lebih dikenal

---

[4]Pratt, WK, "Digital Image Processing", John Wiley & Sons, New York, 1978.

dengan nama *palette* CMYK. Warna hitam disini berfungsi untuk mengatur intensitas gelap terang komposisi citra CMYK.

Citra yang dihasilkan dari proses pengambilan obyek nyata tiga dimensi ke dalam bentuk dua dimensi tentunya akan selalu mengalami penurunan kualitas gambar sehingga tidak sama persis dengan obyek tiga dimensinya. Hal ini disebabkan oleh berbagai hal antara lain :

- Noise atau pun gangguan

  - disebabkan oleh peralatan elektronik yang digunakan sehingga terjadi penyimpangan dari obyek aslinya.

- Penurunan resolusi

  - disebabkan oleh keterbatasan media tayang (monitor) serta media cetak (printer ataupun fotografi) sehingga akan terlihat agak kasar setelah mengalami perbesaran.

- Degradasi warna

  - disebabkan oleh keterbatasan *palette* untuk menampilkan obyek *true color*. Meskipun sekarang telah muncul teknologi warna 32-bit yang mampu menyamai *true color*, akan tetapi ini hanyalah pendekatan dari warna–warna yang ada yang sebenarnya memiliki kombinasi warna yang tidak terbatas. Bisa juga disebabkan oleh keadaan pada saat pengambilan obyek (terlalu gelap maupun terlalu terang) sehingga akan mengakibatkan deviasi warna bila dibandingkan dengan obyek aslinya.

Pada pengolahan citra dengan memanfaatkan teknologi komputer, terlebih dahulu dilakukan proses transformasi citra ke dalam bentuk data digital yang disebut proses digitalisasi. Selanjutnya data tersebut dapat disimpan kedalam media penyimpanan pada komputer seperti harddisk, compact disk ataupun floppy disk.

Citra dirumuskan sebagai suatu fungsi intensitas cahaya dua dimensi, yakni $f(x,y)$, yang menyatakan nilai intensitas (kecerahan) dari citra pada koordinat (x,y). Pemetaan citra dalam koordinat dua dimensi (x,y) disebut *image sampling*. Citra yang terbentuk di layar monitor menempati koordinat ruang dua dimensi dengan elemen – elemen warna dalam bentuk diskrit. Koordinat (x,y) yang membentuk susunan citra ini disebut piksel.

Persamaan berikut ini menunjukkan fungsi $f(x,y)$ pada suatu citra berukuran $M \times N$, dimana setiap elemen di dalam array memiliki bentuk nilai diskrit. M menunjukkan panjang citra dan N menunjukkan lebar citra. Setiap elemen yang terdapat di dalam array menunjukkan image elemen atau yang biasa disebut dengan piksel.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \cdots & f(1,N-1) \\ f(2,0) & f(2,1) & f(2,2) & \cdots & f(2,N-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.1 Gambar Citra dalam ruang dua dimensi

2.2 Proses Grayscaling

Proses *grayscaling* ini berfungsi untuk mengabaikan informasi warna dari citra yang akan dideteksi dan diklasifikasikan konturnya. Proses ini diperlukan karena input citra yang diinginkan dalam proses klasifikasi ini adalah citra yang memiliki tingkat warna derajat keabuan (*grayscale*). Hal ini disebabkan oleh keterbatasan algoritma yang ada sehingga hanya mampu menganalisa citra dengan warna derajat keabuan.

Cara untuk mengubah citra yang memiliki tingkat warna RGB ke dalam *grayscale* adalah dengan menjumlah seluruh komponen warna yang ada pada citra RGB. Kemudian hasil dari penjumlahan tersebut dibagi dengan tiga. Hal ini dikarenakan citra RGB terbentuk oleh tiga macam komponen warna. Secara matematis konversi warna RGB ke dalam *grayscale* pada citra dua dimensi dapat dituliskan sebagai berikut :

$$f(x,y)_r = f(x,y)_g = f(x,y)_b = \frac{f(x,y)_r + f(x,y)_g + f(x,y)_b}{3} \qquad (2.1)$$

dimana $f(x,y)_r$ adalah tingkat intensitas warna merah pada piksel (x,y). $f(x,y)_g$ adalah tingkat intensitas warna hijau pada piksel (x,y). $f(x,y)_b$ adalah tingkat intensitas warna biru pada piksel (x,y).

2.3 Deteksi Garis Tepi (*Edge Detection*)[5]

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

Pada umumnya, ide yang mendasari kebanyakan aplikasi mengenai deteksi garis tepi adalah dengan perhitungan dari operator *local derivative*. Hal ini diilustrasikan dalam gambar 2.2. Pada gambar 2.2(a) diperlihatkan citra dengan jalur terang pada latar belakang yang gelap. Gambar ini memperlihatkan bahwa derivatif pertama dari profil tingkat keabuan adalah positif pada transisi garis tepi awal, negatif pada garis tepi berikutnya, dan nol untuk daerah yang memiliki tingkat keabuan sama.

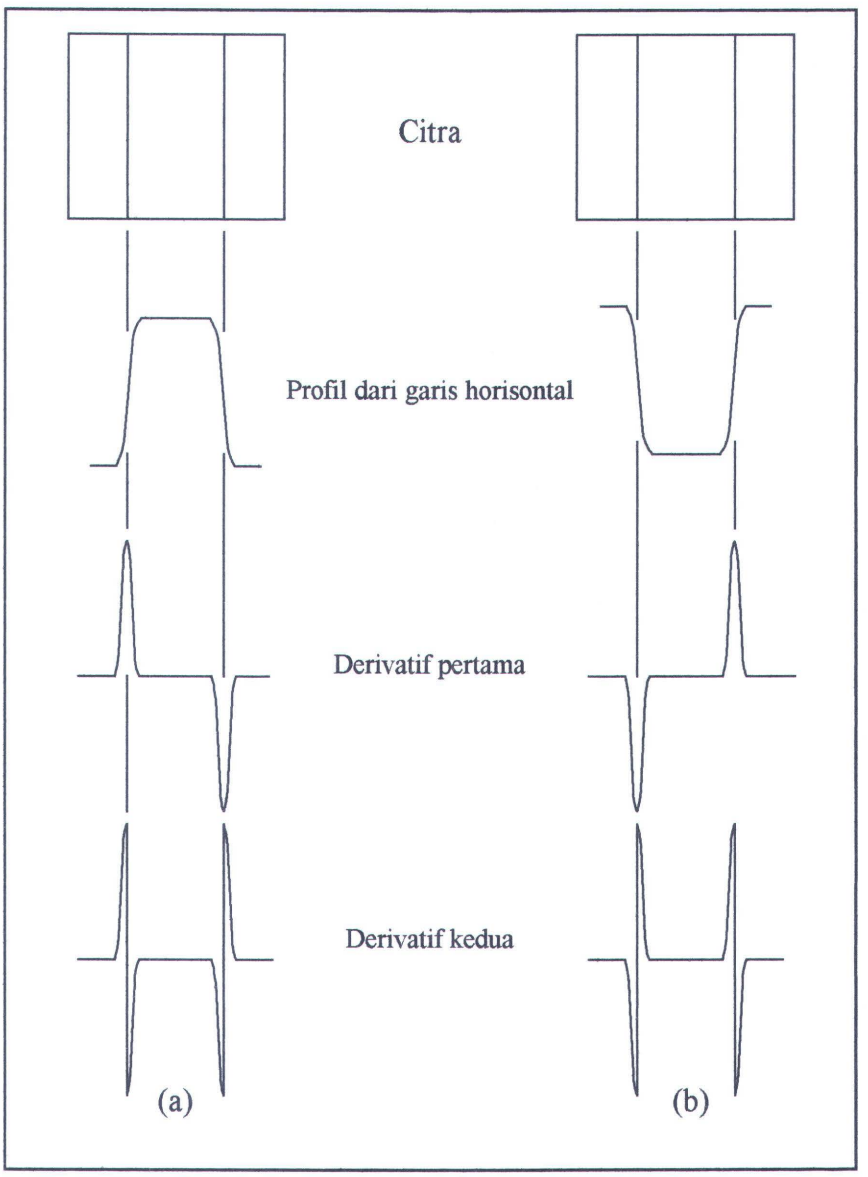
Derivatif kedua bernilai positif untuk bagian transisi yang berhubungan dengan bagian gelap dari garis tepi, negatif untuk bagian transisi yang berhubungan dengan bagian terang dari garis tepi, dan juga nol untuk daerah yang memiliki tingkat keabuan sama.

Derivatif 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 :

---

[5] Rafael C. Gonzales & Richard E. Woods, "Digital Image Processing," *Addison-Wesley*, 1983, pp 416-419.

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.2}$$



Gambar 2.2 Deteksi tepi dengan operator derivatif : (a) lajur terang pada latar belakang gelap; (b) lajur gelap pada latar belakang terang. Derivatif kedua mempunyai persilangan nol (*zero crossing*) pada tiap lokasi garis tepi.
[Rafael C. Gonzales & Richard E. Woods, "Digital Image Processing,"*Addison-Wesley*, 1983, pp. 417]

Vektor intensitas gradien dihitung dengan metode *least squares*, misalnya untuk bagian dari citra dengan dimensi 2×2 seperti pada gambar 2.3 didapatkan :

f_{11}	f_{12}
f_{21}	f_{22}

Gambar 2.3 Contoh bagian citra $f_{x,y}$ pada lokasi (x,y)

$$G_x = \frac{\partial f}{\partial x} = \frac{1}{2}\{(f_{12} + f_{22}) - (f_{11} + f_{21})\} \tag{2.3}$$

$$G_y = \frac{\partial f}{\partial y} = \frac{1}{2}\{(f_{21} + f_{22}) - (f_{11} + f_{12})\} \tag{2.4}$$

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 yang secara umum disebut gradien (*gradient*) atau ∇f . Sehingga intensitas gradien atau ∇f untuk gambar 2.3 pada lokasi f_{11} diberikan oleh persamaan,

$$\nabla f = mag(f) = \left[G_x^2 - G_y^2\right]^{1/2} \tag{2.5}$$

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

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_x}{G_y}\right) \tag{2.6}$$

dimana sudut tersebut dihitung relatif terhadap sumbu x.

Dari penjelasan di atas didapatkan dua buah hasil dari proses deteksi garis tepi, yaitu : *magnitude* untuk gradien intensitas pada citra, dan arah sudut dari gradien tersebut. Tingkat intensitas dari garis tepi yang diinginkan dapat diatur melalui proses ekualisasi histogram.

2.4 Ekualisasi Histogram (*Histogram Equalization*)

Proses ekualisasi histrogram bertujuan untuk membagi dan menyamaratakan jumlah warna yang lebih dominan ke dalam warna lain yang kurang dominan. Akibatnya, detil-detil warna yang semula tak tampak akan lebih jelas terlihat. Proses ini hanya digunakan pada citra dengan tingkat keabuan (*graylevel image*). Proses histogram pada citra digital dengan tingkat keabuan [0,L-1] merupakan sebuah fungsi diskrit, dengan L merupakan jumlah derajat keabuan :

$$p(r_k) = \frac{n_k}{n} \tag{2.7}$$

dimana k = 0,1,2, ... , L-1. r_k menyatakan tingkat keabuan k, n_k menyatakan jumlah piksel pada citra dengan tingkat keabuan k, dan n menyatakan total jumlah

piksel pada citra. Variabel n_k ditentukan dengan menghitung jumlah piksel untuk tiap tingkat k pada citra.

Langkah pertama yang dilakukan dalam proses pengkondisian ini adalah menghitung jumlah piksel untuk tiap tingkat keabuan (n_k). Setelah jumlah piksel untuk tingkat keabuan diketahui, nilai kumulatif untuk tiap tingkat keabuan ditentukan melalui fungsi distribusi kumulatif,

$$C_k = \frac{1}{n}\sum_{i=0}^{k} n_i \tag{2.8}$$

Variable C_k merupakan fungsi distribusi kumulatif dari histogram. Setelah itu, tiap fungsi distribusi kumulatif ditransformasikan dengan bentuk transformasi tertentu untuk mendapatkan pemetaan yang baru dengan tingkat keabuan yang telah ditetapkan,

$$S_{rk} = T(k) \tag{2.9}$$

dimana S_{rk} menyatakan daerah pemetaan dengan tingkat keabuan k setelah proses transformasi[6]. Untuk mengkondisikan hasil yang ingin dicapai melalui proses histogram, ada beberapa variabel yang dapat dimodifikasi. Pengkondisian citra berdasarkan proses histogram ini dispesifikasikan dalam bentuk transformasi

---

[6] Rafael C. Gonzales & Richard E. Woods, "Digital Image Processing," *Addison-Wesley*, 1983, pp 173-178.

berikut ini :

$$T(k) = \begin{cases} \dfrac{C_k low\_val}{low\_pct} & ;C_k < low\_val \\ (high\_val - low\_val)\left(\dfrac{C_k - low\_pct}{high\_pct - low\_pct}\right)^{imm\_pow} + low\_val & ;low\_val < C_k < high\_val \\ (1 - high\_val)\left(\dfrac{C_k - high\_pct}{1 - high\_pct}\right)^{imm\_pow} + high\_val & ;C_k < high\_val \end{cases} \quad (2.10)$$

dimana *low_pct* dan *high_pct* menyatakan jarak persentase bagi piksel pada citra untuk dipetakan, *low_val,* dan *high_val* menyatakan jarak intensitas untuk pemetaan piksel, dan *imm_pow* menyatakan pangkat eksponensial yang digunakan dalam fungsi transformasi. Untuk proses ekualisasi histogram secara umum dapat dicapai dengan mengatur nilai variabel untuk *low_pct*=0, *high_pct*=1, *low_val*=0, *high_val*=1, dan *imm_pow* = 1.

Dari hasil transformasi didapatkan pemetaan tingkat keabuan yang baru, dengan jumlah piksel yang merata untuk tiap tingkat tergantung pada variabel yang telah ditentukan sebelumnya. Jika $I_{x,y}$ menyatakan tingkat keabuan piksel pada lokasi (x,y), maka nilai pemetaan ditentukan melalui persamaan $I_{x,y} = S_{r(Ix,y)}$ yang merupakan fungsi pemetaan terhadap nilai tingkat keabuan baru yang dipetakan terhadap nilai sebelum proses ekualisasi histogram.

2.5 Korelasi Citra (*Image Correlation*)

Korelasi citra dalam pemrosesan data citra ini bertujuan untuk menghasilkan korelasi antar piksel citra. Hal ini dilakukan dengan menghitung hasil perkalian dalam (*inner product*) antara data citra dengan filter untuk citra. Korelasi citra di sini digunakan filter yang berfungsi sebagai media penghalus citra (*image smoothing*) yaitu berupa matriks berdimensi 3 × 3. Matriks ini merupakan matriks kondisi yang bersifat *Gaussian* atau dikenal dengan istilah *Gaussian kernel*[7]. Nilai konstan yang terdapat pada matriks tersebut yaitu :

0.122466	0.049997	0.049997
0.299975	0.122466	0.122466
0.122466	0.049997	0.049997

Tabel 2.1 *Gaussian generating kernel*

Dalam melakukan proses korelasi, ada beberapa batasan yang digunakan yaitu jumlah langkah kolom dan baris untuk tiap piksel terhadap citra *template*, didefinisikan sebagai $Step_{col}$ dan $Step_{row}$. Misalkan luas citra didefinisikan sebagai L, maka hasil dari proses korelasi berupa citra dengan luas,

$$L' = L / (Step_{col} \times Step_{row}) \qquad (2.11)$$

---

[7]A. K. Jain & E. Angel, "Fundamentals of Digital Image Processing," IEEE *Trans. Computer*, pp. 470-476.

Hasil perkalian dalam (*inner product*) didefinisikan dengan persamaan berikut,

$$R_{m,n} = \sum_{x,y=0}^{n_x,n_y} T_{x,y} \times \left((Step_{col}, Step_{row}) + (x,y) - \frac{T_{x,y}}{2} \right) \quad (2.12)$$

Dimana $R_{m,n}$ menyatakan hasil perkalian dalam untuk lokasi piksel (m,n) pada citra, n_x dan n_y masing-masing menyatakan jumlah baris dan kolom pada citra *template*, dan $T_{x,y}$ merupakan nilai tingkat keabuan pada piksel (x,y) pada citra *template*.

Proses korelasi data citra dengan citra *Gaussian* akan menghasilkan efek penghalusan pada citra tersebut. Proses ini digunakan untuk mengurangi adanya distorsi pada citra, sehingga pada saat proses deteksi tepi (*edge mapping*), hasil yang didapatkan lebih jelas sesuai dengan yang diharapkan.

2.6 Piramida Citra (*Image Pyramid*)

Piramida merupakan susunan dari sekelompok citra yang telah mengalami proses korelasi bertingkat terhadap *Gaussian kernel*. Dengan kata lain, piramida citra tersusun dari citra–citra yang telah diperhalus dengan proses korelasi.

Proses pembuatan piramida berdasarkan proses yang telah dijelaskan sebelumnya, yaitu korelasi citra dengan *Gaussian kernel*. Disebut dengan piramida karena ukuran citra pada tingkat yang lebih tinggi adalah seperempat dari ukuran citra pada tingkat sebelumnya. Hal ini disebabkan karena proses korelasi terhadap citra *template* berupa *Gaussian kernel* dalam persamaan (2.12)

mempunyai nilai $Step_{col} = Step_{row} = 2$, sehingga menurut persamaan (2.11) membagi luas citra menjadi empat. Tujuan pembuatan piramida citra, yaitu untuk mengurangi distorsi yang terdapat pada citra dengan memperkecil ukuran citra serta memperhalus citra.

BAB III

PEMODELAN KONTUR DAN CITRA

BAB III

PEMODELAN KONTUR DAN CITRA

3.1 Representasi Bentuk (*Representing Shape*)

Kontur adalah sebuah vektor yang mengandung sekumpulan titik-titik yang berurutan (*ordered set of points*), $V = [\, v_1, v_2, \ldots, v_n \,]$. Masing-masing v_i didefinisikan pada ruang vektor terbatas (*finite grid*): $v \in \mathrm{IE} = \{(x,y) : x,y = 1, 2, \ldots, \mathrm{M}\}$, sehingga $V \in \mathrm{IE}^n$.

Diberikan $U \in \mathrm{IE}^n$ dimana masing-masing $u_i = v_i - g$ merepresentasikan jarak atau perpindahan v_i dari sembarang titik referensi g. Sehingga sebuah bentuk (*shape*) dari sembarang kontur dapat diartikan dalam persamaan bentuk (*shape equation*) seperti dibawah ini,

$$AU^T = 0 \tag{3.1}$$

dimana

$$A = \begin{bmatrix} 1 & -\beta_1 & -\alpha_1 & 0 & \ldots & 0 \\ -\alpha_2 & 1 & -\beta_2 & 0 & 0 & \ldots \\ 0 & -\alpha_3 & 1 & -\beta_3 & 0 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & -\alpha_{n-1} & 1 & -\beta_{n-1} \\ 0 & 0 & \ldots & -\beta_n & -\alpha_n & 1 \end{bmatrix}$$

$$U = [v_1 - g \quad v_2 - g \quad \cdots \quad v_n - g]$$

sehingga,

$$U^T = \begin{bmatrix} v_1 - g \\ v_2 - g \\ \vdots \\ v_n - g \end{bmatrix}$$

Matriks A adalah sebuah matrik bentuk (*shape matrix*) $n \times n$ yang bersifat regeneratif atau dengan kata lain kontur U dapat dihasilkan dari matriks A dengan inversi yang tepat jika sembarang dua titik pada U diketahui. Hal ini dimungkinkan oleh karena rank(A) = n – 2, dan begitu pula dengan (n – 2) × (n – 2) submatriks yang invertibel, tetap berada dalam matriks A. Untuk menunjukkan hal ini, tanpa mengurangi karakteristik matriks secara umum :

$$A_\gamma U_\gamma + \mathrm{b} = 0 \tag{3.2}$$

dengan $A_\gamma = \{A\}_{(2:n-1) \times (1:n-2)}$ adalah submatriks yang invertibel, $U_\gamma = \{U\}_{1:n-2}$ merupakan bagian yang tidak diketahui dari U, dan

$$\mathrm{b} = \begin{bmatrix} \cdots \\ 0 \\ \cdots \\ -\beta_{n-2} u_{n-1} \\ u_{n-1} - \beta_{n-1} u_n \end{bmatrix} \tag{3.3}$$

sehingga

$$U_\gamma = -A_\gamma^{-1} \mathrm{b} \tag{3.4}$$

Di lain pihak, jika U yang kompleks diketahui, koefisien bentuk (*shape coefficients*) dapat dihitung sebagai berikut :

$$\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} x_{i_\alpha} & x_{i_\beta} \\ y_{i_\alpha} & y_{i_\beta} \end{bmatrix}^{-1} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3.5}$$

Hal ini menunjukkan bahwa matriks A adalah stabil. Jika terjadi perubahan pada kontur akan mempengaruhi koefisien dari matriks bentuk, demikian pula sebaliknya. Adanya perubahan pada variabel x dan y menyebabkan terjadinya perubahan pada variabel α dan β seperti telah ditunjukkan dalam persamaan (3.5).

Dapat dilihat bahwa persamaan bentuk (*shape equation*) adalah invarian dan unik untuk mengikat transformasi pada V. Dengan kata lain, untuk nilai A tertentu, jika $\overline{U}$ (vektor basis dari U) memenuhi $A\overline{U}^T = 0$, maka

$$AU^T = 0 \Leftrightarrow U = T\overline{U} \tag{3.6}$$

dimana T adalah sebuah matrik transformasi 2 × 2 yang invertible yang dapat menjabarkan gerakan kaku (*rigid motions*) seperti penskalaan (*scaling*), rotasi, peregangan (*stretching*) dan dilasi (*dilation*). Lebih jauh, perubahan tempat pada titik pusat gravitasi (*center of gravity*), $\overline{g} + d$ (*distance*) tidak memiliki efek pada

persamaan bentuk (*shape equation*). Pada diskusi ini dapat diasumsikan bahwa T dan d adalah diketahui, atau dapat diperkirakan tingkat kepercayaannya menggunakan *generalized Hough transforms*. Dengan kata lain dapat diasumsikan pengetahuan tentang kontur referensi (*reference contour*) $\overline{U}$ dan $\overline{g}$.

3.2 Kontur Kaku dan Kontur yang Bisa Berubah Bentuk (*Rigid and Deformable Contour*)

Sekumpulan keluarga kontur (*family of contours*) $U \in \Omega$ (himpunan vektor basis) $\subseteq IE^n$ dapat dikatakan kontur kaku (*rigid contour*) bila

$$p(U) = \begin{cases} 1; & U = \overline{U} \\ 0; & \textit{lainnya} \end{cases} \qquad (3.7)$$

dengan kata lain, $\Omega = \{\overline{U}\}$, dimana $p(U)$ melambangkan probabilitas kontur.

Sebaliknya, sekumpulan keluarga kontur (*family of contours*) $U \in \Omega$ dapat dikatakan dapat berubah bentuk (*deformable*) bila

$$p(U) > 0, \text{ jika terdapat suatu } U \neq \overline{U} \qquad (3.8)$$

Bagian berikut ini menunjukkan model deformasi yang digunakan untuk mendapatkan distribusi $p(U)$ dalam persamaan (3.8).

3.3 Pemodelan Deformasi (*Modeling Deformation*)

Misalkan beberapa kontur $U \in \Omega$ memiliki kemiripan tetapi menunjukkan ketidakberaturan bentuk yang kecil (*small shape irregularities*). Untuk merepresentasikan fluktuasi yang tidak beraturan ini dalam lokasi kecil, didefinisikan sebuah energi internal (*internal energy*) yang diambil dari matriks bentuk (*shape matrix*) A,

$$E_{\text{int}}(U) = \frac{(AU^T)^T R^{-1}(AU^T)}{l(U)} \tag{3.9}$$

dimana $l(U) = \sum_{i=1}^{n} \left\| u_{i+1} - u_i \right\|^2$ adalah sebuah konstanta normalisasi, dan R = diag $\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2\}$ mengandung varian deformasi-varian deformasi σ_i^2 yang mengijinkan penyerahan dari *location weightings* pada deformasi. E_{int} dapat diuraikan kedalam n bagian :

$$E_{\text{int}}(U) = \sum_{i=1}^{n} \frac{E_{\text{int}}(u_i)}{\sigma_i^2} \tag{3.10}$$

dimana $E_{\text{int}}(u_i) = \frac{1}{l(U)} \left\| u_i - \alpha_i u_{i_\alpha} - \beta_i u_{i_\beta} \right\|^2$ dan sebagai penunjuk basis diberikan,

$$i_\alpha = \begin{cases} i-1; & i > 1 \\ 3; & i = 1 \end{cases} \qquad i_\beta = \begin{cases} i+1; & i < n \\ n-2; & i = n \end{cases} \tag{3.11}$$

Probabilitas pada U ($p(U)$), dapat diberikan menggunakan fungsi energi,

$$p(U) = \frac{1}{Z}\exp(-E_{\text{int}}(U)) \tag{3.12}$$

dimana $Z = \sum_{U\in\Omega}\exp(-E_{\text{int}}(U))$ sebagai konstanta normalisasi. Sebuah ukuran probabilitas dari bentuk (*form*) pada persamaan (3.12) dinamakan *Gibbs measure*. Sebagai referensi, hal ini juga didefinisikan pada Medan Random Markov (*Markov random field*)[8], sebagai contoh,

$$p(u_i|u_1,u_2,\ldots,u_n) = p(u_i|u_{i_\alpha}u_{i_\beta}) \tag{3.13}$$

dimana probabilitas kondisional (*conditional probability*) dari u_i yang diberikan oleh seluruh rantai U dapat secara menyeluruh dispesifikasikan oleh probabilitas kondisional dari u_i yang diberikan oleh dua titik basis (*basis point*).

Sebagai kesimpulan, sebuah model kontur global dapat dikombinasikan dengan Medan Random Markov (*Markov random field*) menjadi distribusi yang telah disebutkan di atas untuk sembarang kontur. Berikut, akan diuraikan bermacam-macam karakteristik dari formulasi ini untuk menentukan kecocokannya untuk tugas deteksi dan klasifikasi.

---

[8] R. Kindermann & J. L. Snell, *Markov Random Fields and their Applications*, Americans Mathematical Society, 1980.

3.4 Karakteristik Model Kontur (Characteristics of Contour Models)

Model kontur yang telah didiskusikan di atas, memiliki beberapa karakteristik yang penting dan layak untuk tugas deteksi dan klasifikasi. Berikut ini akan dijabarkan lebih lanjut mengenai karakteristik-karakteristik yang ada pada model kontur.

Pertama, matrik bentuk (*shape matrix*) bersifat invarian di bawah transformasi yang mengikat, yang sangat penting dalam mendeteksi maupun mengklasifikasi obyek yang mengalami pergerakan kaku (*rigid motions*) dalam ruang tiga dimensi (*3-D space*). Lebih jauh, *invariance* yang mengikat dari matrik bentuk (*shape matrix*) bersifat implisit, tidak seperti *Fourier descriptor*[9] atau *memorized template*[10] yang membutuhkan perbandingan menyeluruh (*exhaustive comparison*) kedalam subruang (*subspace*) yang dibentuk.

Kedua, matrik bentuk (*shape matrix*) bersifat unik. Karena dua buah kontur tidak dapat direlasikan oleh sebuah transformasi yang mengikat, tidak dapat mempunyai matrik bentuk (*shape matrix*) yang sama. Pengenalan (*recognition*) tidak mungkin tanpa karakteristik ini.

Terakhir, energi internal E_{int} memiliki properti metrik (*metric properties*) yang mana berhubungan dengan pergerakan intuitif (*intuitive motion*) pada

---

[9] L. H. Staib & J. Duncan, "Boundary Finding with Parametrically Deformable Models," *IEEE Trans. Pat. Anal. Mach. Intell,vol.*PAMI-14, 1992, pp. 430-449.

[10] G. E. Hinton, C.K.I. Williams & M. D. Revow, "Adaptive Elastic Models for Hand-Printed Character Recognition," *Neural Information Processing Systems,* vol 4, ed. J. E. Moody *et. al.,* 1992, pp. 512-519.
D. P. Huttenlocher, G. A. Klanderman & W. J. Rucklidge, "Comparing Images Using the Hausdorff Distance," IEEE Trans. Pat. Anal. Mach. Intell., vol. PAMI-15, 1993, pp. 850-863.

persamaan bentuk. Misalkan $\overline{U}$ mewakili sebuah kontur referensi (*reference contour*) dimana $A\overline{U}^T = 0$, maka :

- $E_{int}(U) \geq 0$, $E_{int}(U) = 0$ jika dan hanya jika $U = T\overline{U}$ untuk beberapa T. Sehingga sebuah kontur hanya sama dengan dirinya sendiri atau transformasi yang mengikatnya.
- $\left|E_{\text{int}}(U_p) - E_{\text{int}}(U_q)\right| = \left|E_{\text{int}}(U_q) - E_{\text{int}}(U_p)\right|$. Urutan perbandingan dari dua buah kontur tidak berpengaruh.
- $\left|E_{\text{int}}(U_p) - E_{\text{int}}(U_q)\right| \leq \left|E_{\text{int}}(U_p) - E_{\text{int}}(U_r)\right| + \left|E_{\text{int}}(U_r) - E_{\text{int}}(U_q)\right|$. Pertidaksamaan triangular menunjukkan bahwa dua buah kontur yang sangat tidak sama tidak dapat keduanya sama dengan kontur ketiga.

Sebuah kontur, baik kaku maupun mampu berubah bentuk (*deformable*), biasanya berhubungan dengan titik tepi yang kuat (*strong edge point*) atau garis dengan intensitas tinggi (*high intensity lines*) dalam sebuah citra. Bagaimanapun juga, titik-titik tepi maupun intensitas ini mungkin pula timbul diakibatkan oleh adanya gangguan (*noise*). Pada bagian berikut ini akan dibahas mengenai model citra yang mengandung kontur yang ditambah dengan *Gaussian white noise*.

3.5 Model Citra (*Image Model*)

Didefinisikan sebuah citra sebagai fungsi dari vektor **F** : IE $\rightarrow$ ID. Menurut tipe datanya, ada dua jenis ID yaitu ID = IR (citra intensitas atau *edge magnitude*) dan ID = IR^2 (2 $\times$ 1 *intensity gradient vector*).

Sebuah *template* dari kontur adalah citra F = $B_{U,g}$:

$$B_{U,g}(r) = \sum_{i=1}^{n} h_i \delta(r - u_i - g) \tag{3.14}$$

dimana r = (x,y) ∈ IE, δ adalah fungsi delta, h_i ∈ ID dan $|h_i|$ = 1. Dengan kata lain, sebuah *template* $B_{U,g}(r)$ adalah citra khusus dengan nilai sama dengan h_i jika $r = u_i + g$ dan akan bernilai nol bila sebaliknya. Untuk ID = IR^2, h_i merupakan unit vektor yang bersifat normal pada kontur.

Sebuah citra terdistorsi (*noisy image*) **F** = f mengandung sebuah kontur dapat dimodelkan sebagai berikut,

$$\mathrm{f}(r) \mid U,g = B_{U,g}(r) + \eta(r) \tag{3.15}$$

dimana $\eta(r)$: $N(0, \sigma_\eta^2 \mathbf{I})$ bersifat *Gaussian*, **I** = 1 jika ID = IR; dengan kata lain **I** adalah matriks identitas 2 × 2. Konsekuensinya, $p(\mathrm{f} \mid U)$ berdistribusi Gaussian dan dapat disederhanakan menjadi,

$$p(\mathrm{f} \mid U) = \frac{1}{C} \exp\left\{ \frac{-f^T f + n - \sum_{i=1}^{n} E_{ext}(U_i)}{2\sigma_\eta^2} \right\} \tag{3.16}$$

dimana $C = (2\pi\sigma_\eta^2)^{\frac{M^2}{2}}$,dan energi eksternal $E_{ext}(U_i)$ = 1 - $h_i^T f(Ui)$.

Kemudian, model kontur dan citra yang telah dijabarkan diatas digunakan untuk menurunkan deteksi optimum dan tes klasifikasi. Akan dilihat bagaimana tes – tes ini dapat diimplementasikan dalam aplikasi praktis.

BAB IV

DETEKSI DAN KLARIFIKASI

BAB IV

DETEKSI DAN KLASIFIKASI

Deteksi dan klasifikasi merupakan proses terpenting yang menentukan keberhasilan *deformable contour* dalam tugas pengenalan obyek. Proses ini mengolah citra dan kontur yang telah dijabarkan pada bab-bab terdahulu agar dapat diklasifikasikan menurut klasifikasi tertentu. Proses deteksi dan klasifikasi ini akan dibahas lebih lanjut dalam bab ini.

4.1 Penjelasan Awal

Misalkan ada dua kandidat model ω_0 dan ω_1, yang menghasilkan citra $\mathbf{F} = \mathrm{f}$ yang masing-masing terdistribusi menurut fungsi densitas (*density function*) $p_0(\mathrm{f})$ dan $p_1(\mathrm{f})$. Masing-masing model ini dapat didefinisikan menggunakan hipotesa H_0 dan H_1 seperti berikut:

$$\begin{aligned} H_0 &: \mathrm{f} : p_0(\mathrm{f}) \\ H_1 &: \mathrm{f} : p_1(\mathrm{f}) \end{aligned} \qquad (4.1)$$

Asumsi yang akan digunakan adalah bahwa perbedaan antara ω_0 dan ω_1 berada pada distribusi konturnya (*contour distribution*) dan bahwa $\mathrm{f}^T\mathrm{f}$ konstan.

Sehingga probabilitas $p_i(f)$ dapat dihitung dengan memarginalisasi distribusinya yaitu,

$$p_i(f) = \sum_{U \in \Omega_i} p_i(U) p(f|U) \tag{4.2}$$

Jika ω_i berupa kontur kaku yaitu $\Omega_i = \{\overline{U}_i\}$ maka persamaan 4.2 berubah menjadi bentuk yang lebih sederhana yaitu :

$$p_i(f) = p(f|\overline{U}_i) \tag{4.3}$$

4.2 Deteksi (*Detection*)[10]

Dari citra f yang telah diamati, permasalahan deteksi dapat dituangkan ke dalam test *null hypothesis* H_0 dibandingkan dengan H_1. ω_0 dapat dikatakan *null model* jika $p_0(f)$ *uniform* untuk seluruh citra f. Dengan kata lain, *null model* tersebut tidak menghasilkan kontur dengan struktur yang koheren.

Biasanya, nilai probabilitas kesalahan (*probability of false alarm*) P_{fa}, dan probabilitas kegagalan (*probability of miss*) P_{miss} ditekan sekecil mungkin. Tetapi dalam banyak kasus, P_{fa} dan P_{miss} tidak dapat diminimalkan secara simultan. Sehingga dicari sebuah test yang akan meminimalkan P_{miss} , atau memaksimalkan

[10] K. F. Lai & R. T. Chin, "Deformable Contours: Detection and Classification," *IEEE Trans. Pat. Anal. Mach.Intell.*, 1994, pp. 69-70.

probabilitas deteksi (*probability of detection*), $P_D = 1 - P_{miss}$ untuk probabilitas kesalahan tertentu P_{fa}. Test tersebut dinamakan *the most powerful test*.

Didefinisikan rasio keserupaan (*likelihood ratio*) $L_{ij}(f) = p_i(f)/p_j(f)$. *Neyman-Pearson lemma* menyatakan sebuah test mengambil bentuk,

$$\text{pilih} \begin{cases} H_1 : jika \quad L_{10}(f) > k \\ H_0 : lainnya \end{cases} \tag{4.4}$$

untuk beberapa $k \geq 0$, adalah *the most powerful test* untuk $P_{fa} > 0$ yang diberikan.

Karena $p_0(f)$ *uniform* untuk *null models*, test deteksi menjadi

$$\text{pilih} \begin{cases} H_1 : jika \quad p_1(f) > k' \\ H_0 : lainnya \end{cases} \tag{4.5}$$

untuk beberapa $k' > 0$.

Sebuah test deteksi sederhana muncul jika ω_1 adalah model kontur kaku. Dengan menggunakan persamaan (4.3) dan (3.12), dipilih H_1 jika

$$\sum_{i=1}^{n} h_i^T f(\bar{u}_i) > k'' \tag{4.6}$$

untuk beberapa k" > 0. Dengan kata lain, kontur $\bar{U}$ terdeteksi jika ia menghasilkan korelasi yang cukup dengan citra yang diamati. Hal ini ekivalen

dengan *matched template operation* yang dapat diimplementasikan secara efisien menggunakan *generalized Hough transform*.

Jika ω_1 adalah sebuah model *deformable contour*, maka perlu untuk memarginalisasi distribusinya. Sehingga H_1 dipilih jika,

$$\sum_{U \in \Omega_i} p_i(U) p(f|U) > k' \tag{4.7}$$

untuk beberapa k' > 0.

4.3 Klasifikasi (*Classification*)[11]

Dalam klasifikasi biasanya tidak dibedakan antara P_{fa} dan P_{miss} , akan tetapi lebih pada mencari nilai terkecil dari probabilitas kesalahan total (*total probability of error*), $P_e = P_{fa} + P_{miss}$. Dengan menggunakan *Neyman-Pearson lemma*, sebuah test dalam bentuk persamaan (4.4) dengan $k = 1$ memenuhi persamaan tersebut. Dengan kata lain, kaidah Bayes dapat digunakan untuk menghasilkan

$$k = \frac{p_0 C_{01}}{p_1 C_{10}} \tag{4.8}$$

[11] K. F. Lai & R. T. Chin, "Deformable Contours: Detection and Classification," *IEEE Trans. Pat. Anal. Mach.Intell.*, 1994, pp. 70-72.

dimana p_i adalah probabilitas terdahulu model yang diamati ω_I dan C_{ij} adalah *cost* yang berhubungan dengan pemilihan H_j yang diberikan dimana H_i sedang berlaku. Tanpa kehilangan keberlakuan secara umum (*loss of generality*), $k = 1$ akan digunakan dalam pembahasan berikut.

Jika baik model ω_1 maupun ω_0 merupakan model kontur kaku maka dipilih H_1 jika ia menghasilkan korelasi yang lebih tinggi dengan citranya :

$$\sum_{i=1}^{n} h_i^T f(\bar{u}_{1,i}) > \sum_{i=1}^{n} h_i^T f(\bar{u}_{0,i}) \tag{4.9}$$

Secara umum, bagaimanapun juga sebuah test klasifikasi mungkin melibatkan satu atau lebih model deformasi sehingga memerlukan proses marginalisasi. Dalam kasus ini, dipilih H_1 jika

$$\sum_{U \in \Omega_1} p_1(U) p(f|U) > \sum_{U \in \Omega_0} p_0(U) p(f|U) \tag{4.10}$$

Tes klasifikasi dapat dengan mudah digeneralisasi menjadi permasalahan dari beberapa kelas (*class*) $\omega_1, \omega_2, \ldots, \omega_q$. Dalam kasus ini dipilih ω_γ jika

$$\gamma = \arg\max_{1 \le i \le q} \sum_{U \in \Omega_i} p_i(U) p(f|U) \tag{4.11}$$

Dapat dilihat dari bentuk persamaan (4.11), test yang melibatkan model deformasi lebih membutuhkan komputasi yang panjang tergantung kepada ukuran dari Ω_I. Pada kasus terburuk (*worst case*), sejumlah M^{2n} contoh dari $U \in IE^n$ pada ruang vektor terbatas (*finite grid*) masing-masing harus dikalkulasi seluruh kemungkinannya. Lebih jelasnya, beban komputasi ini menjadi tidak dapat dilakukan sekalipun untuk citra yang berukuran relatif kecil.

Beberapa permasalahan praktis dalam hal test deteksi dan klasifikasi ini akan dibahas pada bagian berikut ini. Pertama, akan dijabarkan karakteristik lokal dari medan random Markov (*Markov random field*) untuk mendesain algoritma penjumlahan (*summation algorithm*) dengan kompleksitas $O(nm^3)$, dimana n adalah jumlah titik dan m adalah ukuran daerah. Penjumlahan mengalami puncaknya (*peaked*) disekitar *posterior estimation* pada sebagian besar aplikasi sehingga waktu komputasi dapat direduksi dengan mengaplikasikan algoritma tersebut dalam daerah rektangular yang berukuran kecil (*small rectangular region*). Akhirnya dapat ditunjukkan bagaimana metoda *Monte-Carlo* dan *importance sampling* untuk mendapatkan konstanta normalisasi Z.

4.4 Hal-hal yang Perlu Diperhatikan dalam Implementasi Praktis (Issues in Practical Implementation)[12]

Banyak hal yang harus diperhatikan dalam pengimplementasian algoritma yang digunakan dalam test deteksi dan klasifikasi, agar hasil yang didapat sesuai

[12] K. F. Lai & R. T. Chin, "Deformable Contours: Detection and Classification," *IEEE Trans. Pat. Anal. Mach.Intell.*, 1994, pp. 72-75.

dengan yang diharapkan. Hal ini menyangkut beberapa algoritma yang digunakan dalam implementasi praktis seperti yang telah disebutkan pada pokok bahasan sebelumnya. Berikut ini akan dijabarkan mengenai perihal tersebut.

4.4.1 Algoritma Penjumlahan (*Summation Algorithm*)

Misalkan bahwa dalam Ω terdapat m_i lokasi yang mungkin untuk setiap u_i. Untuk menghitung secara keseluruhan

$$\sum_{U\in\Omega} p(U)p(f|U) = \sum_{u_1}\sum_{u_2}\cdots\sum_{u_n} p(U)p(f|U) \tag{4.12}$$

dibutuhkan masing-masing $\prod_{i=1}^{n} m_i$ komputasi. Beban komputasi ini bersifat faktorial dan dengan mudah mengarah kepada ledakan kombinatoris (*combinatoric explosion*).

Akan tetapi, karakteristik lokal dari medan random Markov (*Markov random field*) dapat digunakan untuk menghasilkan sebuah algoritma yang dapat menghitung dalam waktu polinomial (*polynomial time*). Untuk menyelesikannya, didefinisikan variabel statis c_i, dimana

$$c_i(u_{i_\alpha}, u_i, u_{i_\beta}) = \exp\left\{-\frac{E_{\text{int}}(u_i|u_{i_\alpha}u_{i_\beta})}{\sigma_i^2} - \frac{E_{ext}(u_i)}{\sigma_r^2}\right\} \tag{4.13}$$

dan mengamati bahwa

$$p(U)p(f|U) = \frac{\exp(-f^T f + n)}{ZC} \prod_{i=1}^{n} c_i(u_{i_\alpha}, u_i, u_{i_\beta}) \qquad (4.14)$$

Sehingga permasalahan ini dapat dipecah menjadi n-*stage proses* dan hanya memperhatikan tiga *snake point* pada tiap *stage*.

Dimulai dengan menghitung penjumlahan parsial $S_1(u_2,u_3)$:

$$S_1(u_2,u_3) = \sum_{u_1} c_1(u_3,u_1,u_2)c_2(u_1,u_2,u_3) \qquad (4.15)$$

Penjumlahan parsial berikutnya $S_2(u_3,u_4)$ didapatkan dari c_3 dan S_1 :

$$S_2(u_3,u_4) = \sum_{u_2} S_1(u_2,u_3)c_3(u_2,u_3,u_4)$$

$$= \sum_{u_2}\sum_{u_1} c_1(u_3,u_1,u_2)c_2(u_1,u_2,u_3)c_3(u_2,u_3,u_4) \qquad (4.16)$$

Dilanjutkan seperti bentuk diatas, dapat diakumulasikan penjumlahan parsial S_i untuk $i = 3$ sampai n-3 secara berturut-turut menggunakan c_{i+1} dan S_{i-1} :

$$S_i(u_{i+1},u_{i+2}) = \sum_{u_i} S_{i-1}(u_i,u_{i+1})c_i(u_i,u_{i+1},u_{i+2}) \qquad (4.17)$$

Akhirnya, pada *stage* n-2, dapat dikalkulasi :

$$S_{n-2}(u_{n-1},u_n)=\sum_{u_{n-2}} S_{n-3}(u_{n-2},u_{n-1})c_{n-1}(u_{n-1},u_n,u_{n-2})c_{n-2}(u_{n-2},u_{n-1},u_n) \tag{4.18}$$

Sekarang dapat dibuktikan bahwa

$$\begin{aligned} S_\Omega &= \sum_{U\in\Omega}\prod_{i=1}^{n} c_i(u_{i_\alpha},u_i,u_{i_\beta}) \\ &= \sum_{u_n}\sum_{u_{n-1}} S_{n-2}(u_{n-1},u_n) \end{aligned} \tag{4.19}$$

Oleh karena itu, dari persamaan (4.14) diperoleh

$$\sum_{U\in\Omega} p(U)p(f|U)=\frac{\exp(-f^T f+n)}{ZC}S_\Omega \tag{4.20}$$

dan ini menyempurnakan penjumlahan (*summation*).

Dari persamaan (4.17), terlihat bahwa di sana hanya membutuhkan m_i, m_{i+1}, m_{i+2} operasi untuk menghitung penjumlahan parsial S_i. Misalkan $m_i = m$ untuk semua nilai i, maka kompleksitas komputasi telah berhasil direduksi menjadi $O(nm^3)$.

4.4.2 Penjumlahan Terpuncak (*Peaked Summation*)

Penjumlahan dalam persamaan (4.2) mengalami puncaknya (*peaked*) di sekitar *posterior estimate* U_{map} pada sebagian besar aplikasi praktis, dimana

$$U_{map} = \max_{U \in \Omega} p(U)p(f|U) \tag{4.21}$$

merupakan solusi untuk minimisasi energi dalam *g-snake*. Dengan kata lain, sebagian besar dari sampel yang memberi kontribusi pada penjumlahan (*summation*), terkonsentrasi dalam daerah (*region*) kecil disekitar U_{map}, seperti terlihat pada gambar 4.1. Konsekuensinya, waktu komputasi dapat direduksi secara signifikan dengan memanfaatkan algoritma penjumlahan (*summation algorithm*) dalam daerah rektangular kecil (sebagai contoh 3 × 3 *window*) terpusat di sekitar U_{map}. Oleh karena itu solusi dari *g-snake* dapat diproses secara cepat pada bagian akhirnya, agar menghasilkan solusi untuk deteksi maupun klasifikasi.

Gambar 4.1 Daerah yang memberikan kontribusi secara signifikan pada algoritma penjumlahan (*summation algorithm*)

Gambar di atas memberikan gambaran bahwa beberapa titik atau daerah tertentu di sepanjang kontur telah mengalami penghilangan.

4.4.3 Konstanta Normalisasi (*Normalizing Constant*)

Dari persamaan (4.2) dan persamaan (4.20), fungsi keserupaan (*likehood function*) L_{ij}(f) dapat ditulis sebagai berikut :

$$L_{ij}(f) = \frac{p_i(f)}{p_j(f)} = \frac{S_{\Omega_i} Z_j}{S_{\Omega_j} Z_i} \tag{4.22}$$

dengan mengasumsikan bahwa gangguan (*noise*) citra bersifat seragam (*uniform*) pada seluruh kelas. Sehingga konstanta normalisasi (*normalizing constant*) Z harus dikalkulasi sekali pada saat *training stage*. Sekarang, karena

$$Z_i = \sum_{U \in \Omega_i} \exp(-E_{\text{int}}(U)) \tag{4.23}$$

komputasi secara menyeluruh (*exhaustive*) dari Z_i adalah proses yang memakan waktu jika Ω_i besar. Sekalipun demikian, konstanta normalisasi dapat diperkirakan menggunakan metoda *Monte-Carlo integration* dan *importance sampling*.

Metoda *Monte-Carlo integration* memperlakukan *integrand* sebagai variabel random dan menggunakan *interative sampling* dan *averaging scheme* untuk menghasilkan sebuah parameter estimasi dari nilai rata-rata (*mean*) yang diambil dari variabel random. Untuk mengaplikasikan metoda ini untuk mendapatkan *Z*, diambil sebuah himpunan dari *q* sampel independen $U_1, U_2, \ldots,$

U_q yang digambar secara *uniform* dari Ω. Estimasi ke-*q* (*q*th *estimate*) dari *Z* adalah

$$Z^q = size(\Omega) \cdot \frac{1}{q} \sum_{i=1}^{q} \exp(-E_{int}(U_i)) \tag{4.24}$$

dimana *size*(Ω) adalah jumlah total dari sampel dalam Ω. Dengan menggunakan *strong law of large number*, $|Z^q - Z| \to 0$ pada saat q $\to \infty$. Lebih jauh, varian dari estimasi $\mathbf{E}[(Z^q - Z)^2]$ berkurang pada nilai 1/q.

Untuk varian deformasi σ_i yang cukup kecil, penjumlahan (*summation*) mengalami puncaknya (*peaked*) pada disekitar kontur referensi $\overline{U}$. Sehingga memungkinkan untuk memanfaatkan konsep *importance sampling* untuk mengkonsentrasikan sampel pada daerah puncak (*peaked region*), yang akan menyebabkan berkurangnya waktu komputasi secara signifikan.

BAB V

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

BAB V

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Dalam bab ini dijelaskan tentang hal-hal yang berkaitan dengan perancangan dan pembuatan perangkat lunak. Dimulai dari kebutuhan serta deskripsi sistem yang digunakan. Kemudian dilanjutkan dengan pembahasan mengenai perancangan perangkat lunak termasuk perancangan tentang data masukan dan keluaran. Untuk menjelaskan perancangan struktur data, dibahas tentang hirarki proses yang digunakan, struktur data serta penjelasan implementasi fungsi-fungsi yang digunakan.

5.1 Kebutuhan Sistem

Dalam perancangan dan pembuatan perangkat lunak ini sistem yang dibutuhkan adalah berupa sistem komputer dengan processor Pentium MMX 166 MHz dengan kapasitas memori sebesar 64 megabyte. Juga dibutuhkan kartu grafik video (*Video Graphic Card*) dengan kapasitas memori minimum sebesar 2 megabyte dan alat masukan berupa keyboard dan mouse.

Alasan menggunakan sistem komputer dengan spesifikasi seperti disebut di atas adalah untuk meningkatkan kinerja perangkat lunak serta mempercepat proses yang dilakukan, terutama proses-proses yang berkaitan dengan pengolahan citra. Hal ini disebabkan proses-proses pada pengolahan citra menggunakan

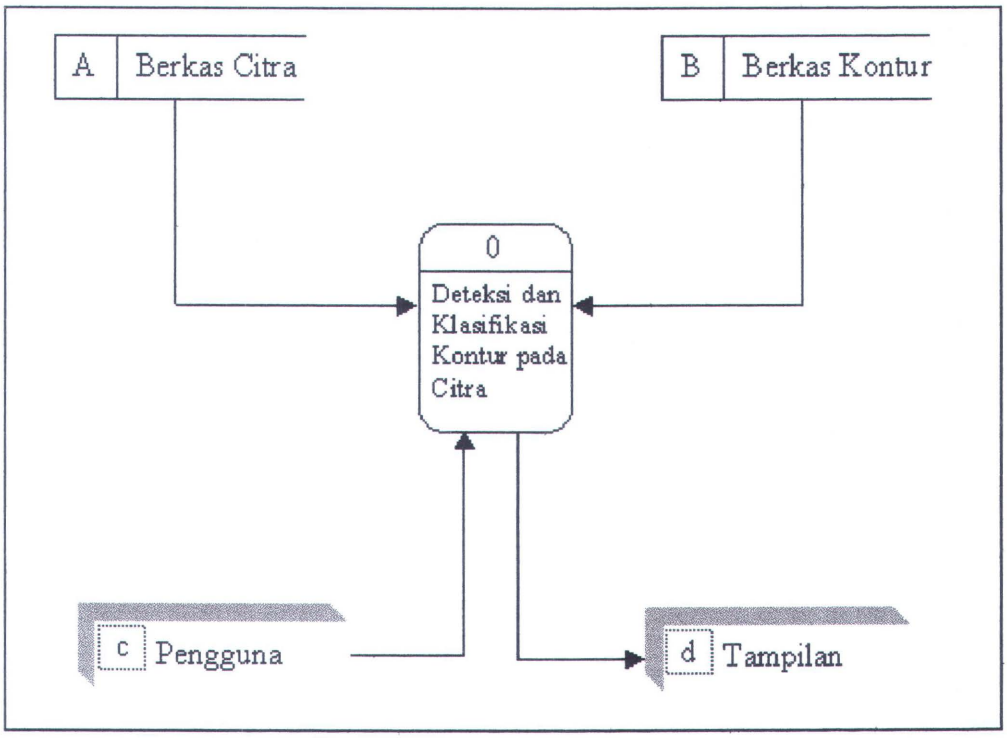
kalkulasi matematis yang rumit serta memakan banyak *memory*. Dan juga disebabkan sistem operasi yang digunakan (Microsoft Windows^{tm} 98) akan bekerja dengan optimal bila diimbangi dengan spesifikasi hardware yang telah disebutkan di atas.

5.2 Deskripsi Sistem Perangkat Lunak

Deskripsi sistem perangkat lunak yang dirancang dijelaskan dengan menggunakan diagram aliran data (*Data Flow Diagram – DFD*) dari model *Gane/Sarson*.

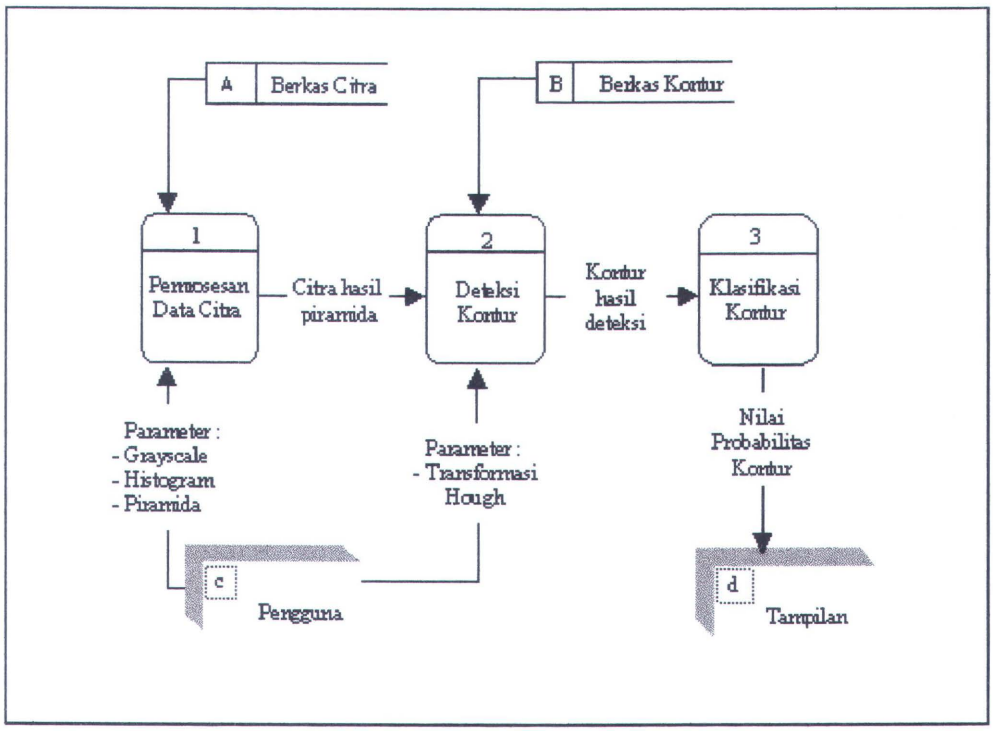
Gambar 5.1 menunjukkan diagram aliran data level 0 dari perangkat lunak yang dirancang. Terdapat aplikasi deteksi dan klasifikasi kontur *deformable* pada citra, yang menerima masukan berupa data dari berkas citra dan berkas kontur untuk diproses. Entitas luar terdiri atas entitas pengguna (*user*) dan entitas tampilan. Entitas pengguna memberikan masukan berupa parameter-parameter yang diperlukan dalam proses, dan entitas tampilan berupa layar monitor sebagai media yang digunakan untuk menampilkan hasil dari proses. Hasil dari proses berupa nilai probabilitas yang mewakili nilai faktor kepercayaan (*Confidence Factor*) dari klasifikasi citra oleh kontur. Nilai inilah yang nantinya dibandingkan dengan nilai probabilitas yang dihasilkan oleh citra lain untuk dapat mengklasifikasikan citra menurut konturnya.

Proses deteksi dan klasifikasi kontur *deformable* pada citra dapat dibagi kedalam beberapa proses tertentu yang saling berkaitan, yaitu proses pengolahan data citra, proses deteksi kontur dan proses klasifikasi kontur pada citra.



Gambar 5.1 Diagram DFD Level 0 : Aplikasi Deteksi dan Klasifikasi Kontur pada Citra

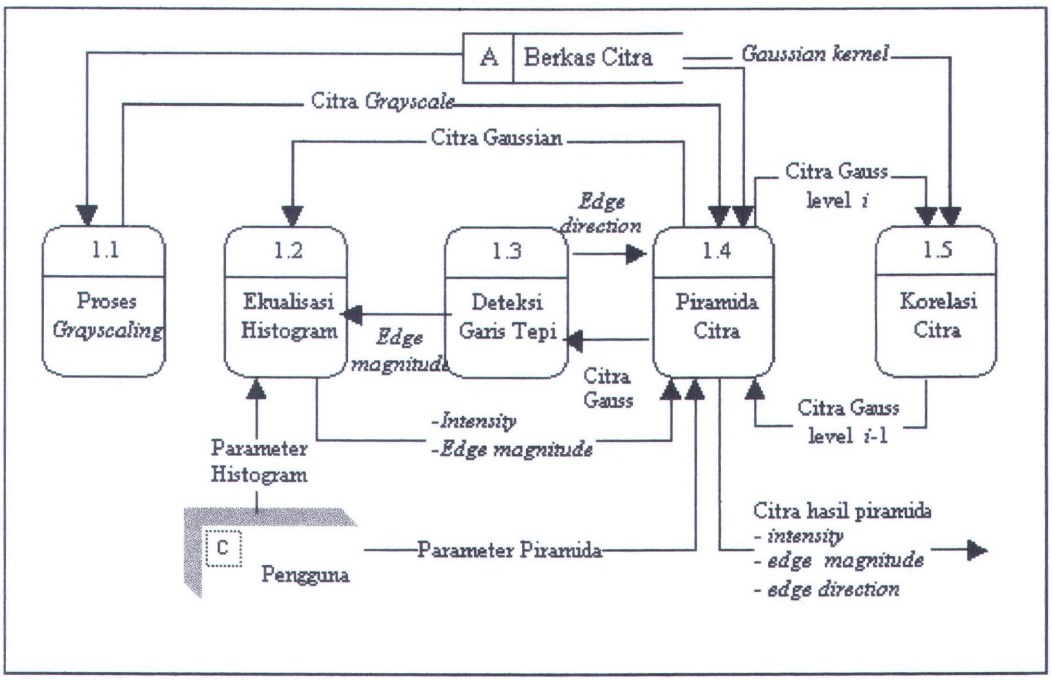
Dalam gambar 5.2 terdapat tiga buah proses utama dalam perangkat lunak ini. Proses pertama yaitu proses pengolahan data citra, digunakan untuk menghasilkan data citra yang diperlukan dalam proses deteksi dan klasifikasi. Proses ini menerima parameter *grayscaling* (proses merubah citra RGB kedalam citra *grayscale*) , histogram (tingkat intensitas dan keabuan) dan parameter piramida citra (jumlah level dan tipe energi eksternal). Proses selanjutnya ialah proses deteksi kontur. Proses ini mengolah citra yang telah dirubah kedalam bentuk piramid oleh proses pertama. Pengguna memberikan nilai-nilai untuk parameter proses, yaitu parameter Transformasi Hough untuk menghasilkan data kontur yang siap diklasifikasikan. Proses ketiga ialah proses klasifikasi kontur. Proses ini mengklasifikasikan kontur yang telah diproses pada proses deteksi



Gambar 5.2 Diagram DFD Level 1 : Proses-proses dalam Deteksi dan Klasifikasi Model Kontur

kontur untuk kemudian dikalkulasi nilai probabilitasnya. Pengguna memberikan parameter-parameter untuk mengkalkulasi nilai probabilitas dari kontur yang telah siap diklasifikasikan. Hasil dari proses ini berupa nilai *confidence level* yang dapat digunakan sebagai pembanding dari nilai probabilitas kontur yang dihasilkan oleh citra lain.

Dalam proses pengolahan data citra, terbagi menjadi lima buah proses seperti yang ditunjukkan dalam diagram aliran data level 2 pada gambar 5.3, yaitu *grayscaling*, ekualisasi histogram, deteksi garis tepi, piramida citra dan korelasi citra. Proses *grayscaling* mengambil data citra dari berkas citra untuk kemudian diolah menjadi citra *grayscale*. Proses piramida citra mengambil data



Gambar 5.3 Diagram DFD Level 2 : Pengolahan Data Citra

citra dari proses *grayscaling* ataupun langsung dari berkas citra apabila citra tersebut telah berbentuk citra *grayscale*. Kemudian citra *grayscale* ini dikorelasikan dengan citra *gaussian kernel*. Kemudian hasilnya diolah pada proses deteksi garis tepi dan ekualisasi histogram. Entitas pengguna memberikan nilai untuk parameter histogram dan piramida. Proses piramida citra menghasilkan tiga jenis data citra tergantung pada tipe dari energi eksternal yang digunakan, yaitu citra *intensity*, *edge magnitude* dan *edge direction*.

5.3 Perancangan Perangkat Lunak

Dalam sub bab ini akan dijelaskan langkah-langkah yang digunakan dalam perancangan data masukan dan data keluaran, struktur data serta penjelasan obyek dan fungsi yang digunakan. Juga akan dijelaskan hirarki proses dan implementasi perangkat lunak secara umum.

5.3.1 Perancangan Data Masukan

Dalam perancangan data masukan, diperhatikan hal-hal yang berkaitan dengan pemrosesan data awal. Dilihat dari proses awal yang akan dikerjakan, ada dua jenis data masukan yang paling utama yaitu data citra dan data kontur.

5.3.1.1 Data Citra

Data citra yang digunakan dalam perangkat lunak ini adalah berupa berkas (*file*) data bitmap, yang dalam sistem operasi Windows dikenal dengan ekstensi BMP (*.BMP). Karena proses pengolahan data citra membutuhkan tingkat presisi dan jumlah data yang cukup memadai, maka dalam perangkat lunak ini hanya dirancang untuk data citra yang memiliki jumlah warna 65535 (24-bit).

Proses pembacaan data citra berupa berkas bitmap dengan mode 24-bit terbagi atas beberapa langkah. Dalam lingkungan sistem operasi Windows berkas bitmap juga dikenal dengan nama berkas *Device-Independent Bitmap* (DIB). Berkas DIB memiliki *file header* dengan struktur (dalam C++) sebagai berikut :

```
Typedef struct tagBITMAPFILEHEADER {
     UINT    bfType;
     DWORD   bfSize;
     UINT    bfReserved1;
     UINT    bfReserved2;
     DWORD   bfOffBits;
};
```

Struktur BITMAPFILEHEADER mengandung informasi tentang tipe, ukuran, dan mode *layout* dari berkas DIB. Deskripsi tiap anggota dari struktur ini dijelaskan dalam tabel 5.1 :

Anggota	Deskripsi
bfType	Menyatakan tipe berkas, untuk DIB harus berisi "BM"
bFSize	Menyatakan ukuran (*size*) dari DIB dalam *byte*
bfReserved1	*Reserved*; harus diset dengan 0
bfReserved2	*Reserved*; harus diset dengan 0
bfOffBits	Menyatakan offset byte dari struktur BITMAPFILEHEADER ke data bitmap sesungguhnya.

Tabel 5.1 Deskripsi anggota BITMAPFILEHEADER

Selanjutnya, struktur BITMAPFILEHEADER diikuti oleh struktur BITMAPINFO yang terdiri atas dua bagian yaitu *bagian bitmap info header* dan *color tabel* (bmiColors[1]). Struktur *bitmap info header* adalah sebagai berikut :

```
typedef struct tagBITMAPINFOHEADER {
     DWORD     biSize;
     LONG      biWidth;
     LONG      biHeight;
     WORD      biPlanes;
     WORD      biBitCount;
     DWORD     biCompression;
     DWORD     biSizeImage;
     LONG      biXPelsPerMeter;
     LONG      biYPelsPerMeter;
     DWORD     biClrUsed;
     DWORD     biClrImportant;
} BITMAPINFOHEADER;
```

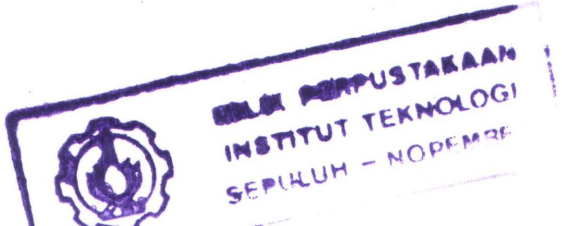
Deskripsi tiap anggota dari struktur ini dijelaskan dalam tabel 5.2 :

Anggota	Deskripsi
biSize	Menyatakan ukuran yang dibutuhkan struktur `BITMAPINFOHEADER` dalam byte
biWidth	Menyatakan lebar dari bitmap, dalam piksel
biHeight	Menyatakan tinggi dari bitmap, dalam piksel
biPlanes	Menyatakan jumlah bidang untuk piranti tujuan, harus bernilai 1
biBitCount	Menyatakan jumlah bit per piksel (1,2,4,8, dan 24)
biCompression	Menyatakan tipe kompresi untuk bitmap yang terkompresi
biSizeImage	Menyatakan ukuran bitmap
biXPelsPerMeter	Menyatakan resolusi horizontal (piksel per meter)
biYPelsPerMeter	Menyatakan resolusi vertikal (piksel per meter)
biClrUsed	Menyatakan jumlah indeks warna yang digunakan pada tabel warna
biClrImportant	Menyatakan indeks warna yang penting, jika bernilai 0 semua warna dianggap penting.

Tabel 5.2 Deskripsi anggota `BITMAPINFOHEADER`

Struktur tabel warna dideskripsikan dalam struktur `RGBQUAD` yang menjelaskan intensitas warna relatif dari merah, hijau dan biru. Anggota *bmiColors* dari struktur `BITMAPINFO` terdiri dari sebuah array yang bertipe data `RGBQUAD`.

Selanjutnya dialokasikan memori sebesar ukuran bitmap untuk membaca data pada offset yang telah ditentukan dalam *bfOffBits*. Dari data bitmap ini, dapat diproses dan ditampilkan ke layar monitor sesuai dengan indeks tabel warna untuk tiap piksel.



5.3.1.2 Data Kontur

Dalam perancangan ini, data kontur merupakan kumpulan dari beberapa buah *snaxel* (*snake pixel*) yang membentuk sebuah rantai berupa kontur. Berkas data kontur memiliki ekstensi con (*.con). Struktur berkas kontur merupakan berkas teks, dengan format :

Bagian	Tipe data
Tipe/*header*	Berkas kontur ('1010')
Mode	0 = OPENED atau 1 = CLOSED
Jumlah *snake pixel*	Integer
White noise variance	Notasi eksponensial
Koordinat snaxel (kolom, baris)	Double
Matriks bentuk (α,β)	Double
Parameter regularisasi (λ)	Notasi eksponensial

Tabel 5.3 *Field* dan tipe data dari berkas kontur

Atribut tipe/*header* memberikan kode 1010 pada awal berkas untuk memberikan tanda bahwa berkas yang ingin dibaca adalah merupakan berkas kontur. Dengan tipe data integer, nilai 0 berarti kontur bersifat terbuka (*opened contour*) dan nilai 1 berarti kontour bersifat tertutup (*closed contour*). Atribut jumlah *snaxel* mendeskripsikan jumlah *snake pixel* yang membentuk kontur. Atribut *white noise variance* mendefinisikan nilai (σ_η^2) pada persamaan (3.16). Posisi tiap *snaxel* didefinisikan pada atribut koordinat dalam susunan (kolom, baris), kemudian diikuti oleh atribut matriks bentuk (α dan β) dan atribut parameter regularisasi (λ).

5.3.2 Perancangan Data Keluaran

Perangkat lunak ini menampilkan berkas citra ke dalam layar monitor, demikian pula untuk berkas kontur. Citra ditampilkan pada *form* utama sebelah kiri sedangkan kontur ditampilkan pada bagian kanan atas *form* utama. Properti dari kontur ditampilkan di bagian bawah gambar kontur untuk memudahkan pengecekan nilai-nilai yang terdapat dalam berkas kontur.

Setelah proses deteksi dan klasifikasi selesai dijalankan maka nilai *Confidence Level* akan terlihat. Apabila proses deteksi belum selesai sepenuhnya maka akan ditampilkan jendela peringatan bahwa proses deteksi belum selesai dan nilai *Confidence Level* tidak ditampilkan.

5.3.3 Implementasi Struktur Data

Berikut ini adalah implementasi struktur data yang digunakan dalam proses deteksi dan klasifikasi kontur yang dideklarasikan dalam bentuk tipe data dari bahasa pemrograman Borland Delphi :

```
type
  // Type of External Energy
  EExtType = (_EdgeMag,_Intensity);
  // Gauss Template for Image Correlation
  GaussType = array[0..2,0..2]of real;
  // Template for accessing image data (intensity)
  TImgTemplate = array[0..200,0..200] of real;
  // Simple Image data type
  _TImage = record
              data : TImgTemplate;
              row,col : integer;
            end;
  // Snake data
  AxRec = record
            row,col : real;       //row , col of snake points
            alpha,beta,lambda : real;
            Eint,Eext,Esnaxel : real;
          end;
```

```
// GSnake properties for Gsnake data calculation
GSnakeRec = record
              global_lambda : Real;
              Eint,Eext,Esnaxel : real;
              EType : EExttype;
              Edgemap,GaussImage : TImgTemplate;
            end;
// Contour data type
TContour = record
             Axel : array[1..100] of AxRec;
             mode,numsnaxel : byte;
             cg_row,cg_col : double;
             sig_nu_sqr : real;
           end;
// Edge Mapping from image
TEdge = record
          Mag, Ang : _TImage;
        end;
// Generalized Hough Transform properties for data calculation
TGHT_OBJ = record
             // Planes of accumulator cells
             Planes   : array[0..15] of _TImage ;
             // Templates under transformations
             Template : array[0..15] of TContour;
             // desired gradient angles
             Angle    : array[0..15,1..100] of double;
             // actual # of planes
             NumPlane : integer;
             // X, Y resolutions of planes
             _Qx, _Qy   : integer;
             // scale resolutions
             _QR, _Qrxy : double;
             // theta resolutions
             _Qt        : double;
             // dilation resolutions
             _Qdx, _Qdy : double;
             // # of cells in each of the axis
             _NR, _Nrxy, _Nt, _Ndx, _Ndy : integer;
             // scaling factor
             _R         : double;
             // initial angle for ref. line angle
             _THETA     : double;
             // store index for blanking
             plane_max: integer;
             row_max  : integer;
             col_max  : integer;
           end;
```

5.3.4 Penjelasan Obyek, Prosedur dan Fungsi yang Digunakan

Dalam proses implementasi, terdapat obyek, prosedur (*procedure*) dan fungsi (*function*) yang digunakan dalam perangkat lunak ini. Obyek yang digunakan dalam perangkat lunak ini merupakan obyek bantu yang mampu mempercepat akses citra bitmap baik dari maupun ke dalam memory.

5.3.4.1 Obyek FastBMP

Obyek ini digunakan untuk mengambil data citra dari berkas citra untuk kemudian ditampilkan ke kanvas dari *windows's form*. Obyek ini mengatasi masalah pembacaan bitmap, penulisan bitmap, pengaturan palette, serta transformasi data citra dari berkas bitmap ke layar monitor. Obyek ini melakukan akses langsung ke memory sehingga dapat mempercepat proses pembacaan maupun pemberian nilai terhadap piksel citra. Penjelasan dan deskripsi tentang fungsi dasar yang digunakan dijelaskan dalam tabel 5.3 :

Fungsi/Prosedur	Kegunaan
Create(cx,cy)	Untuk mangalokasikan bitmap di memory
CreateFromFile(lpFile)	Untuk membaca bitmap dari berkas bitmap
CreateFromHWnd(hBmp)	Untuk membaca bitmap dari bitmap *handle*
CreateCopy(hBmp)	Untuk mengcopy bitmap ke obyek FastBMP lain
Pixels[x,y]	Untuk mengakses data piksel pada x,y
Width	Menyatakan lebar citra bitmap
Height	Menyatakan tinggi citra bitmap
Handle	Menyatakan *handle* dari obyek FastBMP
Draw(hDC,x,y)	Untuk mengirim data bitmap ke layar monitor dalam bentuk gambar

Tabel 5.4 Deskripsi operasi obyek FastBMP

5.3.4.2 Operasi-operasi pada Bitmap

Operasi yang melibatkan bitmap antara lain adalah *grayscale*, *conditioning* (histogram), *correlation*, dan *innerproduct*. Penjelasan masing-masing operasi adalah :

- Operasi *grayscale* digunakan untuk mengubah mode warna pada citra menjadi citra dengan tingkat keabuan.
- Operasi *conditioning* mengimplementasikan proses ekualisasi histogram seperti telah dijelaskan pada persamaan (2.10).
- Operasi *correlation* dan *innerproduct* melakukan proses korelasi citra dengan mengimplementasikan perumusan pada persamaan (2.12), yaitu melakukan korelasi untuk tiap piksel ke piksel dengan menghitung *innerproduct* antara citra *template* (dalam hal ini Gaussian *template*) dan citra yang diproses.

5.3.4.3 Operasi-operasi pada Edge

Operasi yang terkait pada edge antara lain adalah *compute* dan *conditioning*. Penjelasan masing-masing operasi adalah :

- Operasi *computeEdgeMap* melakukan proses deteksi garis tepi yang menghasilkan gradien *magnitude* dan gradien arah dari garis tepi dengan *least square estimation*. Hasil dari proses ini berupa citra *magnitude* dan *angle* yang nantinya dimanfaatkan pada proses berikutnya.
- Operasi *conditioning* meningkatkan intensitas dan ketajaman dari citra *magnitude*. Parameter histogram yang digunakan (*high_pct*=0.95,

low_pct=0.9, *high_val*=0.9, *low_val*=0.2) merupakan nilai *default* yang ditentukan secara empirik, dan mampu menghasilkan kualitas citra *magnitude* yang cukup baik.

5.3.4.4 Operasi-operasi pada Piramida Citra

Operasi-operasi yang berkaitan dengan piramida citra antara lain adalah *genpyramid* dan *conditioning*. Penjelasan masing-masing operasi adalah :

- Operasi *genpyramid*, menghasilkan piramida citra dengan jumlah level tertentu. Operasi pembentukan piramida dimulai dengan mengkorelasikan citra dasar dengan *Gaussian kernel* untuk memperoleh citra yang diperhalus (*smoothed image*) dan menghitung gradien garis tepi untuk tiap-tiap level untuk menghasilkan citra *magnitude* yang dinamakan *edgemap*.
- Operasi *conditioning* dilakukan pada tiap-tiap level untuk mempertajam intensitas citra.

5.3.4.5 Operasi-operasi pada Kontur

Operasi-operasi yang melibatkan data kontur antara lain adalah *dupcon*, *computeCG*, *computeAvgLength*, *contourCentered*, *imageCentered* dan *expand*. Penjelasan masing-masing operasi adalah :

- Operasi *dupcon* melakukan proses duplikasi kontur untuk keperluan transformasi dan kalkulasi agar data kontur awal tidak hilang setelah operasi dilakukan.

- Operasi *computeCG* melakukan perhitungan *centre of gravity* dari kontur sebagai persiapan untuk proses transformasi Hough.
- Operasi computeAvgLength melakukan perhitungan jarak rata-rata tiap *snaxel*.
- Operasi *contourCentered* melakukan konversi sebuah *snake* dari bentuk V (koordinat yang terpusat pada citra) ke dalam bentuk U (koordinat yang terpusat pada obyek kontur yang bersangkutan) relatif terhadap *centre of gravity*.
- Operasi *imageCentered* merupakan kebalikan dari operasi *contourCentered* yaitu melakukan konversi sebuah *snake* dari bentuk U (koordinat yang terpusat pada obyek kontur yang bersangkutan) ke bentuk V (koordinat yang terpusat pada citra).
- Operasi *expand* melakukan perbesaran maupun pengecilan koordinat kontur sebagai persiapan untuk proses transformasi Hough.

5.3.4.6 Operasi-operasi pada Transformasi Hough

Operasi-operasi yang terlibat antara lain adalah *GHTlocalize*, *GHTtransform* dan *GHTgetMax*. Penjelasan tiap-tiap operasi adalah :

- Operasi *GHTlocalize* melokasikan kontur pada posisi yang dianggap paling memungkinkan, dengan menghitung probabilitas terbesar dari posisi dan energi tiap *snaxel* terhadap piksel citra.
- Operasi *GHTTransform* melakukan berbagai macam transformasi linier terhadap kontur dengan memanggil operasi *affineTransform*.

- Operasi *GHTgetMax* menghitung probabilitas terbesar dari posisi dan energi tiap *snaxel* terhadap piksel citra.

5.3.4.7 Operasi-operasi pada Gsnake

Operasi-operasi yang terlibat antara lain adalah *ESnake*, *ESnaxel*, *Einternal* dan *Eexternal*. Penjelasan tiap-tiap operasi adalah :

- Operasi *ESnake*, *ESnaxel*, *Einternal* dan *Eexternal* masing-masing menghitung nilai energi total dari *snake*, energi total *snaxel*, energi internal dan energi eksternal untuk *snaxel*.

5.3.5 Penjelasan prosedur dan Implementasinya

Dalam implementasi perangkat lunak ini terdapat prosedur dan fungsi yang fundametal untuk tiap perumusan dan perhitungan dari dasar teori yang telah dijelaskan sebelumnya. Bagian ini menjelaskan tentang implementasi prosedur dan fungsi berdasarkan teori, dalam urutan pemecahan permasalahan yang ada.

Proses awal dimulai dengan pembacaan data citra bitmap yang penjelasannya dapat dilihat pada perancangan data masukan. Kemudian dilakukan pembacaan data kontur yang digunakan dalam proses deteksi dan klasifikasi ini. Setelah itu dilakukan proses *grayscaling* untuk citra berwarna. Setelah citra mengalami proses *grayscaling* maka dilakukan proses pendeteksian garis tepi menggunakan *least square estimation* dengan operasi *computeEdge*. Proses ini dilakukan bila tipe eksternal energi (*EextType*) yang dipilih oleh pengguna adalah *_EDGEMAG*. Prosedure ini di definisikan sebagai berikut :

```
procedure TMainForm.ComputeEdgeMap(Var Edge : TEdge;
                                   Img : TFastBMP;
                                   low_pct,high_pct,low_val,
                                   high_val,exp : real);
 var i,j                 : integer;
     dx,dy               : double;
     right,left,up,down : real;
 begin
   Edge.Mag.Col := img.Width;
   Edge.Mag.Row := img.Height;
   Edge.Ang.Col := img.Width;
   Edge.Ang.Row := img.Height;
   for i:=0 to img.Height-1 do
     for j:=0 to img.Width-1 do
     begin
       // least square estimation
       down := img.Pixels[CAPS(0,j,img.Width-1),
               CAPS(0,i,img.Height-1)].r +
               img.Pixels[CAPS(0,j+1,img.Width-1),
               CAPS(0,i,img.Height-1)].r;
       up   := img.Pixels[CAPS(0,j,img.Width-1),
               CAPS(0,i+1,img.Height-1)].r +
               img.Pixels[CAPS(0,j+1,img.Width-1),
               CAPS(0,i+1,img.Height-1)].r;
       right := img.Pixels[CAPS(0,j+1,img.Width-1),
               CAPS(0,i,img.Height-1)].r +
               img.Pixels[CAPS(0,j+1,img.Width-1),
               CAPS(0,i+1,img.Height-1)].r;
       left  := img.Pixels[CAPS(0,j,img.Width-1),
               CAPS(0,i,img.Height-1)].r +
               img.Pixels[CAPS(0,j,img.Width-1),
               CAPS(0,i+1,img.Height-1)].r;
       dy := 0.5 * (up - down);
       dx := 0.5 * (right - left);
       // put in gradient magnitude and angle values in edgemap
       Edge.Mag.data[i,j] := hypot(dx, dy);
       Edge.Ang.data[i,j] := ATAN2(dy,dx);
     end;
 Condition(Edge.Mag.data,Edge.Mag.Row,Edge.Mag.Col,low_pct,
           high_pct,low_val,high_val,exp);
 end;
```

Tiap piksel pada data citra dilakukan perhitungan *least square estimation*, dengan perumusan seperti pada persamaan (2.3) untuk nilai d_x dan persamaan (2.4) untuk nilai d_y. Intensitas gradien garis tepi ditentukan oleh fungsi *hypot(dx,dy)*. Fungsi ini merupakan fungsi pada matematika dalam Delphi untuk mencari hipotenusa (sisi miring) pada segitiga siku-siku, seperti model persamaan

(2.5). Arah vektor tangen ditentukan oleh fungsi *ATAN2(dy,dx)* yang juga merupakan fungsi matematika yang merupakan implementasi dari persamaan (2.6).

Setelah seluruh piksel diproses kemudian dilakukan proses ekualisasi histogram dengan parameter dan proses seperti yang telah dijelaskan pada bab sebelumnya, yaitu persamaan (2.8), (2.9) dan (2.10), untuk lebih meningkatkan kekuatan intensitas garis tepi.

Apabila jenis tipe energi yang diberikan adalan *_INTENSITY* maka proses deteksi garis tepi di atas tidak dilakukan dan langsung menginjak proses selanjutnya yaitu proses lokalisasi kontur dengan menggunakan transformasi Hough. Proses ini merupakan bagian dari proses deteksi kontur pada citra. Deskripsi proses yang dimaksud :

```
procedure TMainForm.GHTLocalize(var contour : TContour;
                                Img :_TImage);
begin
  initGHT(Img.row,Img.col);
  GHTTransform(contour); // get various instances of ref. contour
  GHTcorrelate(img) ;    // perform correlation
  getMax(contour);
end;
```

Prosedur *GHTTransform* melakukan transformasi linier kontur terhadap perputaran (rotasi), perubahan skala, dan dilasi/translasi (peregangan). Transformasi linear tersebut diimplementasikan dalam prosedur :

```
procedure TMainForm.GHTTransform(var contour : TContour);

var i,j,k,iR,irxy,it,idx,idy : byte;
    index                    : integer;
    rx,ry,t,tdx,tdy          : double;
    active                   : TContour;
    _angle                   : double;
    uv0_x, uv0_y             : double;     // unit vector
    uv1_x, uv1_y             : double;     // unit vector
    v0_x, v0_y, v1_x, v1_y : double; // directional vectors
    len_v0, len_v1           : double;     // length of v0 and v1
    m0, m1                   : double;     // scalar factor
begin
  computeCG(contour);  // Calculate center of gravity
  contourCentered(contour); // Change from V to U form
  // generate various transformations
  index := 0;
  for iR:=0 to GHT_OBJ._NR-1 do
    for irxy:=0 to GHT_OBJ._Nrxy-1 do
      for it:=0 to GHT_OBJ._Nt-1 do
        for idx:=0 to GHT_OBJ._Ndx -1 do
          for idy:=0 to GHT_OBJ._Ndy - 1 do
            begin
              inc(index);
              rx  := GHT_OBJ._R * (1 + round(irxy -
                     round(GHT_OBJ._Nrxy/2))*GHT_OBJ._Qrxy)*
                     ( 1 + round(iR-round(GHT_OBJ._NR/2))*
                     GHT_OBJ._QR);
              ry  := GHT_OBJ._R * (1 +
                     round(round(GHT_OBJ._Nrxy/2)-
                     irxy)*GHT_OBJ._Qrxy) *
                     ( 1 + round(iR -
                     round(GHT_OBJ._NR/2))*GHT_OBJ._QR) ;
              t   := GHT_OBJ._THETA + round(it -
                     round(GHT_OBJ._Nt/2))*GHT_OBJ._Qt;
              tdx := 0 + round(idx -
                     round(GHT_OBJ._Ndx/2))*GHT_OBJ._Qdx;
              tdy := 0 + round(idy -
                     round(GHT_OBJ._Ndy/2))*GHT_OBJ._Qdy;
              dupcon(contour,GHT_OBJ.Template[index]);
              affinetransform(GHT_OBJ.Template[index],rx, ry, t,
                              0, 0, tdx, tdy);
            end;
 //get desired angle (gradient of snaxel)
 for i:=1 to index do
 begin
   j:=0;
   for k:=1 to GHT_OBJ.Template[i].numsnaxel do
   begin
     if (GHT_OBJ.Template[i].mode = 0) and (k=1) then
       _angle := ATAN2((GHT_OBJ.Template[i].Axel[k].col -
                 GHT_OBJ.Template[i].Axel[k+1].col),
                 GHT_OBJ.Template[i].Axel[k].row -
                 GHT_OBJ.Template[i].Axel[k+1].row))
     else if (GHT_OBJ.Template[i].mode = 0) and
             (k=GHT_OBJ.Template[i].numsnaxel) then
```

(... sambungan dari procedure `TMainForm.GHTTransform`)

```
      _angle := ATAN2((GHT_OBJ.Template[i].Axel[k].col -
                GHT_OBJ.Template[i].Axel[k-1].col),
                (GHT_OBJ.Template[i].Axel[k].row -
                GHT_OBJ.Template[i].Axel[k-1].row))
    else begin
       // === Compute Tangent Vector ===
       // compute directional vector
       v0_x := GHT_OBJ.Template[i].Axel[k].col -
               GHT_OBJ.Template[i].Axel[k-1].col;
       v0_y := GHT_OBJ.Template[i].Axel[k].row -
               GHT_OBJ.Template[i].Axel[k-1].row;
       v1_x := GHT_OBJ.Template[i].Axel[k+1].col -
               GHT_OBJ.Template[i].Axel[k].col;
       v1_y := GHT_OBJ.Template[i].Axel[k+1].row -
               GHT_OBJ.Template[i].Axel[k].row;

       // compute vector length
       len_v0 := hypot( v0_x, v0_y );
       len_v1 := hypot( v1_x, v1_y );

       // compute unit vector
       if len_v0 = 0 then m0 := 0 else m0 := 1.0 / len_v0;
       if len_v1 = 0 then m1 := 0 else m1 := 1.0 / len_v1;
       Uv0_x := m0 * v0_x;
       Uv0_y := m0 * v0_y;
       Uv1_x := m1 * v1_x;
       Uv1_y := m1 * v1_y;

       // compute gradient
       _angle := ATAN2((uv0_x + uv1_x),-1*(uv0_y + uv1_y));
    end;
    GHT_OBJ.angle[i,j] := _angle;
    inc(j)
  end
 end
end;
```

Proses transformasi diatas merupakan proses utama dari deteksi kontur *deformable* pada citra dengan menggunakan transformasi Hough. Setelah dilokalisasi, proses berikutnya adalah kalkulasi energi eksternal dari kontur menggunakan operasi *Esnake*. Prosedur ini diimplementasikan sebagai berikut :

```
// compute total energy of a gsnake

function TMainForm.ESnake(var temp : TContour;
                              level : short):double;
var esnake,eext : double;
    i : byte;
begin
  computeAvgLength(temp);
  // get initial value
  for i:=1 to 100 do
  begin
    temp.Axel[i].Eint := 0;
    temp.Axel[i].Eext := 0;
    temp.Axel[i].Esnaxel := 0;
  end;
  Esnake := 0;      // snake energy
  Eext := 0;        // average gradient power on snake
  // compute snake energy
  for i:=1 to temp.numsnaxel do
  begin
    Esnake := Esnake + ESnaxel(temp,i,TRUE);
    Eext   := Eext + temp.Axel[i].Eext;    // gradient power
  end;
  // compute data strength information
  // ave. gradient power
  GSnake.EExt  := Eext/temp.numsnaxel;
  // compute ave. snake energy
  GSnake.Esnake  := Esnake/temp.numsnaxel;
  result := GSnake.Esnake;
end;
```

Prosedur Esnake ini memanggil fungsi *Snaxel* yang melakukan proses kalkulasi energi *snake* pada posisi *snake pixel* sekarang (*current snake pixel*). Proses ini mengkalkulasi energi snake sesuai dengan type energinya yaitu *_INTENSITY* atau *_EDGEMAG*. Sedangkan energi *snake* piksel itu sendiri terbagi tiga yaitu *model energy*, *image energy* dan *point energy*. Energi inilah yang menentukan nilai dari *Confidence Level*.

BAB VI

UJI COBA DAN EVALUASI PERANGKAT LUNAK

BAB VI

UJI COBA DAN EVALUASI

PERANGKAT LUNAK

Dalam bab ini akan dibahas tentang evaluasi dan uji coba perangkat lunak terhadap beberapa contoh dan permasalahan. Juga akan dijelaskan langkah-langkah dalam melakukan uji coba dan evaluasi. Pembahasan evaluasi dan uji coba diimplementasikan pada model kontur *deformable* yang telah diekstraksi sebelumnya terhadap sampel data citra yang ada.

6.1 Persiapan Data Citra dan Kontur

Perangkat lunak ini memerlukan data citra dan kontur sebagai variabel input. Untuk data citra digunakan bermacam-macam citra yang sebagian telah diberikan gangguan (*noise*) yang bersifat *Gaussian* hingga tingkat tertentu. Gangguan (*noise*) yang diberikan pada citra tertentu ini berfungsi sebagai pembanding pada proses deteksi dan klasifikasi sehingga nilai *Confidence Level* yang dapat diamati.

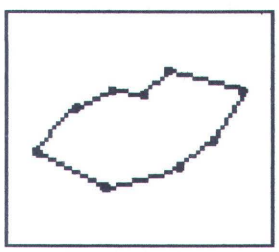
Sedangkan untuk data kontur digunakan data kontur yang sebelumnya telah mengalami proses pelatihan serta ekstraksi kontur. Proses ini dilakukan oleh perangkat lunak lain yang memang khusus dirancang untuk tugas tersebut. Dari proses pelatihan kontur tersebut akan dihasilkan kontur yang memiliki koefisien bentuk (*shape coefficient*) berupa α dan β pada tiap-tiap koordinatnya. Selain itu, diperlukan juga nilai parameter regularisasi (λi) yang mengatur proses deformasi

pada kontur. Parameter regularisasi ini juga dihasilkan oleh perangkat lunak yang sama pada proses pelatihan deformasi. Nilai-nilai dari koefisien bentuk (α ,β) serta parameter regularisasi (λi) yang digunakan dalam proses uji coba pada perangkat lunak ini dapat dilihat dalam tabel 6.1 dibawah ini.

No	col	row	alpha(α)	beta(β)	lambda(λ)
1	119	174	0,666303	1,220284	0,561139
2	128	164	0,443196	0,762797	0,781904
3	138	159	0,451872	0,949198	0,711545
4	146	160	0,296842	0,355789	0,638754
5	153	154	1,561468	0,258716	0,594846
6	172	159	0,941469	1,059776	0,855851
7	164	171	0,508897	0,67298	0,985748
8	153	179	0,646091	0,588477	0,978592
9	138	183	1,004488	0,563734	0,853067

Tabel 6.1 Data kontur yang digunakan

Jenis kontur yang digunakan pada proses uji coba ini adalah kontur tertutup (*closed*) dengan jumlah *snake pixels* sebanyak sembilan buah. Secara visual, bentuk kontur yang digunakan sebagai variabel input pada proses uji coba ini dapat dilihat pada gambar 6.1. Karena bentuknya yang menyerupai bibir maka kontur ini diberi nama kontur bibir.



Gambar 6.1 Bentuk kontur bibir yang digunakan dalam uji coba

6.2 Evaluasi dan Uji Coba

Setelah data citra dipersiapkan dan data kontur diperoleh maka proses uji coba dapat dilaksanakan. Untuk setiap sampel citra yang diuji dilakukan proses penambahan gangguan (*noise*) yang bersifat *Gaussian*. Kuantitas *noise* yang ditambahkan ada dua macam yaitu 10 % dan 20 %. Citra yang telah ditambah *noise* ini digunakan sebagai pembanding nilai *confidence level* yang diperoleh dari citra asli.

Langkah pertama yang dilakukan adalah memberikan parameter-parameter energi yang digunakan serta jumlah sel untuk proses rotasi dan jumlah sel untuk proses peregangan (stretching). Nilai total energi maksimum telah diberikan sebesar 235 sebagai *default*. Nilai ini merupakan nilai yang didapat dari proses uji coba. Dari proses uji coba yang dilakukan nilai energi tertinggi yang didapatkan adalah sebesar 235. Nilai ini akan berpengaruh pada proses penentuan *confidence level*. Bila sebuah citra memiliki nilai total energi kontur hasil deteksi sebesar 235 maka nilai *confidence level*-nya akan mencapai 100 %. Sedangkan apabila nilai total energinya 0 maka nilai *confidence level*-nya akan bernilai 0 %.

Sedangkan apabila nilai total energi dari kontur hasil deteksi kurang dari 0 maka proses deteksi menghasilkan kontur yang tidak valid. Atau dengan kata lain proses deteksi gagal melokalisasi bentuk kontur yang ada di dalam citra. Hal ini dapat disebabkan oleh gagalnya proses transformasi Hough. Proses transformasi Hough dapat mengalami kegagalan apabila kontur *template* yang digunakan untuk melokalisasi telah mencapai tepi dari citra sehingga tidak didapatkan nilai energi yang maksimal. Atau dapat juga disebabkan oleh kurangnya jumlah bidang

(*planes*) yang diperlukan untuk proses transformasi Hough. Jumlah bidang dalam proses transformasi Hough merepresentasikan tingkat keaktifan dan kelenturan dari kontur dalam melakukan proses transformasi.

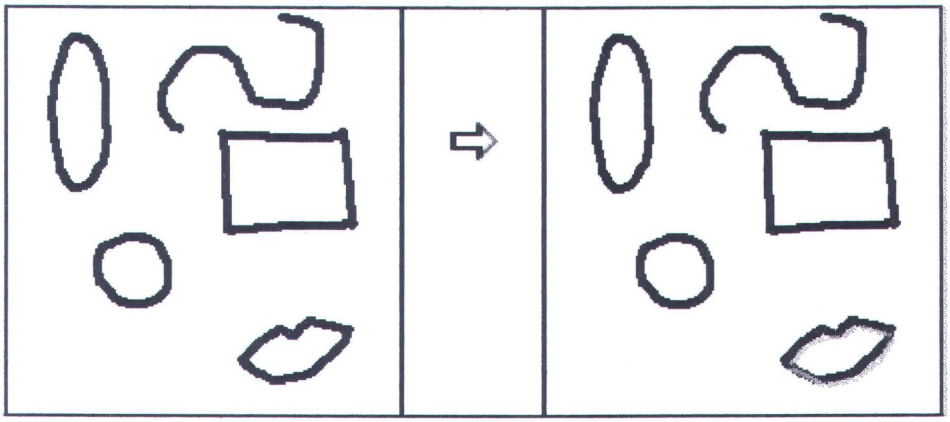
Nilai-nilai yang menentukan keaktifan kontur telah diberikan secara *default* untuk mempersingkat waktu proses (*processing time*). Nilai-nilai tersebut adalah :

- Jumlah bidang rotasi Nt=1 (dapat ditingkatkan).
- Jumlah bidang penskalaan NR=1 (konstan).
- Jumlah bidang translasi arah sumbu x dan y Ndx=1, Ndy=1 (konstan).
- Jumlah bidang peregangan Nrxy=3 (dapat diubah).

Nilai-nilai yang dibatasi tersebut dimaksudkan untuk menghemat waktu proses. Jumlah total bidang transformasi yang diperlukan untuk proses ini adalah perkalian dari jumlah masing-masing jenis transformasi. Jumlah total bidang transformasi maksimal yang dapat diolah oleh perangkat lunak ini sebesar 16 buah bidang (*planes*).

Selain itu diperlukan juga untuk memberikan jenis energi eksternal (*EextType*) yang digunakan dalam proses deteksi dan klasifikasi ini. Nilai *default* untuk *EextType* adalah *_EDGEMAG*. Hal ini disebabkan energi eksternal *_EDGEMAG* memiliki akurasi yang lebih tinggi dibandingkan *_INTENSITY* dalam proses deteksi dan klasifikasi kontur.

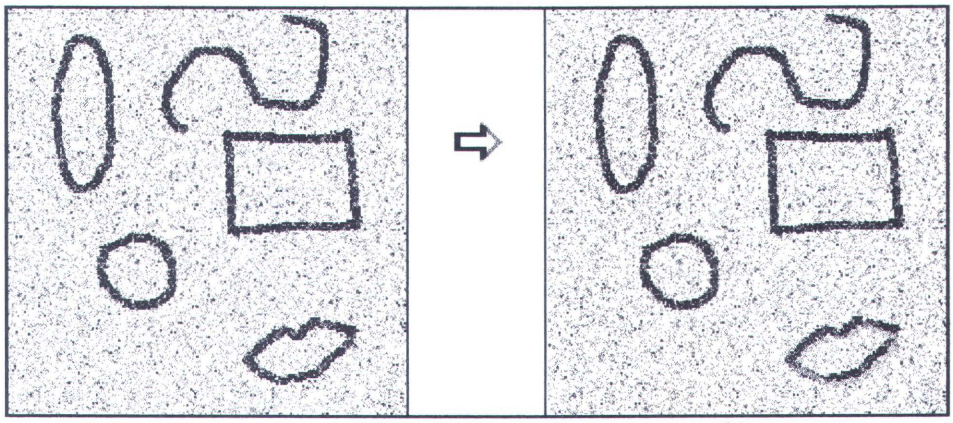
Uji coba pertama dilakukan pada citra A (gambar 6.2 (a)) tanpa adanya *noise*. Hasil proses deteksi menggunakan kontur bibir terlihat pada gambar 6.2 (b). Nilai *confidence level* yang didapat sebesar 16,565 %.



(a) (b)

Gambar 6.2 Proses deteksi kontur pada citra A

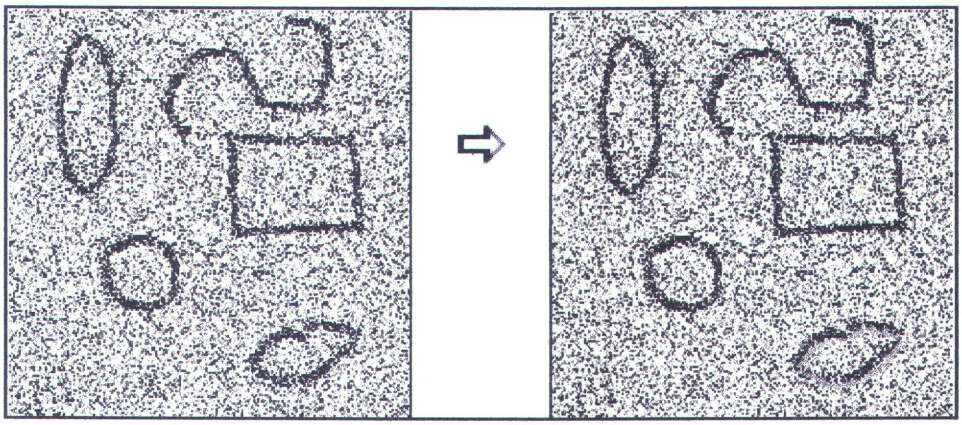
Uji coba kedua dilakukan pada citra A yang dikenai *noise* sebesar 10 % (menggunakan Adobe PhotoDeluxe Business Edition v 1.1) yang untuk selanjutnya diberi nama citra A' (gambar 6.3 (a)). Hasil proses deteksi menggunakan kontur bibir terlihat pada gambar 6.3 (b). Nilai *confidence level* yang didapat sebesar 13,955 %.



(a) (b)

Gambar 6.3 Proses deteksi kontur pada citra A'

Uji coba ketiga dilakukan pada citra A yang dikenai *noise* sebesar 20 % yang untuk selanjutnya diberi nama citra A" (gambar 6.4 (a)). Hasil proses deteksi menggunakan kontur bibir terlihat pada gambar 6.4 (b). Nilai *confidence level* yang didapat sebesar 7,162 %.

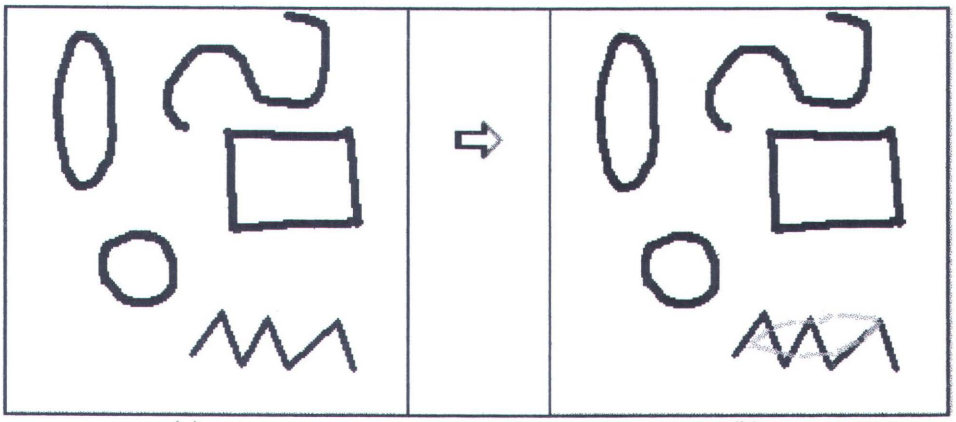


(a) (b)

Gambar 6.4 Proses deteksi kontur pada citra A"

Hasil uji coba di atas menunjukkan bahwa untuk noise yang semakin besar nilai *confidence level* yang didapat semakin kecil. Meskipun demikian proses deteksi telah berhasil melokalisasi kontur bibir dengan sempurna untuk tiap citra baik untuk citra tanpa *noise* maupun citra yang ditambah dengan *noise*.

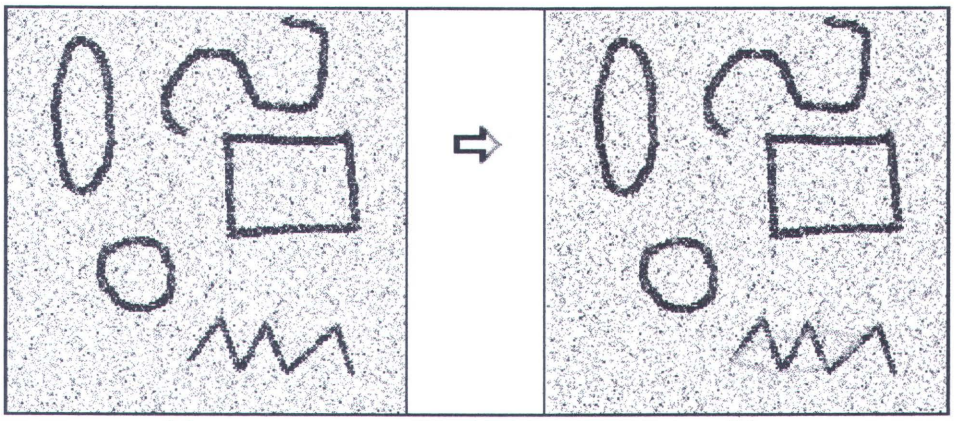
Selanjutnya uji coba akan dikenakan pada citra yang didalamnya tidak terdapat bentuk bibir. Untuk uji coba pertama akan dikenakan pada citra B (gambar 6.5 (a)) tanpa adanya *noise*. Hasil proses deteksi menggunakan kontur bibir terlihat pada gambar 6.5 (b). Nilai *confidence level* yang didapat sebesar 27,976 %.



(a) (b)

Gambar 6.5 Proses deteksi kontur pada citra B

Uji coba kedua dilakukan pada citra B yang dikenai *noise* sebesar 10 % yang untuk selanjutnya diberi nama citra B' (gambar 6.6 (a)). Hasil proses deteksi menggunakan kontur bibir terlihat pada gambar 6.6 (b). Nilai *confidence level* yang didapat sebesar 19,195 %.

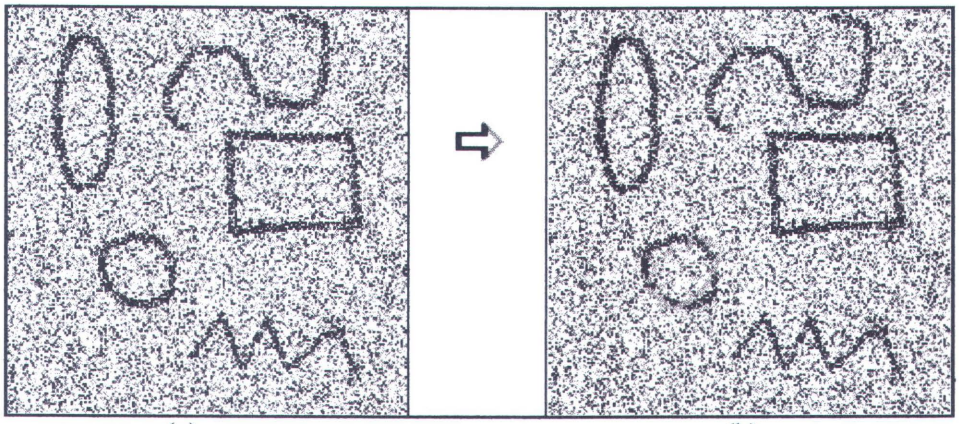


(a) (b)

Gambar 6.6 Proses deteksi kontur pada citra B'

Uji coba berikutnya dilakukan pada citra B dengan *noise* sebesar 20 % yang untuk selanjutnya diberi nama citra B" (gambar 6.7 (a)). Hasil proses deteksi

menggunakan kontur bibir terlihat pada gambar 6.7 (b). Nilai *confidence level* yang didapat sebesar 31,458 %.



(a) (b)

Gambar 6.7 Proses deteksi kontur pada citra B"

Hasil uji coba di atas menunjukkan bahwa tidak selamanya nilai *confidence level* turun untuk *noise* yang semakin besar. Kejadian di atas dapat dimungkinkan oleh adanya *deformable contour*. Kontur bibir yang digunakan dalam tes deteksi melokalisasi lingkaran sebagai dirinya. Sehingga nilai *confidence level* yang dihasilkan dapat lebih tinggi dari citra B dengan *noise* yang lebih rendah ataupun dari citra B tanpa noise sekalipun. Hal ini menunjukkan bahwa kontur *deformable* lebih fleksibel dalam melakukan proses deteksi dan klasifikasi kontur pada citra.

BAB VII

KESIMPULAN DAN SARAN

BAB VII

KESIMPULAN DAN SARAN

Dalam bab ini akan dijelaskan tentang kesimpulan dan saran-saran dari hasil perancangan dan pembuatan perangkat lunak untuk deteksi dan klasifikasi citra dengan *deformable contour*. Kesimpulan yang dihasilkan didasarkan atas penerapan dasar teori, implementasi dalam pembuatan perangkat lunak, serta dari hasil uji coba dan evaluasi perangkat lunak.

7.1 Kesimpulan

Setelah merancang dan membuat perangkat lunak untuk deteksi dan klasifikasi citra dengan *deformable contour*, dan dari hasil uji coba serta evaluasi perangkat lunak, dapat dikemukakan sejumlah kesimpulan.

Pertama, telah dirancang sebuah aplikasi yang mampu mendeteksi keberadaan kontur aktif yang mampu melakukan proses deformasi dalam menelusuri bentuk kontur obyek tertentu pada citra.

Kedua, untuk dapat mendeteksi lokasi kontur pada citra dilakukan proses transformasi Hough. Proses ini dapat tetap melokalisasi kontur meskipun terdapat *noise* (yang bersifat *Gaussian*), dengan melakukan proses deformasi seperlunya hingga mendapatkan korelasi yang maksimal.

Ketiga, untuk dapat mengklasifikasikan citra berdasarkan konturnya, dapat dibuat kelas-kelas yang memiliki batasan-batasan berupa nilai *confidence level*

serta membatasi tingkat kelenturan kontur agar kontur tidak terlalu leluasa untuk bergerak.

Keempat, dari hasil uji coba dan evaluasi perangkat lunak yang dilakukan, diperoleh hasil yang sesuai dengan harapan. Dengan parameter deformasi dan tipe energi yang tepat, didapatkan solusi dari permasalahan deteksi dan klasifikasi citra dengan *deformable contour*.

7.2 Saran

Deteksi dan klasifikasi citra dengan *deformable contour* ini dapat digunakan untuk proses pengenalan obyek, seperti pengenalan bentuk dan model tulisan tangan, dan juga pengenalan karakteristik bentuk obyek tertentu. Diperlukan proses modeling dan ekstraksi kontur yang mampu melakukan pelatihan kontur agar dapat menghasilkan kontur yang dapat digunakan pada proses deteksi dan klasifikasi kontur.

Aplikasi deteksi dan klasifikasi ini hanyalah berupa aplikasi dasar yang nantinya dapat lebih dikembangkan lagi menjadi aplikasi yang lebih terarah pada proses pengenalan obyek ataupun untuk keperluan lain.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

1. Anton, Howard, "Elementary Linear Algebra", *Anton Textbooks, Inc., Fifth Edition*, 1987.
2. Ballard, D. H., "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition,* vol 13, 1981.
3. Duda, R. O. & Hart, P. E., "Use of the Hough Transformation to Detect Lines and Curves in Picture," *Communs. Ass. Comput. Mach.*, vol. 15, 1972.
4. Gonzales, Rafael C. & Woods, Richard E., "Digital Image Processing," *Addison-Wesley Publishing*, 1983.
5. Kindermann, R. & Snell, J. L., "Markov Random Fields and their Application," *American Mathematical Society*, 1980.
6. Lai, K. F. & Chin, R. T., "Deformable Contour: Modelling, Extraction, Detection and Classification," *Proceeding of Computer Visual and Pattern Recognition,* 1994.
7. Lai, K. F. & Chin, R. T., "On Regularization, Formulation and Initialization of the Active Contour Models (Snakes)," *IEEE Trans. Pat. Anal. Mach. Intell.*, 1993.
8. Marr, D. & Hildreth, E., "Theory of Edge Detection," *Proceeding of Royal Society of London*, B207, 1980.
9. Walpole, Ronald E. & Myers, Raymond H.,"Probability and Statistics for Engineers and Scientists," *Macmillan Publishing Co., Inc., Second Edition*, 1978.

LAMPIRAN

LAMPIRAN

Pengamatan Data Kontur Bibir pada Proses Deteksi Kontur dengan Menggunakan Transformasi Hough

Untuk mengetahui logika dan alur dari proses transformasi Hough, dilakukan pengamatan data kontur bibir pada tiap iterasi. Data kontur bibir yang diamati hanya meliputi data koordinat kontur saja karena nilai α, β, dan λ tidak mengalami perubahan.

Sebelum proses iterasi transformasi Hough dilakukan, terlebih dahulu dilakukan perhitungan *center of gravity* dari kontur bibir. Kemudian dilakukan proses perubahan bentuk koordinat kontur dari bentuk V (*image centered coordinates*) menjadi bentuk U (*object centered coordinates*) dengan cara mengurangi koordinat kontur dengan *center of gravity*-nya. Pada perhitungan *center of gravity* dihasilkan koordinat (145.667, 167). Hasil perubahan nilai koordinat kontur dari data koordinat kontur semula dapat diamati pada tabel 8.1.

No	col(V)	row(V)	col(U)	row(U)
1	119	174	-26,667	7
2	128	164	-17,667	-3
3	138	159	-7,667	-8
4	146	160	0,333	-7
5	153	154	7,333	-13
6	172	159	26,333	-8
7	164	171	18,333	4
8	153	179	7,333	12
9	138	183	-7,667	16

Tabel 8.1 Konversi koordinat kontur bibir dari bentuk V ke bentuk U

Prosedur yang digunakan dalam proses transformasi Hough adalah sebagai berikut :

```
// sx,  x-scale changes
// sy,  y-scale changes
// rt,  rotation changes
// tx,  x-translation changes
// ty,  y-translation changes
// idx, x-dilation changes
// idy  y-dilation changes
procedure TMainForm.affinetransform(var contour : TContour;
                    sx,sy,rt,tx,ty,idx,idy : double);
var i : byte;
    row,col : double;
begin
  for i:=1 to contour.numsnaxel do
  begin
    col := (cos(rt) + sin(rt)*idy) * contour.axel[i].col +
           (cos(rt) * idx + sin(rt)) * contour.axel[i].row;
    row := (-sin(rt)+ cos(rt)*idy) * contour.axel[i].col +
           (-sin(rt)*idx + cos(rt)) * contour.axel[i].row;
  // scaling
    col := col * sx;
    row := row * sy;
  // translation
    col := col + tx;
    row := row + ty;
    contour.Axel[i].col := col;
    contour.Axel[i].row := row;
  end;
end;
```

Setelah diaplikasikan pada prosedur transformasi diatas, berikut ini adalah hasil pengamatan nilai koordinat kontur bibir pada tiap iterasi dari proses transformasi Hough :

Iterasi ke	Koordinat kontur bibir dalam bentuk U (col, row)
1	(-20, 8.75); (-13.25, -3.75); (-5.75, -10); (0.25, -8,75); (5.5 , -16.25); (19.75, -10); (13.75, 5); (5.5, 15); (-5.75, 20)
2	(-26.67, 7); (-17.67, -3); (-7.67, -8); (0.33, -7); (7.33, -13); (26.33, -8); (18.33, 4); (7.33, 12); (-7.67, 16)
3	(-33.33, 5.25); (-22.08, -2.25); (-9.58, -6); (0.42, -5.25); (9.17, -9.75); (32.92, -6); (22.92, 3); (9.17, 9); (-9.58, 12)

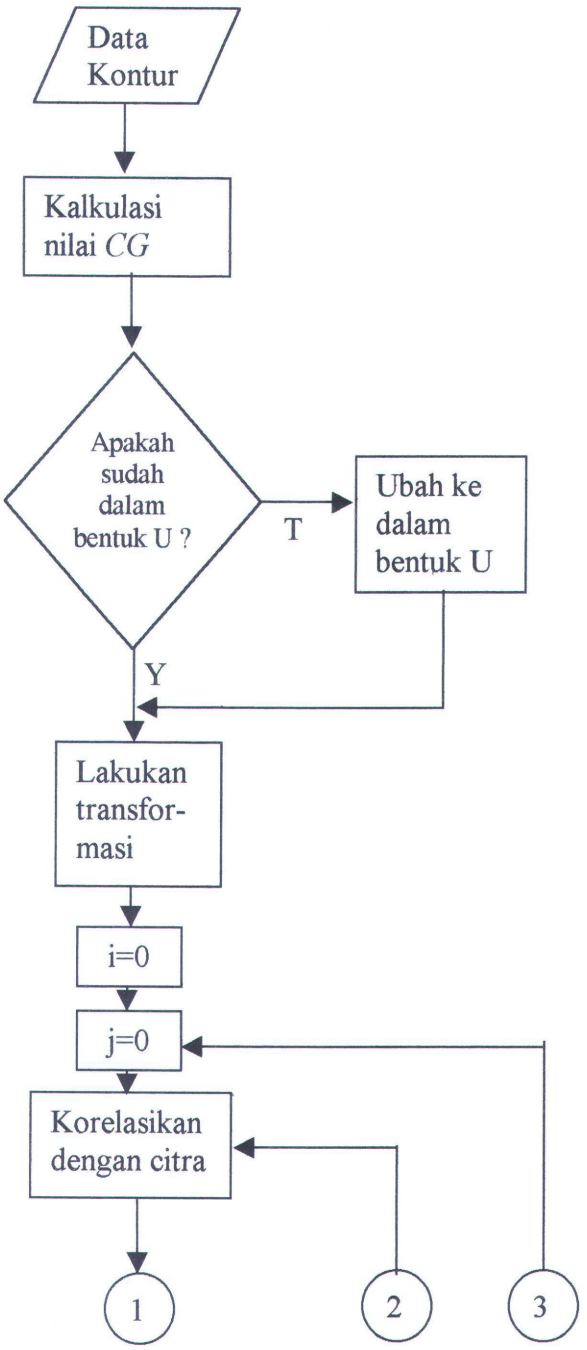
Tabel 8.2 Koordinat kontur bibir hasil transformasi Hough

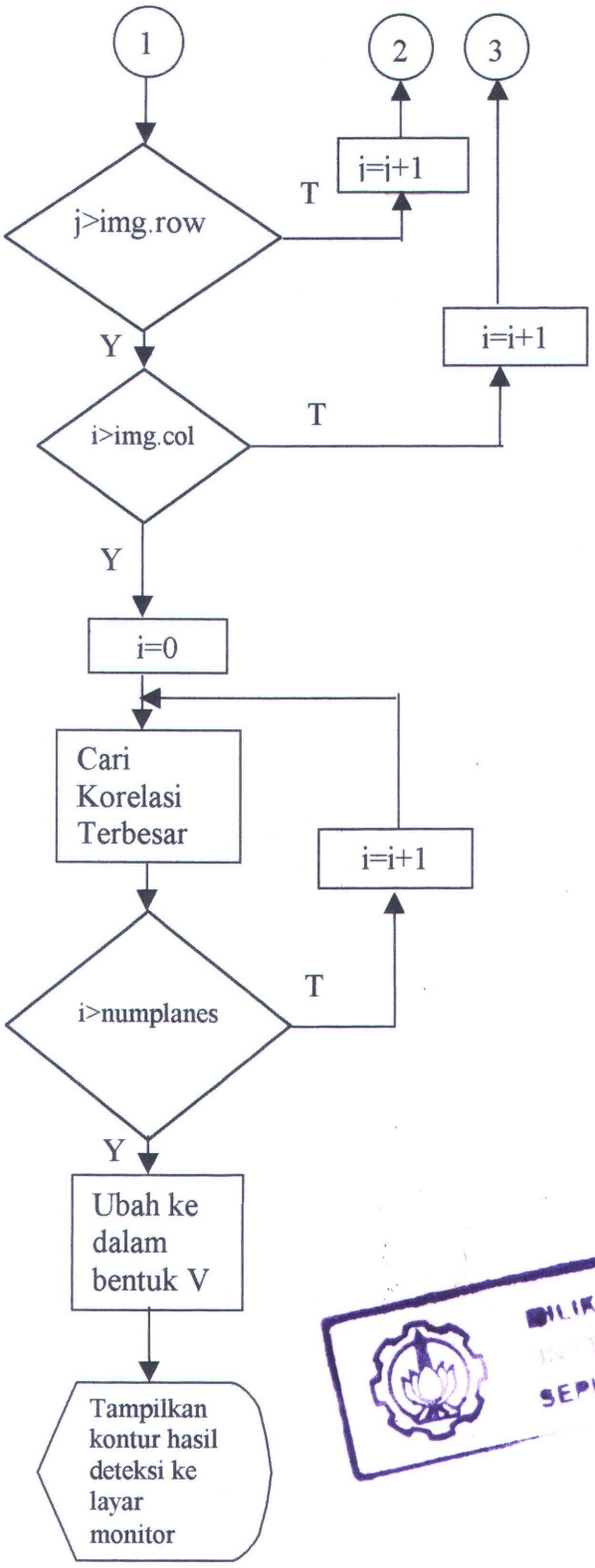
Dari iterasi diatas dapat dilihat bahwa kontur hasil transformasi telah terbentuk. Kontur-1 terbentuk dari iterasi pertama, kontur-2 terbentuk dari iterasi kedua dan kontur-3 terbentuk dari iterasi ketiga.

Setelah itu pada tiap-tiap kontur hasil transformasi (kontur-1, kontur-2, kontur-3) dilakukan proses korelasi antara citra *pyramid* dengan masing-masing kontur dari posisi (0,0) hingga posisi (*img.col*, *img.row*). Kemudian dilakukan proses seleksi kontur dengan membandingkan nilai korelasi masing-masing kontur. Pada proses korelasi kontur dilakukan kalkulasi bidang (*planes*) dari masing-masing kontur. Kemudian dilakukan pembandingan bidang (*planes*) dari masing-masing kontur untuk mencari korelasi terbesar. Setelah bidang dari kontur dengan korelasi terbesar didapatkan maka koordinat kontur tersebut diubah dari

bentuk U menjadi bentuk V. Kontur inilah yang memenuhi kriteria deteksi dan hasilnya ditampilkan ke layar monitor.

Secara umum *flowchart* dari proses deteksi kontur dapat dilihat pada gambar berikut ini.





Gambar 8.1 Flowchart dari proses deteksi kontur