3100001032 48

PERPUSTAKAAN ITS

# PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENCOCOKAN OBYEK DENGAN DEFORMABLE TEMPLATE YANG MENGGUNAKAN REPRESENTASI KURVA B-SPLINE

## TUGAS AKHIR

RSIf
005.1
Ikh
p-1
2000

Oleh :

ZULI IKHSANTO
2692.100.018

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

Rp. 30.000
PERPUSTAKAAN
Tgl. 04/01/2001
H
No. 21 2673

# PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENCOCOKAN OBYEK DENGAN DEFORMABLE TEMPLATE YANG MENGGUNAKAN REPRESENTASI KURVA B-SPLINE

## TUGAS AKHIR

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

**Mengetahui / Menyetujui**

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Dr. Ir. ARIEF DJUNAIDY, M.Sc. | RULLY SOELAIMAN, S.Kom. |
| NIP. 131 633 403 | NIP. 132 085 802 |

**SURABAYA**
**Februari, 2000**

"Jangan lupakan kebaikan orang lain kepada kita
tetapi lupakanlah keburukannnya
dan
jangan ingat kebaikan kita kepada orang lain
tetapi ingatlah keburukan kita."

*Kupersembahkan buku ini*
*untuk Ayah dan Ibu tercinta*

# ABSTRAK

# ABSTRAK

Pencarian obyek berdasarkan bentuknya dari suatu citra yang komplek telah banyak diaplikasikan sebelumnya. Pada awalnya obyek yang dapat dikenali adalah bersifat *rigid* (kaku). Dalam tugas akhir ini pencarian dilakukan dengan menggunakan suatu *deformable template* sehingga obyek yang dapat dikenali bersifat lebih elastis. Untuk membentuk *deformable template* digunakan suatu prototipe template yang menggambarkan kontur/tepi dua dimensi dari suatu obyek dalam bentuk bitmap dan set parameter *B-Spline*.

Proses pencocokan dilakukan dengan cara mendeformasi *deformable template* ke arah tepi obyek. Deformasi dilakukan dalam dua tahap, yaitu: global dan lokal. Deformasi global dilakukan dengan proses translasi, rotasi, dan *Generalized Hough Transform* (GHT). Sedangkan deformasi lokal dilakukan dengan cara memindahkan *control point* ke arah tepi obyek terdekat dengan menggunakan metoda *Descent Gradient*. Proses pencocokan itu sendiri dilakukan dengan menggunakan algoritma *Multiresolution Coarse-to-Fine*. Proses pencocokan ini dilakukan secara bertahap pada tiga tingkatan resolusi, yaitu: *coarsest*, *finer*, dan *finest*. Dengan menggunakan kerangka kerja *Bayesian*, informasi *prior* dan *likelihood* digunakan untuk menemukan suatu nilai kecocokan antara *deformable template* dengan obyek dalam citra masukan.

Dari proses pencocokan ini, obyek yang dapat dikenali adalah obyek berbentuk kurva/poligon yang terbuka/tertutup dengan tidak dibatasi oleh posisi, orientasi, dan ukuran yang telah ditentukan. Selain itu, perangkat lunak yang telah dibuat dapat juga digunakan untuk melakukan pencarian obyek dari citra masukan yang banyak jumlahnya dengan menggunakan sebuah *deformable template*.

# KATA PENGANTAR

# KATA PENGANTAR

Syukur Alhamdulillah Penulis ucapkan, bahwa hanya dengan ridhlo Allah SWT Penulis bisa menyelesaikan laporan Tugas Akhir dengan judul:

## "PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENCOCOKAN OBYEK DENGAN DEFORMABLE TEMPLATE YANG MENGGUNAKAN REPRESENTASI KURVA B-SPLINE."

Tugas akhir ini dibuat guna memenuhi sebagian persyaratan untuk memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Dalam tugas akhir ini dibahas tentang pencocokan obyek dengan menggunakan *deformable template*. Dimana teknik ini merupakan salah satu teknik pencarian obyek citra yang berdasarkan pada bentuknya.

Dengan terselesaikannya penulisan laporan Tugas Akhir ini, Penulis tak lupa mengucapkan rasa terima kasih yang tulus dan sebesar-besarnya kepada semua pihak yang telah membimbing dan membantu selama proses pembuatan Tugas Akhir ini. Ucapan terima kasih ini Penulis sampaikan khususnya kepada:

1. Ayah dan ibu yang selalu memberikan dorongan baik berupa materi, semangat, doa dan kasih sayang.
2. Bapak Dr. Ir. Arief Djunaidy, M.Sc. dan Bapak Rully Soelaiman, S.Kom., selaku dosen pembimbing yang telah banyak mencurahkan waktu dalam rangka memberikan bimbingan dan saran untuk penyusunan Tugas Akhir ini.

3. Ibu Dr. Ir. Handayani Tjandrasa, M.Sc. selaku dosen wali yang telah banyak membantu dalam hal perwalian selama kuliah.
4. Para Dosen Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember yang telah banyak memberikan ilmunya.
5. Para Staf Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember yang telah banyak membantu kelancaran dalam hal urusan administrasi.
6. Rekan-rekan Penulis, diantaranya: Edul, Cahyo, Anang, Irfan, Wanto, Agung, Choliq, Rum, Ucok, Ndaru, Bowo dan rekan-rekan Lab. Sistem Informasi, Ipul, Yono, Hasan, Mas Budi, dan rekan-rekan se-kost atas bantuannya secara langsung kepada Penulis baik dalam hal pemrograman, pembahasan rumus, penyediaan sarana dan prasarana, dan pengumpulan referensi yang Penulis gunakan.
7. Rekan-rekan C-08 semuanya tanpa terkecuali yang telah banyak membantu dalam perkuliahan.
8. Rekan-rekan C-09, C-0A serta rekan-rekan Lab. AJK yang pernah membantu Penulis.
9. Saudari Nunuk atas doanya.
10. Kakek dan nenek keduanya, paman dan bibi semuanya, serta adik-adikku atas bantuan dan doanya.
11. Semua pihak yang telah membantu baik secara langsung maupun tidak langsung yang belum disebutkan diatas.

Dengan tersusunnya Tugas Akhir ini, harapan Penulis semoga dapat memberikan manfaat sebaik-baiknya kepada pembaca yang membutuhkan informasi tentang Tugas Akhir ini.

Penulis menyadari bahwa laporan ini belum sempurna seperti yang diharapkan. Oleh karena itu, kritik dan saran yang bersifat membangun dari pembaca sangat berharga sekali untuk laporan ini demi terciptanya suatu karya yang lebih baik dimasa mendatang.

Surabaya, Februari 2000

Zuli Ikhsanto

# DAFTAR ISI

# DAFTAR ISI


| | Hal. |
|---|---|
| **ABSTRAK** | **i** |
| **KATA PENGANTAR** | **ii** |
| **DAFTAR ISI** | **v** |
| **DAFTAR GAMBAR** | **vii** |
| **DAFTAR TABEL** | **xi** |
| **BAB I PENDAHULUAN** | **1** |
| 1.1 Latar Belakang | 1 |
| 1.2 Perumusan Masalah | 2 |
| 1.3 Tujuan Penelitian | 2 |
| 1.4 Manfaat Penelitian | 3 |
| 1.5 Batasan Masalah | 3 |
| 1.6 Metodologi Penelitian | 4 |
| 1.7 Sistematika Pembahasan | 5 |
| **BAB II MODEL DEFORMASI TEMPLATE** | **7** |
| 2.1 Hirarki Pencocokan Obyek dengan Template | 7 |
| 2.2 Representasi Deformable Template | 9 |
| 2.2.1 Prototipe Template | 9 |
| 2.2.2 Transformasi Deformasi dengan B-Spline | 10 |
| 2.2.3 Model Probabilistik Deformasi | 16 |
| 2.3 Formulasi Bayesian dan Fungsi Obyektif | 17 |
| 2.3.1 Distribusi Prior | 17 |
| 2.3.2 Likelihood | 18 |
| 2.3.3 Probabilitas Densitas Posterior | 21 |
| 2.3.4 Fungsi Obyektif | 21 |
| **BAB III PROSES PENCOCOKAN OBYEK** | **23** |
| 3.1 Segmentasi Citra | 24 |

3.2 Transformasi Hough yang Dibakukan 29
3.3 Metoda Gradien Menurun 31
3.4 Algoritma Multiresolusi Coarse-to-Fine 32

**BAB IV PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK 35**

4.1 Perancangan Perangkat Lunak 35
4.1.1 Perancangan Data 35
4.1.1.1 Data Masukan 36
4.1.1.2 Data Internal 37
4.1.1.3 Data Keluaran 38
4.1.2 Perancangan Proses 38
4.2 Pembuatan Perangkat Lunak 43

**BAB V UJI COBA DAN ANALISA HASIL PERANGKAT LUNAK 59**

5.1 Uji Coba dengan Obyek yang Orientasi dan Ukurannya Tidak Jauh Berbeda 60
5.2 Uji Coba dengan Obyek yang Berbeda Orientasinya 63
5.3 Uji Coba dengan Obyek yang Berbeda Ukurannya 63
5.4 Uji Coba dengan Obyek yang Berbeda Orientasi dan Ukurannya 64
5.5 Uji Coba dengan Banyak Obyek 65
5.6 Uji Coba dengan Banyak Citra 67
5.7 Uji Coba Perbandingan 68

**BAB VI PENUTUP 71**

6.1 Kesimpulan 71
6.2 Kemungkinan Pengembangan 72

**DAFTAR PUSTAKA 74**

**LAMPIRAN PANDUAN PENGGUNAAN PERANGKAT LUNAK 75**

# DAFTAR GAMBAR

# DAFTAR GAMBAR


Hal.

Gambar 2.1: Hirarki teknik pencocokan obyek dengan menggunakan *template*. 8

Gambar 2.2: Prototipe *template* dengan parameter-parameternya (tanda silang). 10

Gambar 2.3: Kurva *B-Spline non rational non uniform* terbuka. (a) n = 6 dan k = 3. (b) n = 6 dan k = 4. 13

Gambar 2.4: Kurva *B-Spline non rational non uniform* tertutup. (a) n = 6 dan k = 3. (b) n = 6 dan k = 4. 13

Gambar 2.5: Kurva *B-Spline non rational uniform* terbuka. (a) n = 6 dan k = 3. (b) n = 6 dan k = 4. 14

Gambar 2.6: Kurva *B-Spline non rational uniform* tertutup. (a) n = 6 dan k = 3. (b) n = 6 dan k = 4. 14

Gambar 2.7: Representasi *deformable template*. (a) Prototipe *template* dengan parameter-parameternya (tanda silang). (b) Bentuk tepi/kontur hasil transformasi dengan menggunakan fungsi basis *B-Spline non rational uniform* tertutup dengan $k = 3$. 15

Gambar 2.8: Medan potensial citra tepi masukan. Nilai potensial ditunjukkan oleh nilai *grayscale*-nya, semakin besar nilai *grayscale* semakin besar potensialnya. 19

Gambar 2.9: *Deformable template* diletakkan dalam medan potensial. $(x,y)$ adalah sebuah titik pada *deformable template*. $(x_e,y_e)$ adalah titik tepi terdekat dari $(x,y)$. $\beta(x,y)$ adalah sudut yang dibentuk antara $(x,y)$ dengan $(x_e,y_e)$. 20

Gambar 3.1: *Non Maximum Suppression*. 27

Gambar 3.2: Deteksi tepi *Canny*: (a) Citra asal. (b) Citra tepi ($\sigma = 1$, *HighTh* = 0,7, *LowTh* = 0,5, dan ukuran jendela = 9x9). Citra tepi ($\sigma = 1$, *HighTh* = 0,9, *LowTh* = 0,5, dan ukuran jendela = 9x9). (d) Citra tepi ($\sigma = 3$, *HighTh* = 0,7, *LowTh* = 0,5, dan ukuran jendela = 9x9). (e) Citra tepi ($\sigma = 3$, *HighTh* = 0,9, *LowTh* = 0,5, dan ukuran jendela = 9x9). 28

Gambar 3.3: Konstruksi *R-table* dengan *Generalized Hough Transform*. 31

Gambar 4.1: DFD proses pencocokan obyek tingkat 0. 39

Gambar 4.2: DFD proses pencocokan obyek tingkat 1. 40

Gambar 4.3 DFD proses pembentukan *deformable template* tingkat 2. 41

Gambar 4.4: DFD praproses citra tingkat 2. 42

Gambar 4.5: DFD algoritma *Multiresolution Coarse-to-Fine* tingkat 2. 42

Gambar 5.1: Data *deformable template* dengan parameter-parameternya. 60

Gambar 5.2: Citra asal (Duck.bmp – 350x275). 60

Gambar 5.3: Citra tepi dari Gambar 5.2 ($\sigma = 1$, *HighTh* = 0,9, dan *LowTh* = 0,5). (a) Tingkat *coarsest*. (b) Tingkat *finer*. (c) Tingkat *finest*. 61

Gambar 5.4: Medan potensial tepi dari Gambar 5.3: (a) Tingkat *coarsest* ($\rho = 12$). (b) Tingkat *finer* ($\rho = 8$). (c) Tingkat *finest* ($\rho = 3$). 61

Gambar 5.5: Arah tepi dalam medan potensial tepi dari Gambar 5.4: (a) Tingkat *coarsest*. (b) Tingkat *finer*. (c) Tingkat *finest*. 61

Gambar 5.6: Hasil proses pencocokan dengan algoritma *Multiresolution Coarse-to-Fine*. (a) Tingkat *coarsest*. (b) Tingkat *finer*. (c) Tingkat *finest*. 62

Gambar 5.7: Hasil proses pencocokan dengan algoritma *Multiresolution Coarse-to-Fine* pada tingkat *finest* yang terbaik ($L = 0,228$). 62

Gambar 5.8: Pencocokan obyek dengan *deformable template* yang berbeda orientasinya. (a) *Deformable template*. (b) Hasil pencocokan ($L = 0,232$). 63

Gambar 5.9: Pencocokan obyek dengan *deformable template* yang berbeda ukurannya. (a) *deformable template*. (b) Hasil pencocokan ($L = 0,229$). 64

Gambar 5.10: Pencocokan obyek dengan *deformable template* yang berbeda orientasi dan ukurannya. (a) *Deformable template*. (b) Hasil pencocokan ($L = 0,223$). 65

Gambar 5.11: *Deformable template* dengan obyek variasinya. (a) *Deformable template*. (b) Obyek dengan orientasi yang berbeda. (c) Obyek dengan ukuran yang berbeda. (d) Obyek dengan orientasi dan ukuran yang berbeda. 66

Gambar 5.12: Pencocokan *deformable template* dengan banyak obyek dalam sebuah citra masukan. (a) *Deformable template*. (b) Hasil pencocokan. 66

Gambar 5.13: Pencocokan sebuah *deformable template* dengan beberapa citra masukan. (a) *Template*. (b) Alligator.bmp ($L$ = 0,223). (c) Chameleon.bmp ($L$ = 0,239). (d) Column.bmp ($L$ = 0,228). Eagle.bmp ($L$ = 0,242). (f) Liberty.bmp ($L$ = 0,238). (g) Lizard.bmp ($L$ = 0,229). (h) Monkey.bmp ($L$ = 0,249). (i) Rocket.bmp ($L$ = 0,231). (j) Seal.bmp ($L$ = 0,248). (k) Shark.bmp ($L$ = 0,235). 68

Gambar 5.14: Pencocokan perbandingan beberapa *deformable template* dengan beberapa obyek. (a) *Deformable template*. (b) Citra masukan. 69

Gambar L.1: Tampilan antarmuka utama perangkat lunak. 75

Gambar L.2: Tampilan antarmuka hasil proses pencocokan antara citra masukan dengan template yang dihasilkan oleh perangkat lunak ini. 77

Gambar L.3: Tampilan template dengan ukuran yang sebenarnya. 78

Gambar L.4: Tampilan menu *File*. 78

Gambar L.5: Tampilan antarmuka *file list template*. 79

Gambar L.6: Tampilan antarmuka *file list image*. 79

Gambar L.7: Tampilan menu *Run*. 80

Gambar L.8: Tampilan menu *Setting*. 80

Gambar L.9: Tampilan antarmuka konfigurasi *directory* dan *file*. 81

Gambar L.10: Tampilan antarmuka konfigurasi parameter-parameter *Canny*. 81

Gambar L.11: Tampilan antarmuka konfigurasi parameter-parameter pencocokan. 81

Gambar L.12: Tampilan menu *Tools*. 82

Gambar L.13: Tampilan menu *View Template*. 82

Gambar L.14: Tampilan menu *Colors*. 83

Gambar L.15: Tampilan menu *Help*. 83

Gambar L.16: Tampilan *About*. 83

Gambar L.17: Tampilan antarmuka untuk uji coba pembentukan kurva *B-Spline*. 84

Gambar L.18: Tampilan antarmuka Editor Template. 85

Gambar L.19: Tampilan menu *File* dalam Editor Template. 86

Gambar L.20: Tampilan menu *Edit* dalam Editor Template. 87

Gambar L.21: Tampilan menu *B-Spline* dalam Editor Template. 88

Gambar L.22: Tampilan antarmuka attribut kurva *B-Spline*. 88

Gambar L.23: Tampilan menu *Tools* dalam Editor Template. 89

Gambar L.24: Tampilan antarmuka masukan untuk proses deformasi template secara global. 89

Gambar L.25: Tampilan menu *View* dalam Editor Template. 89

Gambar L.26: Tampilan menu *Colors* dalam Editor Template. 90

Gambar L.27: Tampilan menu *File* dalam Editor File List. 91

Gambar L.28: Tampilan antarmuka Editor File List. 93

# DAFTAR TABEL

# DAFTAR TABEL

Hal.

Tabel 5.1: Hasil pencocokan perbandingan. 69

# BAB I

# PENDAHULUAN

# BAB I

# PENDAHULUAN

Dalam bab ini dijelaskan beberapa hal pokok tentang penelitian yang telah dilakukan. Hal-hal pokok yang dijelaskan yaitu tentang latar belakang, perumusan masalah, tujuan, manfaat, batasan masalah, metodologi, dan sistematika pembahasan penelitian.

## 1.1 Latar Belakang

Dalam era multimedia sekarang ini, pencarian suatu obyek citra merupakan hal yang sangat dibutuhkan. Metoda pencarian citra dapat dikelompokkan berdasarkan informasi yang digunakan untuk proses pencariannya, misalnya: teks, deskripsi, warna, tekstur, bentuk obyek, dan lain-lain. Penelitian ini hanya difokuskan untuk membahas salah satu dari metoda pencarian tersebut. Metoda yang dimaksudkan adalah pencarian citra berdasarkan pada bentuk obyeknya. Teknik yang digunakan dalam pencarian citra ini adalah dengan menggunakan metoda *deformable template* (model yang dapat dibentuk kembali). Metoda ini dipilih karena merupakan teknik baru dalam pencarian obyek citra yang berdasarkan pada bentuk obyeknya.

Dengan menggunakan *deformable template* diharapkan bentuk obyek yang dapat dikenali adalah tidak harus sama persis akan tetapi dapat bervariasi dan masih dalam batas kemiripan sehingga tidak bersifat *rigid* (kaku). Untuk merepresentasikan model *deformable template* digunakan kurva *B-Spline*. Dengan

menggunakan sifat-sifat dari kurva *B-Spline* maka dapat dilakukan proses deformasi secara global seperti translasi, rotasi, dan perubahan skala dengan tidak merubah bentuk detil obyek dan deformasi lokal untuk merubah bentuk detil obyek. Dengan demikian, maka obyek yang dapat dikenali tidak dibatasi oleh posisi, orientasi dan perbedaan skala yang telah ditentukan.

## 1.2 Perumusan Masalah

Beberapa permasalan yang dapat dirumuskan dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana menggunakan *template* yang berupa citra *bitmap* yang menggambarkan tepi/kontur dari sebuah obyek dua dimensi agar dapat digunakan untuk mengidentifikasi semua obyek di dalam citra masukan.
2. Bagaimana agar obyek yang dapat dikenali tidak harus sama persis dengan *template* yang digunakan baik dalam hal posisi, orientasi, dan skala yang telah ditentukan.

## 1.3 Tujuan Penelitian

Tujuan dilaksanakannya penelitian ini adalah untuk:

1. Dapat mengidentifikasi semua obyek dalam sebuah citra yang memiliki kemiripan bentuk dengan *template* yang digunakan tanpa dibatasi oleh posisi, orientasi, dan skala yang telah ditentukan.

2. Mengeluarkan semua citra dari sekumpulan data citra dimana di dalam citra-citra tersebut terdapat obyek yang memiliki kemiripan bentuk dengan *template* yang digunakan.

## 1.4 Manfaat Penelitian

Dari penelitian yang dilakukan diharapkan dapat bermanfaat dalam proses identifikasi citra yang berdasarkan pada bentuk obyeknya. Proses identifikasi citra ini banyak digunakan dalam bidang kedokteran, biologi, dan bidang-bidang lainnya yang menggunakan data citra sebagai acuannya.

## 1.5 Batasan Masalah

Pembahasan dalam penelitian ini dibatasi pada beberapa masalah sebagai berikut:

1. Citra yang digunakan untuk merepresentasikan prototipe *template* adalah citra *bitmap* yang menyatakan tepi/kontur dari suatu obyek dua dimensi. Dimana tepi/kontur atau bentuk obyek tersebut hanya terdiri dari satu rangkaian kurva atau poligon baik terbuka maupun tertutup dan tidak harus terhubung secara baik.
2. *Deformable template* dibentuk dengan memberikan parameter *control point* (titik kontrol) *B-Spline* pada citra kontur/tepi (prototipe *template*). Kurva *B-Spline* yang digunakan adalah kurva *non rational* dengan fungsi basis *non uniform* dan *uniform*. Derajat kurva *B-Spline* dan jumlah *control point*

maksimum dibatasi masing-masing adalah 7 dan 100. Tidak ada penambahan ataupun pengurangan nilai *knot* dalam *knot vector*.

3. Perbedaan skala antara *deformable template* dengan obyek dalam citra yang dicari tidak terlalu besar (sekitar 0,7 sampai 1,3).
4. Ukuran obyek dalam citra tidak boleh terlalu kecil, karena diproses dalam tiga tingkatan resolusi.
5. Citra masukan tidak boleh banyak mengandung tekstur, karena dapat mempersulit proses pencocokan. Hal ini dikarenakan proses pencocokan dilakukan secara otomatis, dimana posisi awal *deformable template* tidak diletakkan di atas obyek yang dicari secara langsung.
6. Ukuran citra masukan maksimal adalah 640 x 480 satuan piksel.

## 1.6 Metodologi Penelitian

Metoda penelitian yang digunakan dalam penelitian ini meliputi:

1. Studi literatur, tentang:
   - *Deformable Template,*
   - *B-Spline,*
   - *Teori Bayesian,*
   - *Segmentasi Citra,*
   - *Generalized Hough Transform,* dan
   - *Teori Descent Gradient.*
2. Penentuan spesifikasi *deformable template* dan citra masukan.

3. Penentuan metoda, yaitu: Deteksi tepi citra dengan menggunakan *Canny Edge Detector* dan proses pencocokan dengan algoritma *Multiresolution Coarse-to-Fine*.
4. Perancangan dan pembuatan perangkat lunak.
5. Pengujian unjuk kerja perangkat lunak yang telah dibuat, sekaligus melakukan evaluasi dan koreksi yang diperlukan.
6. Penulisan laporan.

## 1.7 Sistematika Pembahasan

Selanjutnya pembahasan yang akan dilakukan dalam penelitian ini dapat dikelompokkan sebagai berikut:

**Bab 2** Menjelaskan tentang model deformasi *template* yang diawali dengan penjelasan secara singkat tentang hirarki pencocokan obyek dengan *template* mulai dari yang bersifat *rigid* (kaku) hingga yang bersifat *deformable* (dapat dibentuk kembali). Kemudian menjelaskan tentang representasi *deformable template* dan dilanjutkan dengan penjelasan tentang formulasi *Bayesian* dan fungsi obyektif.

**Bab 3** Menjelaskan tentang algoritma pencocokan yang dimulai dengan teori segmentasi citra, teori *Generalized Hough Transform*, teori *Descent Gradient*, dan algoritma *Multiresolution Coarse-to-Fine*.

**Bab 4** Membahas tentang perancangan dan pembuatan perangkat lunak yang dimulai dari perancangan data dan proses. Kemudian dilanjutkan dengan

pembuatan perangkat lunak dengan pembahasan beberapa kelas, prosedur, dan fungsi utama yang digunakan.

**Bab 5** Melaporkan tentang hasil uji coba dari perangkat lunak yang telah dibuat dan analisanya.

**Bab 6** Menyimpulkan pendekatan yang telah diajukan, hasil percobaan dan keterbatasan dari perangkat lunak yang telah dibuat. Juga diberikan beberapa catatan untuk pengembangan selanjutnya dari perangkat lunak yang telah dibuat.

# BAB II

# MODEL DEFORMASI TEMPLATE

# BAB II

# MODEL DEFORMASI TEMPLATE

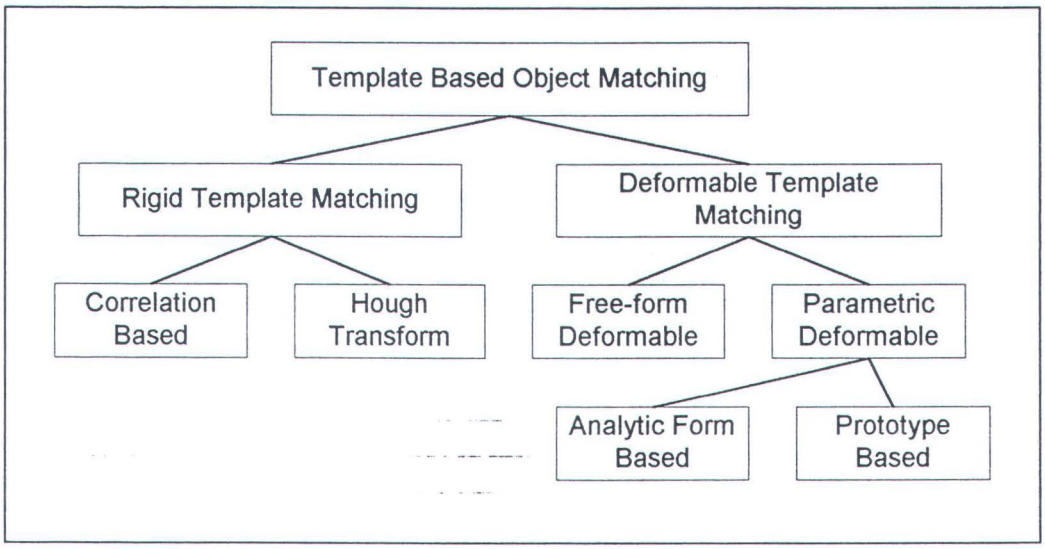
Pencocokan obyek dengan *template* adalah masalah yang telah banyak diaplikasikan dalam *computer vision* dan *image processing*. Aplikasi tersebut termasuk segmentasi citra, pencocokan obyek dan *tracking* (penelusuran), pengenalan dan interpretasi obyek, dan pencarian citra dari basis data citra. Pencocokan obyek dengan menggunakan *deformable template* merupakan hal yang baru dalam *image processing*. Oleh karena itu, penelitian ini difokuskan pada pencocokan obyek dengan menggunakan *deformable template*. Penelitian ini mengambil sebagian proses dalam penelitian yang telah dilakukan oleh Yu Zhong[1].

## 2.1 Hirarki Pencocokan Obyek dengan Template

Pencocokan obyek dengan menggunakan *template* dapat dikelompokkan dalam beberapa teknik (Gambar 2.1). Pada awalnya penelitian tentang pencocokan obyek dengan menggunakan *template* dikonsentrasikan pada *rigid template matching* (pencocokan *template* yang bersifat kaku), dimana bentuk yang cocok dapat diperoleh dengan melakukan transformasi sederhana seperti: translasi, rotasi, perubahan skala, dan transformasi yang terikat terhadap model *template*. Contoh *rigid template matching* adalah *correlation-based matching* dan *Hough Transform*

[1] Y. Zhong, *Object Matching Using Deformable Templates,* PhD. Thesis, Department of Computer Science, Michigan State University, May 1997.

(HT). Karena terdapat kekakuan pada pendekatan ini, maka penggunaannya terbatas pula. Suatu teknik yang lebih baik dalam menangani keterbatasan ini adalah teknik *deformable template matching*. Teknik pencocokan dengan *deformable template matching* bersifat lebih dapat diubah-ubah dan fleksibel dibandingkan dengan *rigid shape matching*. Sehingga lebih baik untuk mengatasi bentuk obyek yang bermacam-macam variasinya. Metoda *deformable template matching* dapat dikelompokkan menjadi dua kelas utama, yaitu: (i) *free-form*, dan (ii) *parametric*. Pengelompokan ini berdasarkan pada apakah *template* diparameterkan atau tidak. Metoda *parametric deformable template* selanjutnya dikelompokkan menjadi dua, yaitu: *analytical form-based* dan *prototype-based*. Di dalam *analytical form-based* bentuk *template* diidentifikasikan dengan rumus analitis, sedangkan dalam *prototype-based* bentuk *template* diperoleh dengan melakukan transformasi secara parametrik dari prototipe *template*.




Gambar 2.1: Hirarki teknik pencocokan obyek dengan menggunakan *template*.

Penelitian ini termasuk dalam *parametric deformable template matching* dengan menggunakan prototipe *template*. Parameterisasi prototipe *template* yang digunakan adalah dengan properti kurva *B-Spline*.

## 2.2 Representasi Deformable Template

Dalam penelitian ini, untuk membentuk suatu *derformable template* digunakan prototipe *template* yang berupa sketsa kontur/tepi yang sederhana (hanya satu rangkaian kurva/poligon) dalam bentuk *bitmap*. Untuk mengubah *bitmap* prototipe *template* tersebut menjadi parameter yang dapat dideformasi, maka digunakan transformasi deformasi dengan *B-Spline* dimana set *control point* merupakan parameter deformasi. Untuk membatasi deformasi dari *deformable template*, maka digunakan suatu model probabilistik deformasi.

### 2.2.1 Prototipe Template

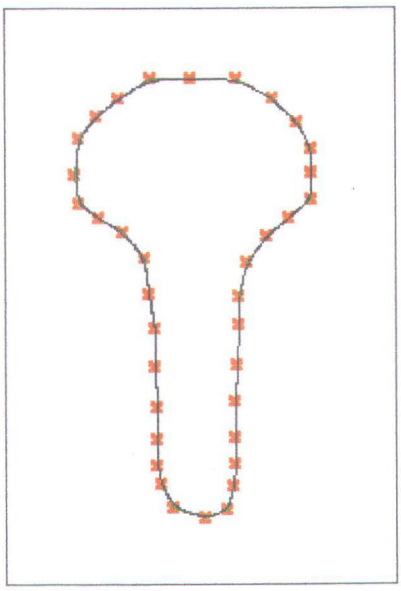
Untuk merepresentasikan prototipe *template* digunakan suatu citra yang terdiri dari set titik-titik yang menggambarkan tepi/kontur dua dimensi dari suatu obyek. Sifat kontur yang digunakan dapat merupakan kontur terbuka maupun tertutup yang tidak harus terhubung dengan baik. Untuk menyatakan bahwa titik-titik tersebut merupakan obyek, maka kami menetapkan titik-titik ini berwarna kontras dengan warna latar belakangnya (misalnya: warna kontur hitam dan warna latar putih).

MILIK PERPUSTAKAAN ITS



### 2.2.2 Transformasi Deformasi dengan B-Spline

Transformasi bertujuan untuk membuat prototipe *template* menjadi parameter-parameter yang dapat dideformasi. Transformasi yang digunakan untuk membentuk *deformable template* adalah transformasi dengan fungsi basis *B-Spline*. Parameter-parameter deformasi dari *deformable template* diwakili oleh set *control point* (titik kontrol) pada kurva *B-Spline*. *Control point* diset pada sekitar kontur dalam prototipe *template* (Gambar 2.2).



Gambar 2.2: Prototipe *template* dengan parameter-parameternya (tanda silang).

**B-Spline**[2]

Dalam penelitian ini bentuk kurva yang digunakan adalah bentuk *non rational* dengan fungsi basis *B-Spline* yang *uniform* maupun *non uniform* dan tanpa adanya pengurangan ataupun penambahan nilai *knot* dalam *knot vector*.

---

[2] M. E. Mortenson, *Geometric Modelling*, Second Edition, Wiley Computer Publishing, 1997, 113-137.
D. Hearn and M. P. Baker, *Computer Graphics*, Second Edition, Prentice-Hall International, Inc., 1994, 334-344.

Secara umum persamaan kurva *B-Spline* dengan bentuk *non rational* adalah:

$$\mathbf{P}(u) = \sum_{i=0}^{n} p_i N_{i,k}(u) \qquad (2\text{-}1)$$

dimana $p_i$ adalah set $(n + 1)$ titik kontrol, $N_{i,k}$ adalah fungsi basis *B-Spline* dengan derajat $(k - 1)$.

Sedangkan fungsi basis kurva *B-Spline non uniform* didefinisikan oleh rumus rekursif Cox-deBoor sebagai berikut:

$$N_{i,1}(u) = 1 \quad \text{jika} \quad t_i \le u < t_{i+1}$$

$$= 0 \quad \text{untuk } u \text{ lainnya}$$

dan $$N_{i,k}(u) = \frac{u - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(u) + \frac{t_{i+k} - u}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(u) \qquad (2\text{-}2)$$

dimana $t_i$ adalah nilai *knot vector* yang diperoleh dengan aturan berikut ini:

$$t_i = 0 \quad \text{jika} \quad i < k$$

$$t_i = i - k + 1 \quad \text{jika} \quad k \le i \le n$$

$$t_i = n - k + 2 \quad \text{jika} \quad i > n \qquad (2\text{-}3)$$

dengan $0 \le i \le n + k$.

Banyaknya *knot* dalam *knot vector* harus memenuhi persamaan berikut ini:

$$n + k + 1 = T \qquad (2\text{-}4)$$

dimana $T$ adalah jumlah *knot*. Dengan menggunakan Persamaan (2-3), maka *knot vector* **T** dapat dikarakteristikkan sebagai berikut:

$$\mathbf{T} = \{\alpha, \alpha, \ldots, \alpha, t_k, \ldots t_{T-k-1}, \beta, \beta, \ldots \beta\}$$

dimana $\alpha$ dan $\beta$ diulang sebanyak $k$ dan sesuai dengan aturan pengisian nilai *knot*, maka nilai $\alpha = 0$ dan $\beta = n - k + 2$. Dan range variabel parametrik $u$ adalah:

$$0 \le u \le n - k + 2 \text{ atau } \mathbf{T}_k \le u \le \mathbf{T}_{T-k} \qquad (2\text{-}5)$$

Karena nilai pembagi dalam Persamaan (2-2) bisa nol, maka harus didefinisikan untuk $0/0 = 0$.

Aturan-aturan di atas dapat digunakan untuk membuat kurva *B-Spline non uniform* terbuka dengan derajat maksimal $n$. Untuk membuat kurva tertutup dapat dilakukan dengan cara memodifikasi titik terakhir dan *knot vector*-nya. Titik terakhir dimodifikasi sehingga sama dengan titik awalnya. Derajat maksimal yang diperbolehkan dalam pembentukan kurva tertutup ini adalah $n+1$.

Untuk membuat kurva *B-Spline uniform* dapat dilakukan dengan cara memodifikasi nilai *knot* yang ada pada *knot vector* sehingga perubahan setiap nilainya menjadi *uniform*. Berikut ini adalah perubahan nilai *knot vector* yang harus dilakukan:

$$t_i = i \qquad \text{dimana} \qquad 0 \le i \le n + k \qquad (2\text{-}6)$$

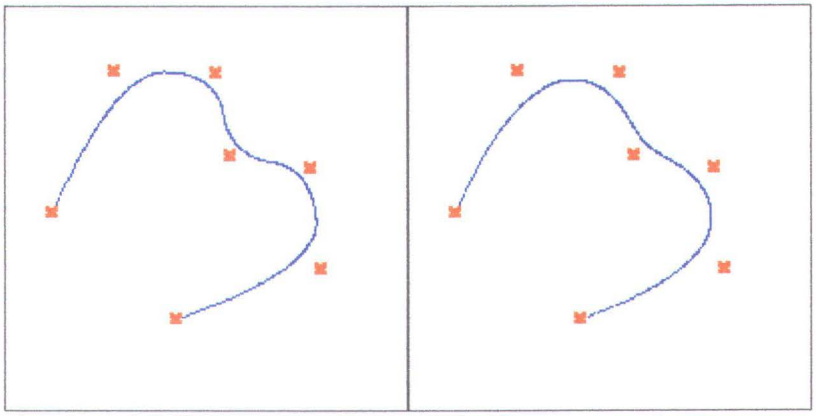
Dengan menggunakan nilai *knot* yang baru ini dapat dibuat kurva *B-Spline uniform* terbuka dengan derajat maksimum adalah $n$. Untuk membuat kurva tertutup maka harus dijamin bahwa titik-titik yang diinterpolasi setelah $i = n$ adalah merupakan perulangan dari titik awalnya (modulus $(n+1)$).

Selain dapat digunakan untuk membuat kurva dengan derajat maksimum adalah $n$, pada prinsipnya juga dapat digunakan untuk membuat kurva dengan

derajat lebih besar daripada $n$. Akan tetapi hal ini hanya dapat dilakukan pada kurva tertutup dengan fungsi basis yang *uniform* saja. Untuk keperluan tersebut maka nilai *knot* dalam *knot vector* harus dikembangkan menjadi sebagai berikut:

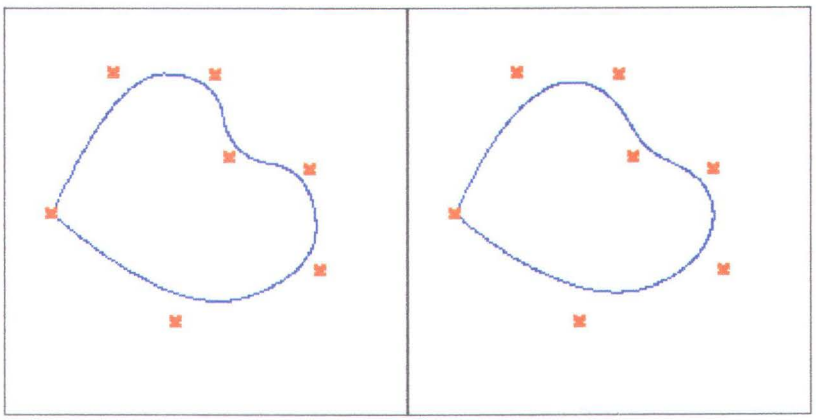
$$t_i = i \qquad \text{dimana} \qquad 0 \le i \le n + 2k - 2 \tag{2-7}$$

Gambar 2.3 – 2.6 berikut ini menunjukkan beberapa sifat dari kurva *B-Spline non rational* baik *non uniform* maupun *uniform* yang terbuka maupun tertutup.



(a) (b)

Gambar 2.3: Kurva *B-Spline non rational non uniform* terbuka.
(a) n = 6 dan k = 3. (b) n = 6 dan k = 4.

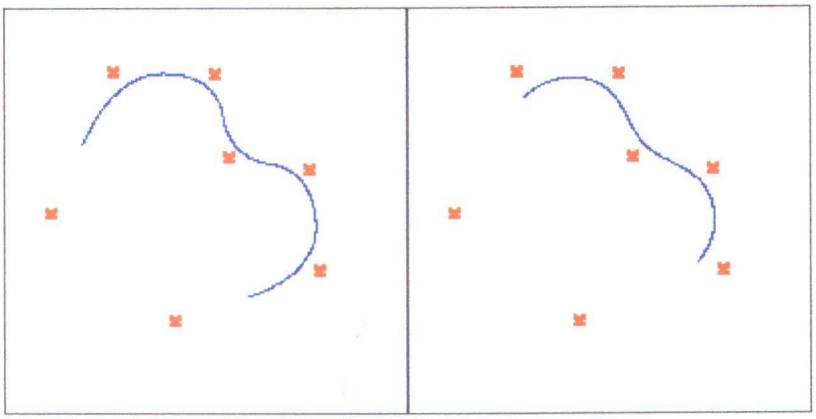


(a) (b)

Gambar 2.4: Kurva *B-Spline non rational non uniform* tertutup.
(a) n = 6 dan k = 3. (b) n = 6 dan k = 4.

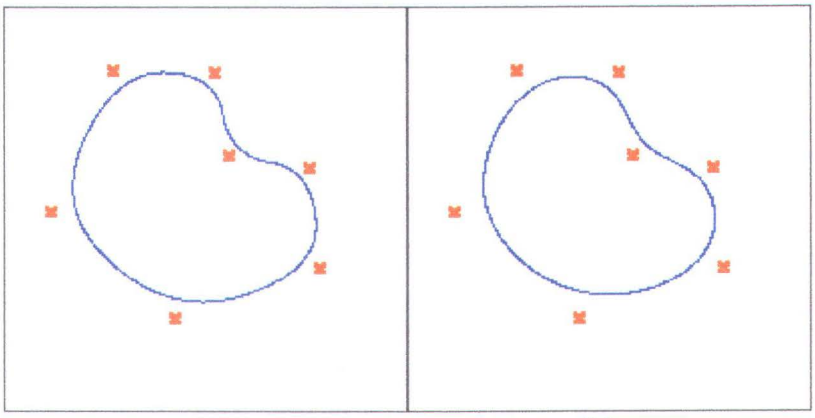
PERPUSTAKAAN ITS





(a) (b)

Gambar 2.5: Kurva *B-Spline non rational uniform* terbuka.
(a) n = 6 dan k = 3. (b) n = 6 dan k = 4.



(a) (b)

Gambar 2.6: Kurva *B-Spline non rational uniform* tertutup.
(a) n = 6 dan k = 3. (b) n = 6 dan k = 4.

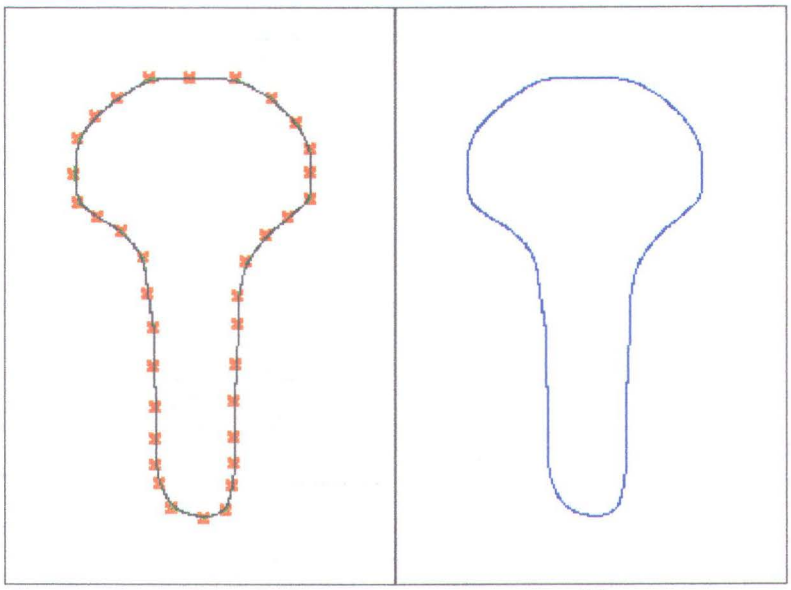
**Representasi Deformable Template dengan B-Spline**

Untuk merepresentasikan kontur dua dimensi, maka penghitungan koordinat $x$ dan $y$ dengan fungsi basis *B-Spline* dilakukan secara terpisah sebagai berikut:

$$[f_x(u) f_y(u)] = \sum_{i=0}^{n-k+1} p_i N_{i,k}(u) \qquad (2\text{-}8)$$

Dimana variabel-variabel yang sama disesuaikan dengan aturan diatas (Persamaan (2-1) – (2-7)).

Dengan proses transformasi *control point* (Gambar 2.2) dengan fungsi basis *B-Spline* (Persamaan (2-8)), maka dapat diperoleh kembali bentuk rata-rata dari prototipe *template* tersebut. Gambar 2.7 menunjukkan hasil transformasi parameter-parameter dari prototipe *template* dengan menggunakan fungsi basis *B-Spline*.



(a) (b)

Gambar 2.7: Representasi *deformable template*. (a) Prototipe *template* dengan parameter-parameternya (tanda silang). (b) Bentuk tepi/kontur hasil transformasi dengan menggunakan fungsi basis *B-Spline non rational uniform* tertutup dengan $k = 3$.

**Transformasi Deformasi**

Jika *default deformable template* dapat dinyatakan dengan:

$$[x_0(u) y_0(u)] = \sum_{i=0}^{n-k+1} p_{0i} N_{i,k}(u) \qquad (2\text{-}9)$$

dan deviasi titik kontrol dari defaultnya adalah $\Delta p_i = \left[\xi_i^x \xi_i^y\right]$, maka deformasi *deformable template* dapat dinyatakan sebagai berikut:

$$\left[x(u)y(u)\right] = \sum_{i=0}^{n-k+1} p_i N_{i,k}(u) = \sum_{i=0}^{n-k+1} (p_{0i} + \Delta p_i) N_{i,k}(u) \tag{2-10}$$

Dan fungsi transformasi deformasi dapat disamakan dengan deformasi menggunakan fungsi basis *B-Spline*, sehingga dapat dirumuskan sebagai berikut:

$$D_\xi = \sum_{i=0}^{n-k+1} \left(\left[\xi_i^x \xi_i^y\right]\right) N_{i,k}(u) \tag{2-11}$$

### 2.2.3 Model Probabilistik Deformasi

Model probabilistik deformasi merupakan informasi khusus yang dimiliki oleh *deformable template* untuk menyatakan nilai deformasi *deformable template* dari bentuk awalnya. Berikut ini adalah model probabilistik deformasi yang digunakan:

$$\begin{aligned}
\Pr\left(\xi\right) &= \prod_{i=0}^{N_c-1} \Pr\left(\xi_i\right) \\
&= \prod_{i=0}^{M,N} \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{\left(\xi_i^x\right)^2 + \left(\xi_i^y\right)^2}{2\sigma^2}\right\} \\
&= \frac{1}{\left(2\pi\sigma^2\right)^{N_c}} \exp\left\{-\frac{1}{2\sigma^2} \sum_i \left(\left(\xi_i^x\right)^2 + \left(\xi_i^y\right)^2\right)\right\}
\end{aligned} \tag{2-12}$$

dimana $\left(\xi_i^x, \xi_i^y\right)$ adalah deviasi *control point* dari defaultnya. Sedangkan $\sigma$ adalah deviasi standar, semakin besar nilainya menyebabkan *deformable template* lebih elastis dan sebaliknya.

## 2.3 Formulasi Bayesian dan Fungsi Obyektif

Skema *Bayesian* menyatukan informasi yang terdapat dalam *deformable template* (*prior*) dengan informasi dari obyek yang diamati dalam citra (*likelihood*). Informasi *prior* dari *deformable template* terdiri dari:

1. Prototipe *template*,
2. Set transformasi deformasi *template*, dan
3. Probabilitas densitas pada set transformasi deformasi.

Sedangkan informasi *likelihood* merupakan fungsi energi yang dihasilkan oleh tepi obyek dalam citra dengan *deformable template*. Informasi *likelihood* ini selanjutnya dipadukan dengan informasi *prior* dengan menggunakan aturan *Bayes* untuk memperoleh probabilitas densitas *posterior* deformasi *deformable template* terhadap obyek yang ada dalam citra masukan. Tujuan dari aturan *Bayes* ini adalah memaksimalkan probabilitas densitas *posterior*. Probabilitas densitas *posterior* maksimal ketika *deformable template* cocok dengan obyek yang ada dalam citra masukan.

### 2.3.1 Distribusi Prior

Distribusi *prior* digunakan untuk memperkirakan transformasi global (translasi, rotasi, dan perubahan skala) dan deformasi lokal yang terjadi pada *deformable template*. Jika $T_0$ menyatakan prototipe *template*, dan $T_{s,\theta,\xi,d}$ merupakan deformasi *deformable template*, maka deformasi yang dilakukan dengan merotasi *deformable template* dengan sudut $\theta$, kemudian deformasi lokal

oleh set parameter $\xi$, dilanjutkan dengan perubahan skala dengan faktor $s$, dan translasi dengan arah $x$ dan $y$ sejauh $d = (d^x, d^y)$ adalah:

$$T_{s,\theta,\xi,d}(x,y) = T_0\left(s.\left[(x,y) + D_\xi(R_\theta(x,y))\right] + (d^x, d^y)\right) \quad (2\text{-}13)$$

Dimana $D_\xi$ menyatakan fungsi deformasi set titik-titik kontrol dari posisi asalnya (Persamaan (2-11)) dan $R_\theta(x,y)$ adalah rotasi titik $(x,y)$ dengan sudut $\theta$. Jika dianggap bahwa $s, \theta, \xi$, dan $d$ adalah variabel bebas, maka:

$$\Pr(s,\theta,\xi,d) = \Pr(s).\Pr(\theta).\Pr(\xi).\Pr(d) \quad (2\text{-}14)$$

Dan jika translasi, rotasi, dan perubahan skala dianggap sama probabilitasnya selama dalam proses pencocokan, maka Persamaan (2-14) dapat disederhanakan menjadi:

$$\Pr(s,\theta,\xi,d) = \kappa \Pr(\xi) \quad (2\text{-}15)$$

Dimana $\kappa$, adalah konstanta normalisasi dan $\Pr(\xi)$ didefinisikan oleh Persamaan (2-12).

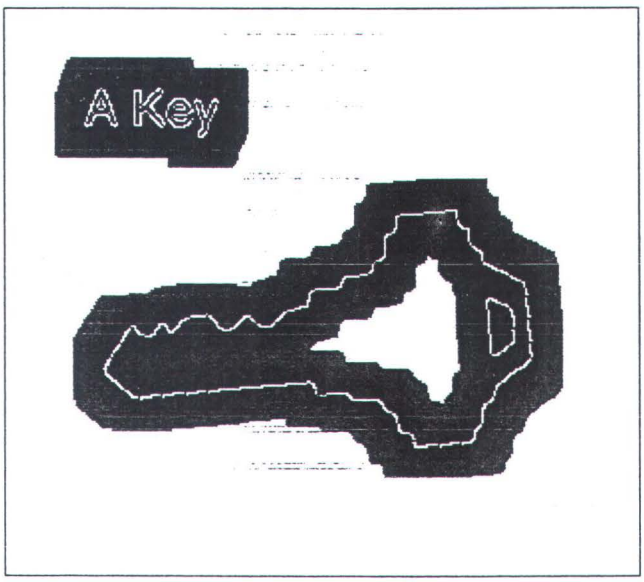
### 2.3.2 Likelihood

*Likelihood* menyatakan probabilitas pada data citra masukan. *Likelihood* menunjukkan ukuran kemiripan antara *deformable template* dengan obyek yang diamati dalam citra masukan. *Likelihood* dalam penelitian ini hanya menyatukan informasi *edge* (tepi) pada citra masukan, dimana kedua posisi tepi dan arahnya diperhatikan.

*Deformable template* ditarik dan diatur ke arah tepi obyek melalui suatu medan potensial yang digabungkan dengan informasi arah tepi dalam citra masukan. Untuk suatu titik dalam citra masukan, medan potensialnya adalah:

$$\Phi(x,y) = -\exp\left\{-\rho\left(\delta_x^2+\delta_y^2\right)^{1/2}\right\} \qquad (2\text{-}16)$$

dimana $(\delta_x, \delta_y)$ adalah jarak perpindahan suatu titik ke tepi obyek terdekat dalam citra masukan, dan $\rho$ adalah faktor kehalusan dari medan potensial. Untuk titik-titik yang jauh dari titik tepi maka potensialnya tinggi dan untuk titik-titik yang dekat dengan titik tepi maka potensialnya rendah (Gambar 2.8). Tujuan medan potensial ini adalah untuk menarik *deformable template* mendekati tepi obyek yang sebenarnya.



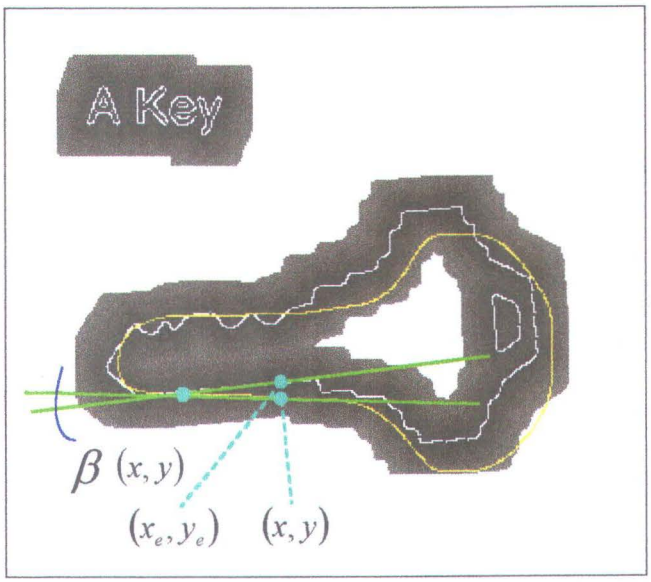

Gambar 2.8: Medan potensial citra tepi masukan. Nilai potensial ditunjukkan oleh nilai *grayscale*-nya, semakin besar nilai *grayscale* semakin besar potensialnya.

Untuk meningkatkan performa agar *deformable template* tidak terjebak oleh tepi obyek yang palsu maka digunakan informasi tambahan yang merupakan

komponen arah dari tepi obyek. Medan potensial yang baru dengan menggunakan informasi arah dapat dirumuskan sebagai berikut:

$$E(T_{s,\theta,\xi,d}, Y) = \frac{1}{n_T} \sum \left( 1 + \Phi(x,y) \left| \cos\left( \beta\,(x,y) \right) \right| \right) \qquad (2\text{-}17)$$

dimana $n_T$ adalah jumlah titik pada *deformable template*, $\beta\,(x,y)$ adalah sudut antara arah tangent pada *deformable template* dengan arah tepi titik terdekat pada obyek di titik $(x,y)$ (Gambar 2.9), dan konstanta 1 ditambahkan sehingga energi potensialnya positif dan bernilai antara 0 dan 1.




Gambar 2.9: *Deformable template* diletakkan dalam medan potensial. $(x,y)$ adalah sebuah titik pada *deformable template*. $(x_e, y_e)$ adalah titik tepi terdekat dari $(x,y)$. $\beta\,(x,y)$ adalah sudut yang dibentuk antara $(x,y)$ dengan $(x_e, y_e)$.

Dengan menggunakan komponen arah maka *deformable template* diletakkan tidak hanya dalam posisi tetapi juga orientasi yang tepat. Probabilitas

densitas *likelihood* pada suatu citra masukan karena pengaruh *deformable template* adalah:

$$\mathbf{Pr}(Y \mid s,\theta,\xi,d) = \alpha \exp\{-\mathrm{E}(T_{s,\theta,\xi,d}, Y)\} \tag{2-18}$$

dimana $\alpha$ adalah konstanta normalisasi..

### 2.3.3 Probabilitas Densitas Posterior

Dengan menggunakan aturan *Bayes*, probabilitas *prior* pada *deformable template* dan *likelihood* pada citra masukan karena pengaruh *deformable template* dapat dipadukan untuk memperoleh suatu probabilitas densitas *posterior* pada *deformable template* terhadap citra masukan. Probabilitas densitas *posterior* tersebut dapat dirumuskan sebagai berikut:

$$\begin{aligned}\Pr(s,\theta,\xi,d|Y) &= \Pr(Y|s,\theta,\xi,d)\Pr(s,\theta,\xi,d) / \Pr(Y) \\ &= C_1 \exp\{-\mathrm{E}(T_{s,\theta,\xi,d}, Y)\}\prod_{i\in P}\tfrac{1}{2\pi\sigma^2}\exp\left\{-\tfrac{1}{2\sigma^2}\left(\xi_i^{x2}+\xi_i^{y2}\right)\right\} \\ &= C_2 \exp\left\{-\mathrm{E}(T_{s,\theta,\xi,d}, Y) - \sum_{i\in P}\tfrac{1}{2\sigma^2}\left(\xi_i^{x2}+\xi_i^{y2}\right)\right\} \qquad (2\text{-}19)\end{aligned}$$

dimana $C_1$ dan $C_2$ adalah konstanta normalisasi, dan $P$ adalah set parameter deformasi.

### 2.3.4 Fungsi Obyektif

Nilai obyektif merupakan ukuran kecocokan antara *deformable template* dengan obyek yang ditempatinya. Selanjutnya, nilai obyektif dapat diperoleh

dengan mengacu pada fungsi densitas *posterior* sebelumnnya. Dengan memberikan logaritma natural (Lon) pada kedua sisi pada Persamaan (2-19) menghasilkan:

$$\Pr(s,\theta,\xi,d \mid Y) = \ln C_2 - \mathrm{E}(T_{s,\theta,\xi,d},Y) - \sum_{i\in P}\frac{1}{2\sigma^2}\left(\xi_i^{x2} + \xi_i^{y2}\right) \qquad (2\text{-}20)$$

Minimasi fungsi probabilitas densitas *posterior* pada Persamaan (2-20) menjadi:

$$L(s,\theta,\xi,d \mid Y) = \mathrm{E}(T_{s,\theta,\xi,d},Y) + \gamma\sum_{i\in P}\left(\xi_i^{x2} + \xi_i^{y2}\right) \qquad (2\text{-}21)$$

dimana $\gamma = \frac{1}{2\sigma^2}$. Fungsi obyektif ini terdiri dari dua bagian:

- Istilah *model-based* yang menyatakan ukuran deviasi *deformable template*, dan
- Istilah *data-driven* yang menyatakan ukuran kebaikan *deformable template* terhadap obyek dalam citra masukan.

# BAB III

# PROSES PENCOCOKAN OBYEK

# BAB III

# PROSES PENCOCOKAN OBYEK

Proses pencocokan *deformable template* dengan obyek yang ada pada citra masukan dilakukan secara otomatis dimana posisi awal *deformable template* adalah di sudut kiri atas (x:0, y:0). Sebelum proses pencocokan, maka terlebih dahulu obyek di dalam citra masukan harus dipisahkan dari latar belakangnya. Untuk memisahkan obyek dari latar belakangnya ini digunakan *Canny Edge Detector* (detektor tepi *Canny*). Setelah obyek dapat dipisahkan dari latarnya, maka langkah selanjutnya adalah menarik *deformable template* ke arah obyek tersebut. Untuk menarik *deformable template* ke arah obyek dilakukan dengan proses translasi dan rotasi dengan menggunakan informasi medan potensial dan arah tepi (seperti dijelaskan pada bab sebelumnya). Setelah obyek terletak pada medan potensial tepi, maka proses selanjutnya adalah deformasi global untuk menempatkan *deformable template* dalam posisi, orientasi, dan ukuran yang lebih sesuai dengan obyek yang ditempatinya. Untuk keperluan ini, maka digunakan algoritma *Generalized Hough Transform* (GHT). Proses selanjutnya adalah deformasi *deformable template* secara lokal ke arah tepi obyek terdekat dengan menggunakan algoritma *Descent Gradient* untuk meningkatkan kecocokan bentuknya. Untuk mempercepat proses pencocokan secara keseluruhan, maka digunakan algoritma *Multiresolution Coarse-to-Fine*.

## 3.1 Segmentasi Citra

Segmentasi citra bertujuan untuk mendapatkan citra tepi dari suatu citra masukan. Dalam penelitian ini digunakan *Canny Edge Detector*[3] (detektor tepi *Canny*). *Canny* mengusulkan suatu kriteria untuk mendeteksi tepi. Kriteria ini terdiri atas tiga bagian, yaitu:

1. *Good detection* (deteksinya baik)

   Kriteria ini menyatakan bahwa probabilitas kegagalan untuk menandai titik-titik tepi yang sebenarnya adalah kecil, dan probabilitas titik-titik bukan tepi yang tertandai sebagai tepi adalah kecil.

2. *Good localization* (lokalisasinya baik)

   Titik-titik yang ditandai sebagai titik-titik tepi seharusnya sedekat mungkin ke pusat tepi yang sebenarnya.

3. *Low spurious response* (sedikit respon yang salah)

   Kriteria ketiga ini menyatakan bahwa hanya ada satu respon untuk satu tepi. Bila ada dua respon untuk satu tepi, maka satu diantaranya adalah salah.

Dari ketiga kriteria tersebut *Canny* memberikan kebebasan untuk mengatur hasil tepi yang terdeteksi. Tepi yang terdeteksi sangatlah tergantung pada beberapa faktor yang ada diantaranya adalah:

---

[3] J. F. Canny, *Finding Edges and Lines in Images*, MSc. Thesis, Massachusetts Institute of Technology, June 1983.
J. F. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, November 1986.

- *Sigma* ($\sigma$)

  *Sigma* adalah faktor penentu untuk membuat tepi yang terdeteksi semakin baik atau tidak, karena *sigma* mengatur faktor bobot *kernel* yang digunakan dalam proses konvolusi dengan citra masukan.

- *High Threshold* (*HighTh*)

  Nilai *high threshold* digunakan untuk membatasi nilai *magnitude* yang merupakan tepi. Batas atas nilai *magnitude* dihitung dari jumlah titik *magnitude* yang tidak nol dan memiliki nilai terbesar dikalikan dengan nilai *high threshold*. Semua titik yang nilai *magnitude*-nya di atas batas atas nilai *magnitude* adalah merupakan tepi. Semakin besar nilai *high threshold* maka semakin sedikit tepi yang terdeteksi.

- *Low Threshold* (*LowTh*)

  *Low threshold* merupakan nilai yang digunakan untuk membatasi nilai *magnitude* terendah suatu titik agar titik tersebut masih disebut tepi. Nilai terendah *magnitude* dihitung dari nilai *magnitude* yang merupakan batas tertinggi dari *high threshold* dikalikan *low threshold*. *Canny* menganjurkan untuk memilih nilai *low threshold* ini adalah sepertiga sampai setengah dari batas atasnya.

Berikut ini adalah salah satu implementasi dengan beberapa urutan proses yang dapat digunakan untuk memenuhi ketiga kriteria *Canny* tersebut:

1. *Konvolusi Gaussian*

   Proses ini merupakan proses penghalusan citra yang bertujuan untuk mengurangi detil tepi dan *noise*. Dalam proses ini, citra dikonvolusikan dengan menggunakan *Gaussian kernel* dengan rumus sebagai berikut:

$$G(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3-1}$$

2. *Konvolusi First Derivative Gaussian*

   Tujuan dari proses ini adalah untuk mendapatkan gradien x dan gradien y. Berikut ini adalah rumus *First Derivative Gaussian* yang digunakan:

$$G'(x) = -\frac{x}{\sigma^2}\exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3-2}$$

3. *Hitung Magnitude*

   Pada proses ini, nilai *magnitude* dihitung dari gradien x dan gradien y yang dihasilkan oleh proses sebelumnya (2). Adapun rumus yang dapat digunakan adalah sebagai berikut:

$$Mag(x,y) = |\nabla x(x,y)| + |\nabla y(x,y)|$$

   atau
$$Mag(x,y) = \left\{\nabla x(x,y)^2 + \nabla y(x,y)^2\right\}^{\frac{1}{2}} \tag{3-3}$$

4. *Hitung Direction*

   Proses ini merupakan proses untuk penghitungan arah sudut suatu titik sesuai dengan titik tepi terdekat dengan menggunakan gradien x dan gradien y. Rumus yang digunakan adalah:

$$tg(\theta) = \frac{\nabla y}{\nabla x} \tag{3-4}$$

ITS



5. *Proses Non Maximum Suppression*

*Non Maximum Suppression* bertujuan untuk menghilangkan titik-titik tepi yang tidak maksimal. Titik-titik tepi yang tidak maksimal dapat ditekan / dihilangkan dengan menggunakan rumus sebagai berikut:

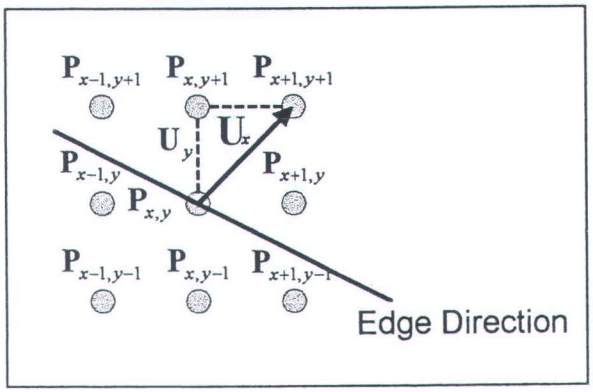
Titik $P_{x,y}$ (Gambar 3.1) ditandai sebagai titik maksimum apabila memenuhi kriteria:

$$G(x,y) > G_1 \text{ dan } G(x,y) > G_2, \tag{3-5}$$

dimana: $G_1 = \frac{U_x}{U_y} G(x+1,y+1) + \frac{U_y - U_x}{U_y} G(x,y+1)$ dan

$$G_2 = \frac{U_x}{U_y} G(x-1,y-1) + \frac{U_y - U_x}{U_y} G(x,y-1)$$

$G$ adalah *magnitude*.



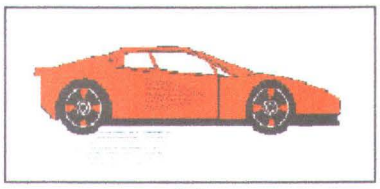

Gambar 3.1: *Non Maximum Suppression.*

6. *Proses Hysteresis*

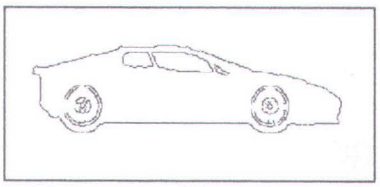
Proses *hysteresis* bertujuan untuk mengeset semua titik yang maksimum menjadi tepi atau bukan. Pengesetan dimulai dari titik maksimum yang nilai *magnitude*-nya di atas batas atas *magnitude*. Jika nilainya di atas batas atas nilai *magnitude*

yang diijinkan, maka titik tersebut diset menjadi tepi. Proses selanjutnya adalah dengan menelusuri dari titik tepi tersebut ke semua titik yang maksimum. Jika nilai *magnitude*-nya diatas batas bawah nilai *magnitude* yang diijinkan, maka titik-titik tersebut juga diset menjadi tepi. Proses yang sama dilakukan untuk setiap titik yang maksimum.

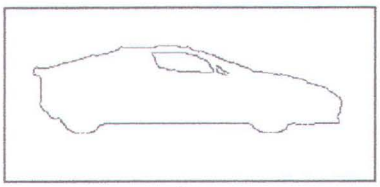
Berikut ini adalah beberapa contoh citra tepi yang dihasilkan oleh *Canny Edge Detector* (Gambar 3.2(b)-3.2(e)) dari citra asal pada Gambar 3.2(a):



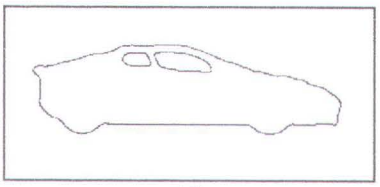
(a)



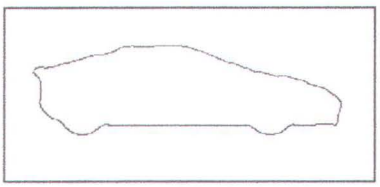
(b)



(c)



(d)



(e)

Gambar 3.2: Deteksi tepi *Canny*: (a) Citra asal.
(b) Citra tepi ( $\sigma = 1$, *HighTh* = 0,7, *LowTh* = 0,5, dan ukuran jendela = 9x9).
(c) Citra tepi ( $\sigma = 1$, *HighTh* = 0,9, *LowTh* = 0,5, dan ukuran jendela = 9x9).
(d) Citra tepi ( $\sigma = 3$, *HighTh* = 0,7, *LowTh* = 0,5, dan ukuran jendela = 9x9).
(e) Citra tepi ( $\sigma = 3$, *HighTh* = 0,9, *LowTh* = 0,5, dan ukuran jendela = 9x9).

Dari Gambar 3.2(b)-3.2(e) terlihat bahwa dengan menggunakan beberapa parameter yang berbeda ( $\sigma$ dan *HighTh*) saja dapat menghasilkan citra tepi yang

berbeda. Hal ini belum termasuk menggunakan parameter-parameter lainnya yang berbeda selain parameter tersebut, tentunya akan menghasilkan citra tepi yang juga berbeda pula. Untuk itu harus ditentukan nilai parameter-parameter tersebut sesuai dengan citra masukan yang dideteksi tepinya dan juga citra tepi yang ingin dihasilkannya.

## 3.2 Transformasi Hough yang Dibakukan

Transformasi *Hough* yang dibakukan / *Generalized Hough Transform* (GHT) merupakan pengembangan teknik dasar pencarian suatu obyek garis dalam citra yang dikemukakan oleh *Hough*. Dalam perkembangan selanjutnya, teknik ini tidak hanya untuk mendeteksi garis saja, akan tetapi juga berupa kurva. *Generalized Hough Transform* yang digunakan disini adalah versi dari *Ballard*. Dimana tujuan digunakannya *Generalized Hough Transform* adalah untuk menempatkan *deformable template* pada posisi, orientasi, dan ukuran yang sesuai dengan obyek yang ditempatinya. Untuk mendapatkan kemungkinan posisi, orientasi, dan ukuran yang sesuai maka dibuat suatu *R- table*.

Berikut ini adalah algoritma *Generalized Hough Transform* (GHT) yang digunakan:

- Buat *R-table* berdasarkan *deformable template*: Setiap *entry* diindeks dengan sudut tangent $\phi$ dari setiap titik pada *deformable template*. Dimana setiap entry terdiri dari pasangan $(r_{ij}, \alpha_{ij})$, $r_{ij}$ adalah jarak titik pada

*deformable template* ke pusat $(x_c, y_c)$ dan $\alpha_{ij}$ adalah sudut yang dibentuk oleh titik pada *deformable template* terhadap titik pusatnya.

- Inisialisasi *accumulator array* dengan nol:

  $A(x_{c\min}:x_{c\max}, y_{c\min}:y_{c\max}, s_{\min}:s_{\max}, \theta_{\min}:\theta_{\max})$

  Parameter-parameter *accumulator* tersebut adalah titik pusat $(x, y)$, skala $s$, dan sudut rotasi $\theta$.

- Untuk setiap titik $\mathbf{x} = (x, y)$ dalam citra masukan, lakukan:

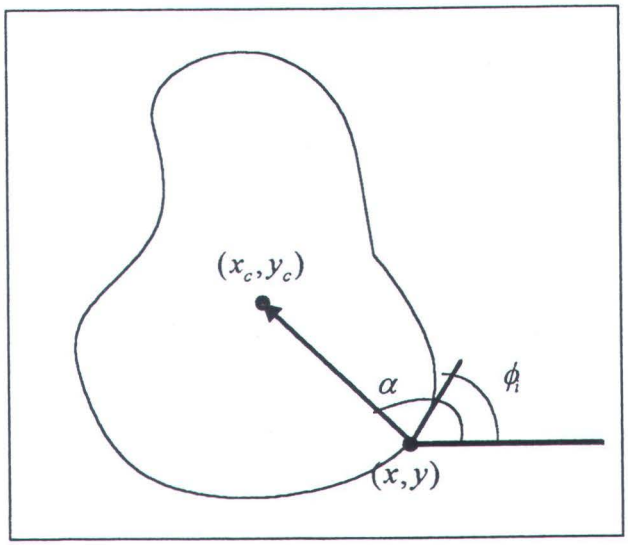
  - Hitung sudut tangent $\phi$ (**x**).
  - Lihat $\phi$ (**x**) pada *R-table* dan diperoleh $(r, \alpha)$.
  - Hitung kemungkinan pusat, skala dan sudut rotasi untuk setiap pasangan $(r, \alpha)$ dan setiap kombinasi $(s, \theta)$:

    $$(x_c, y_c) = (x, y) + r(\phi)s(\cos(\alpha + \theta), \sin(\alpha + \theta))$$

  - Masukkan nilai $(x_c, y_c, s, \theta)$ ke dalam *accumulator array*.

- Kemungkinan posisi pusat, skala, dan rotasi diperoleh dari nilai masing-masing dalam *accumulator array* dengan frekuensi terbanyak.

Gambar 3.3 menunjukkan konstruksi *R-table* dengan *Generalized Hough Transform*.




Gambar 3.3: Konstruksi *R-table* dengan *Generalized Hough Transform*.

## 3.3 Metoda Gradien Menurun

Metoda *Descent Gradient*[4] (gradien menurun) digunakan untuk mengupdate set *control point* pada *deformable template* secara lokal sehingga dapat menyesuaikan bentuk dengan obyek yang ditempatinya. *Deformable template* diupdate dari posisi semula dengan menuruni gradien di sekitarnya. Setiap posisi titik kontrol x dan y pada *template* tidak saling tergantung. Jadi *deformable template* diupdate terhadap masing-masing arah x dan y. Rumus yang digunakan untuk mengupdate *deformable template* ini adalah:

$$x_i(new) = x_i(old) + Sv_i \qquad (3\text{-}6)$$

dimana $S$ adalah parameter *steepest* (kecuraman langkah). Dan $v_i$ adalah:

[4] T. E. Shoup, *A Practical Guide to Computer Methods for Engineers*, Prentice-Hall International, Inc., 1979, 172-174.

ITS



$$v_i = \frac{\frac{\partial F}{\partial x_i}}{\sum_{j=1}^{N}\left[\left(\frac{\partial F}{\partial x_j}\right)^2\right]^{1/2}}, \qquad (3\text{-}7)$$

dimana $\frac{\partial F}{\partial x_i}$ adalah vektor gradien dan $N$ adalah jumlah *control point*.

## 3.4 Algoritma Multiresolusi Coarse-to-Fine

Algoritma *Multiresolution Coarse-to-Fine* membagi proses lokalisasi obyek ke dalam tiga tingkatan, yaitu: *coarsest*, *finer*, dan *finest*. Tujuan dari pembagian tingkatan ini adalah untuk mengurangi proses penghitungan yang dilakukan sehingga waktu komputasi yang dibutuhkan lebih efisien.

Pada tingkatan *coarsest*, medan potensial yang digunakan adalah yang paling halus dan ukuran citra tepi adalah seperempat (1/4) dari masing-masing dimensinya atau seperenambelas (1/16) besarnya dari citra asli. Sedangkan pada tingkatan *finer*, medan potensial yang digunakan lebih kasar apabila dibandingkan dengan tingkatan *coarsest* dan ukuran citra tepi yang digunakan adalah setengah (1/2) dari masing-masing dimensinya atau seperempat (1/4) besarnya dari citra asli. Untuk tingkatan *finest*, medan potensial yang digunakan paling kasar dan ukuran citra sesuai dengan ukuran citra aslinya.

Proses pada tiap-tiap tingkatan resolusi sangat berkaitan, dimana proses dimulai dari tingkatan *coarsest*. Setiap obyek yang dikenali oleh *template* pada tingkatan ini selanjutnya diproses pada tingkatan *finer*. Demikian juga pada tingkatan *finer*, obyek yang dikenali pada tingkatan *finer* selanjutnya diproses pada tingkatan *finest*. Kecocokan hanya diperoleh jika pada tingkatan *finest* terdapat

obyek yang nilai obyektifnya lebih kecil daripada batas nilai obyektif yang telah ditentukan.

Berikut ini adalah algoritma *Multiresolution Coarse-to-Fine* yang digunakan untuk proses pencocokan *deformable template* dengan obyek yang ada dalam citra masukan:

- Praproses citra:
  - Buat citra tepi dan arah gradien dari citra masukan dengan menggunakan *Canny Edge Detector*.
  - Hitung medan potensial dan arah tepi dari citra tepi pada tiga tingkatan resolusi: *coarsest*, *finer*, dan *finest*.
- Proses lokalisasi obyek:

1. Pada tingkatan *coarsest*, *deformable template* diinisialisasi pada semua posisi dan orientasi dengan menggunakan medan potensial dan arah tepi tingkatan *coarsest*. Penyesuaian posisi, orientasi, dan ukuran *deformable template* dilakukan dengan menggunakan algoritma *Generalized Hough Transform*. Untuk setiap posisi, proses pencocokan yang dilakukan adalah sebagai berikut:

loop: Hitung nilai obyektif.

- Jika nilai obyektif lebih kecil dari *threshold*, maka proses dilanjutkan ke tingkat berikutnya.
- Jika nilai obyektif lebih besar dari *threshold* dan jumlah iterasi melebihi batas iterasi, maka tidak ada obyek yang dapat kenali dan proses dihentikan.

- Jika nilai obyektif lebih besar *threshold* dan jumlah iterasi masih dibawah batas iterasi maka dilakukan *update* parameter *deformable template* dengan menggunakan metoda *Descent Gradient*. Proses diulang ke loop.

2. Pada tingkatan *finer*, *deformable template* yang digunakan adalah hasil proses pencocokan pada tingkatan *coarsest* dan medan potensial dan arah tepi yang digunakan adalah pada tingkatan *finer*. Proses yang dilakukan adalah sama dengan proses dalam loop pada tingkatan *coarsest*. Jika nilai obyektifnya lebih kecil dari *threshold*, maka proses dilanjutkan pada tingkatan *finest*.
3. Pada tingkatan *finest*, *template* yang digunakan adalah hasil proses pada tingkatan *finer* dan medan potensial dan arah tepi yang digunakan adalah pada tingkatan *finest*. Proses yang dilakukan adalah sama dengan proses dalam loop pada tingkatan *coarsest*. Jika nilai obyektifnya lebih kecil dari *threshold* maka terdapat obyek dalam citra masukan yang mirip dengan *template*. Proses selesai.

# BAB IV

# PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

# BAB IV

# PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Sebelum proses pembuatan perangkat lunak, maka proses perancangan perangkat lunak sangatlah dibutuhkan. Hal ini berguna untuk menunjang sistem yang akan dibuat sehingga kebutuhan akan sistem tersebut dapat diketahui sebelumnya. Dengan perancangan perangkat lunak juga akan mempermudah untuk mengadakan deteksi, koreksi, bahkan pengembangan lebih lanjut terhadap perangkat lunak yang dibuat.

## 4.1 Perancangan Perangkat Lunak

Perancangan perangkat lunak dalam tugas akhir ini dapat dikelompokkan dalam dua kelompok utama, yaitu: perancangan data dan perancangan proses. Perancangan data berguna untuk mengetahui data apa saja yang dibutuhkan dalam proses yang akan dikerjakan. Sedangkan perancangan proses berguna untuk mengintegrasikan semua proses yang terjadi dalam perangkat lunak yang akan dibuat.

### 4.1.1 Perancangan Data

Pemilihan struktur data yang tepat akan memberikan kemudahan dalam pembuatan perangkat lunak. Dengan telah mempertimbangkan beberapa aspek yang ada, maka struktur data yang digunakan adalah berupa *array*, *record*, dan

PERPUSTAKAAN ITS



*class*. Perancangan data yang akan digunakan untuk membuat perangkat lunak ini dapat dikelompokkan menjadi tiga kelompok utama, yaitu: data masukan, data internal, dan data keluaran.

#### 4.1.1.1 Data Masukan

Data masukan yang digunakan dalam perangkat lunak ini dapat dikelompokkan menjadi dua kelas utama, yaitu: data d*eformable template* dan data *image* (citra masukan).

1. *Data Deformable Template*

   Data *deformable template* yang digunakan adalah data *template* berupa gambar dengan format *bitmap* yang menyatakan kontur obyek dua dimensi dan dilengkapi dengan set *control point* dari properti kurva *B-Spline* sebagai parameter deformasi. Untuk membuat suatu *deformable template*, maka digunakan suatu *Deformable Template Editor*. Sedangkan untuk mempermudah pengambilan data *deformable template* tersebut, maka data *deformable template* disatukan dalam suatu file yang berisi daftar nama file *deformable template* yang berfungsi sebagai *library*. Untuk membuat suatu daftar nama file *deformable template* digunakan suatu *File List Editor*.

2. *Data Image*

   Data *image* (citra masukan) adalah data yang akan dicocokkan dengan *deformable template*. Dimana citra yang digunakan juga dengan format *bitmap*. Data citra yang dianjurkan adalah citra dengan bentuk obyek yang jelas didalamnya, tidak banyak mengandung tekstur karena akan mempersulit proses

pencocokan. Sedangkan ukuran antara obyek di dalam citra tersebut harus tidak jauh berbeda (perbedaan skala antara 0,7 – 1,3) dengan *deformable template* yang digunakan. Demikian juga dimensi citra yang digunakan tidak boleh terlalu besar (maksimal 640x480 satuan piksel), karena akan membutuhkan waktu komputasi yang lama dan juga tidak terlalu kecil ukurannya, karena akan diproses dalam tiga tingkatan resolusi yaitu: *coarsest*, *finer*, dan *finest*. Semua data citra masukan juga disatukan dalam suatu file yang berisi daftar nama file citra yang dapat dibuat dengan menggunakan *File List Editor*.

#### 4.1.1.2 Data Internal

Untuk menyimpan data *internal* (antar proses) digunakan suatu *array* dan *record*. *Array* merupakan struktur data yang mampu menyimpan data bertipe seragam yang banyak dan setiap data di dalamnya ditandai dengan suatu indek tertentu. Dalam hal ini *array* yang digunakan untuk menyimpan data adalah sangat besar sehingga *array* biasa tidak akan mampu menampungnya. Untuk itu, Penulis membuat suatu *array* dalam bentuk *class* dimana dapat menampung data yang cukup besar jumlahnya. *Array* yang digunakan adalah *array* yang bersifat statis. Alasan digunakannya *array* yang bersifat statis adalah faktor kecepatannya, dimana *array* yang bersifat statis lebih cepat dibandingkan dengan *array* yang bersifat dinamis.

#### 4.1.1.3 Data Keluaran

Data keluaran dari perangkat lunak yang dibuat adalah berupa data gambar *bitmap* yang terlokalisasi oleh *deformable template*. Disamping itu, informasi yang disajikan adalah nilai kecocokan antara *deformable template* dengan obyek yang terlokalisasi tersebut. Gambar obyek yang terlokalisasi oleh *deformable template* dapat disimpan kembali ke dalam bentuk format *bitmap*.

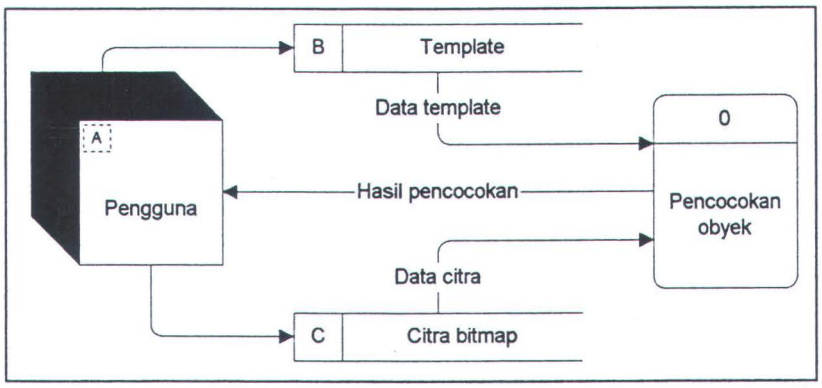
### 4.1.2 Perancangan Proses

Perancangan proses dari suatu sistem perangkat lunak adalah merupakan hal yang sangat penting sebelum perangkat lunak dibuat. Dengan membuat perancangan proses, maka kebutuhan penunjang dari masing-masing proses akan dapat diketahui dengan mudah sehingga segala kekurangannya dapat dideteksi secara dini dan kemudian dapat dilengkapinya.

Untuk perancangan proses perangkat lunak ini digunakan suatu *Data Flow Diagram* (DFD) / Diagram Aliran Data. Diagram yang digunakan adalah Diagram Aliran Data model Gane-Sarson. Diagram Aliran Data (DFD) yang digunakan dapat disusun berdasarkan levelnya masing-masing.

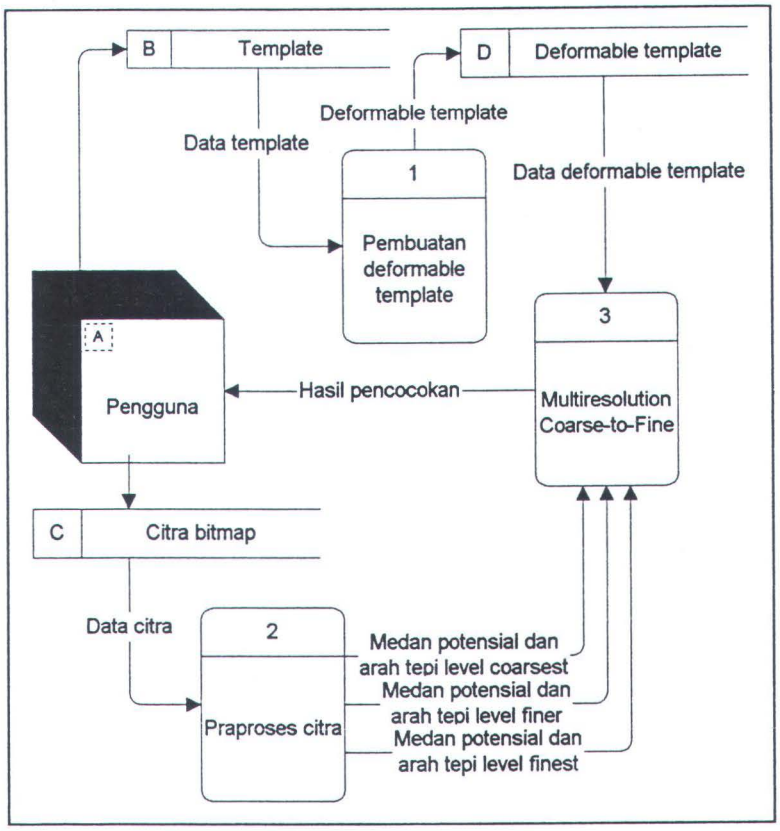
DFD untuk proses pencocokan obyek dengan *template* pada tingkat 0 seperti terlihat pada Gambar 4.1. Proses dimulai dari Pengguna (*external entity*) dengan memilih data *template* dari sekumpulan *template* yang berupa citra *bitmap* yang menggambarkan kontur dari suatu obyek dua dimensi. Kemudian memilih data citra dari sekumpulan citra *bitmap*. Setelah data *template* dan data citra

dipilih, maka proses pencocokan dapat dilakukan. Hasil proses pencocokan ditunjukkan kepada Pengguna sebagai hasil akhirnya.



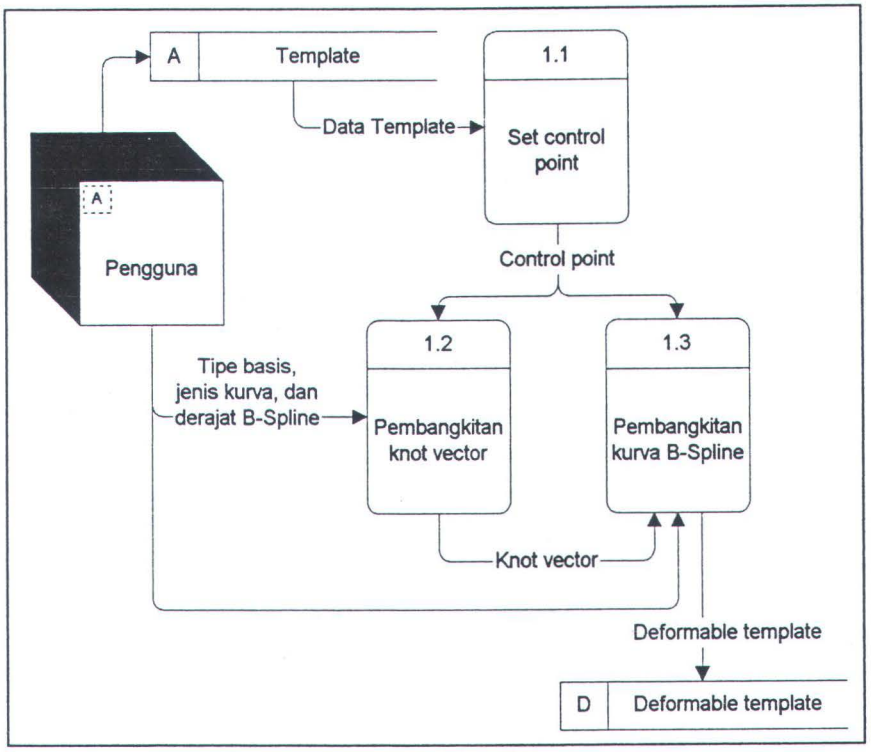

Gambar 4.1: DFD proses pencocokan obyek tingkat 0.

Dalam DFD proses pencocokan obyek pada tingkat 1 terdapat tiga proses utama (Gambar 4.2) yang merupakan penjabaran dari proses tingkat 0, yaitu: *Pembuatan deformable template*, *Praproses citra*, dan *Multiresolution Coarse-to-Fine*. Pembuatan *deformable template* dilakukan dengan menggunakan data *template*, dimana data *deformable template* ini adalah data yang akan dicocokkan dengan citra masukan. Untuk praproses citra dilakukan dengan menggunakan data citra masukan sebelum dicocokkan dengan *deformable template*. Praproses citra ini bertujuan mengambil informasi tepi yang terdapat didalamnya dan menghitung medan potensial dan arah tepi dalam medan potensial dari tepi yang dihasilkan tersebut. Sedangkan proses pencocokannya itu sendiri dilakukan dengan menggunakan algoritma *Multiresolution Coarse-to-Fine*. Detil dari ketiga proses tersebut dijelaskan dengan DFD pada tingkatan berikutnya.



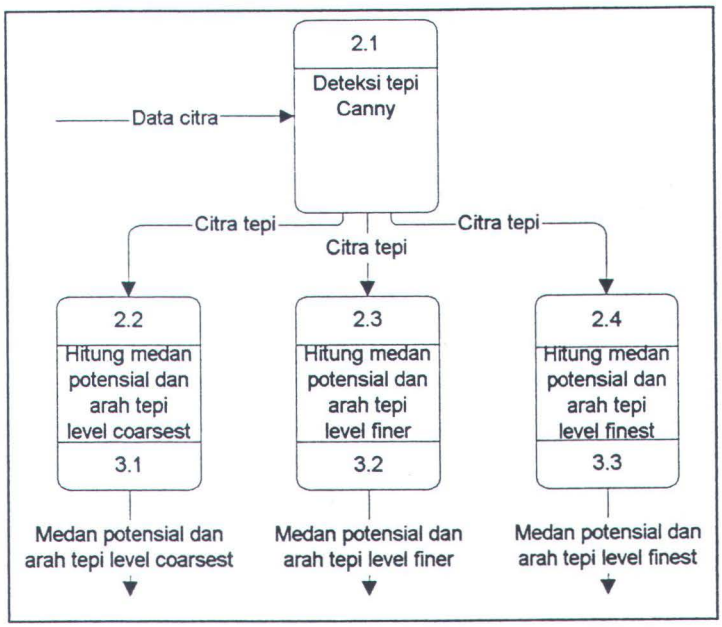

Gambar 4.2: DFD proses pencocokan obyek tingkat 1.

DFD Gambar 4.3 adalah proses pembuatan *deformable template* yang lebih detil. Data *deformable template* dibuat dengan cara memberikan set *control point* pada *template* yang menggambarkan kontur/tepi dua dimensi dari suatu obyek. Kemudian ditentukan tipe basis *B-Spline*, bentuk kurva, dan derajat yang digunakan. Pembangkitan kurva *B-Spline* dilakukan dengan menggunakan *control point*, tipe basis, bentuk kurva, dan derajat *B-Spline* serta *knot vector* yang dihasilkannya.

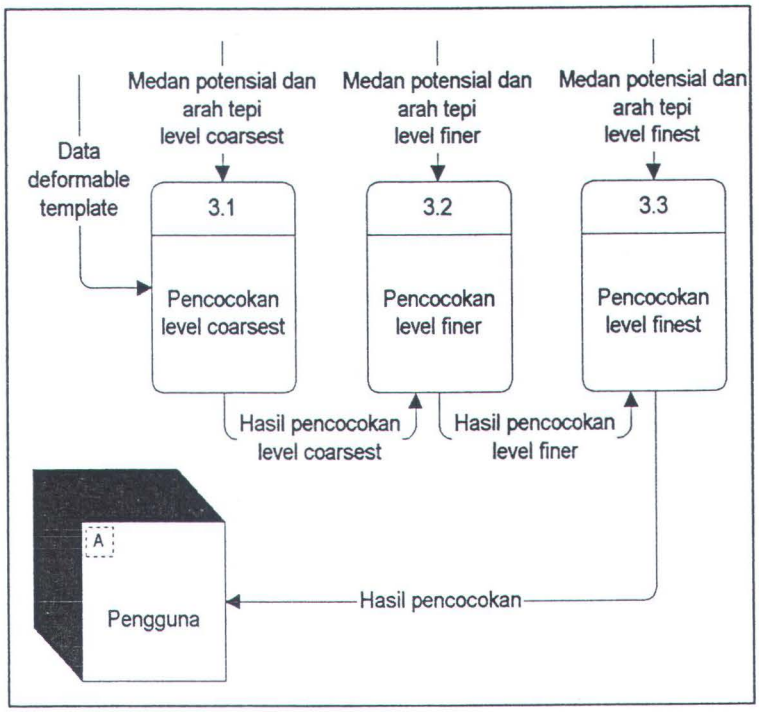



Gambar 4.3: DFD proses pembentukan *deformable template* tingkat 2.

Untuk praproses citra yang lebih detil seperti terlihat pada Gambar 4.4. Dalam praproses citra ini, data citra yang akan dicocokkan dengan *deformable template* terlebih dahulu harus diproses untuk mendapatkan informasi tepi, medan potensial dan arah tepi pada tiga tingkatan resolusi dari obyek yang ada didalamnya. Segmentasi untuk mendapatkan tepi citra digunakan detektor tepi *Canny* seperti yang dijelaskan sebelumnya. Selanjutnya dari citra tepi tersebut akan dihitung medan potensial dan arah tepinya dalam tiga tingkatan resolusi yaitu *coarsest*, *finer*, dan *finest*.




Gambar 4.4: DFD praproses citra tingkat 2.




Gambar 4.5: DFD algoritma *Multiresolution Coarse-to-Fine* tingkat 2.

Setelah praproses citra selesai maka proses selanjutnya adalah proses pencocokan dengan *deformable template*. Gambar 4.5 menunjukkan DFD Proses pencocokan ini dengan menggunakan algoritma *Multiresolution Coarse-to-Fine*, dimana proses pencocokan dilakukan pada setiap tingkatan resolusi. Hasil pencocokan setiap tingkat akan dilanjutkan pada tingkat berikutnya. Hasil pencocokan dari tingkatan *finest* merupakan hasil akhir pencocokan yang kemudian akan disajikan kepada Pengguna.

## 4.2 Pembuatan Perangkat Lunak

Pembuatan perangkat lunak dimulai dengan pembuatan struktur data dari perancangan data yang telah dibuat. Setelah pembuatan struktur data maka dilanjutkan dengan pembuatan beberapa prosedur dan fungsi yang merupakan implementasi dari perancangan proses yang telah dibuat.

Pembuatan struktur data berguna untuk menunjang kebutuhan perangkat lunak yang dibuat. Pembuatan struktur data ini dapat dikelompokkan sesuai fungsinya. Berikut ini adalah beberapa struktur data dalam sintak *Borland Delphi 3.0* yang digunakan, yaitu: *TCtrlPoint*, *TBSpline*, *TTemplate*, *TImageData*, *TZuList*, *TByteList*, *TIntegerList*, *TExtendedList*, *TPointList*, *TCurves*, *TImageInfo*, *TMatchInfo*, *TMatchRes*, dan *TMultiRes*.

Untuk membentuk suatu *deformable template* dibuat tiga kelas, yaitu: *TCtrlPoint*, *TBSpline*, dan *TTemplate*. Berikut ini adalah penjelasan dari masing-masing kelas tersebut:

- ***TCtrlPoint***

*TCtrlPoint* merupakan kelas turunan *TObject* (*TObject* adalah merupakan *parent* dari semua kelas yang dimiliki oleh *Borland Delphi 3.0*) yang digunakan untuk mengolah data titik kontrol. Beberapa fasilitas yang ada diantaranya adalah: menambah titik, menghapus satu maupun semua titik, menyisipkan suatu titik diantara dua titik yang berurutan, memindahkan satu maupun semua titik ke posisi yang baru, memutar orientasi semua titik, dan mengubah ukuran semua titik. Di samping beberapa fasilitas tersebut, kelas ini juga dilengkapi dengan fasilitas untuk mengcopy data titik dari kelas yang sama. Berikut ini adalah abstraksi dari kelas *TCtrlPoint*:

```
const
  MAX_CONTROL_POINT = 100;

type
  TCPIndex = 0..MAX_CONTROL_POINT - 1;
  TCPArray = array [TCPIndex] of TPoint;
  { Abstraksi Kelas TCtrlPoint }
  TCtrlPoint = class(TObject)
  protected
    FCP: TCPArray;
    FCPCount: Integer;
    function GetCP(Idx: Integer): TPoint;
    procedure SetCP(Idx: Integer; const CP: TPoint);
    procedure Error; virtual;
  public
    constructor Create;
    procedure Add(const CP: TPoint);
    procedure Delete(Idx: Integer);
    procedure Insert(Idx: Integer; const CP: TPoint);
    procedure Move(Idx: Integer; const CP: TPoint);
    procedure Clear; virtual;
    procedure Translate(DX, DY: Integer);
    procedure Rotate(CX, CY: Integer; Angle: Extended);
    procedure Scale(CX, CY: Integer; Scale: Extended);
    function GetNearestCPIndex(var Idx: Integer;
      const CP: TPoint): Boolean;
    procedure GetCPBounds(var Left, Right, Top, Bottom,
      CX, CY: Integer);
    procedure CopyFrom(Source: TCtrlPoint);
    property CPCount: Integer read FCPCount;
    property CP[Idx: Integer]: TPoint read GetCP write SetCP;
  end;
```

- ***TBSpline***

Kelas ini merupakan kelas turunan dari *TCtrlPoint*. Tujuan utama dari kelas ini adalah untuk membangkitkan kurva *B-Spline* dengan basis dan *knot vector* yang dihasilkan dari set *control point*, derajat, dan jenis kurva yang digunakan. Kelas ini juga memiliki fasilitas untuk mengcopy data dari kelas yang sama. Di samping itu juga memiliki fasilitas untuk baca/tulis file. Berikut ini adalah abstraksi dari kelas *TBSpline*:

```
const
  MAX_BSPLINE_DEGREE = 7;

type
  TCurveType = (Open, Close);
  TBasisType = (NonUniform, Uniform);
  TKnotIndex = 0..MAX_CONTROL_POINT + 2*MAX_BSPLINE_DEGREE-1;
  TSpaceIndex = 0..MAX_CONTROL_POINT + MAX_BSPLINE_DEGREE-1;
  TKnotVector = array [TKnotIndex] of Integer;
  TSpaceVector = array [TSpaceIndex] of Extended;
  { Abstraksi Kelas TBSpline }
  TBSpline = class(TCtrlPoint)
  private
    FCurveType: TCurveType;
    FBasisType: TBasisType;
    FDegree: Integer;
    function GenerateKnotVector: TKnotVector;
    function GenerateSpaceVector: TSpaceVector;
    function Basis(i, k: Integer; u: Extended;
      const KV: TKnotVector): Extended;
  public
    procedure Clear; override;
    function Spline: TCurves;
    procedure LoadFromFile(const FileName: string); virtual;
    procedure SaveToFile(const FileName: string); virtual;
    procedure CopyFrom(Source: TBSpline);
    property CurveType: TCurveType read FCurveType
      write FCurveType;
    property BasisType: TBasisType read FBasisType
      write FBasisType;
    property Degree: Integer read FDegree write FDegree;
  end;
```

- ***TTemplate***

*TTemplate* merupakan kelas turunan dari *TBSpline*. Kelas ini merupakan kelas utama untuk membentuk suatu *deformable template*. Tujuan dari kelas ini

adalah untuk mengintegrasikan informasi citra *template* yang berupa data *bitmap* dan nama filenya dengan parameter set *control point* dan kurva *B-Spline*. Di dalam kelas ini dilengkapi dengan fasilitas untuk mendeformasi *deformable template* dan menghitung nilai deformasinya. Juga disediakan fasilitas untuk baca/tulis file, menggambar kurva *B-Spline*, dan mengcopy data dari kelas yang sama. Berikut ini adalah abstraksi dari kelas *TTemplate*:

```
type
  { Abstraksi Kelas TTemplate }
  TTemplate = class(TBSpline)
  private
    FFileName: string;
    FBitmap: TBitmap;
    FDeformed: TCPArray;
  public
    constructor Create;
    destructor Destroy; override;
    function Valid: Boolean;
    procedure New(const FileName: string; Bitmap: TBitmap);
    procedure LoadFromFile(const FileName: string);
      override;
    procedure SaveToFile(const FileName: string); override;
    procedure CopyFrom(Source: TTemplate);
    procedure Draw(Dest: TCanvas;
      ShowCP, ShowPolygon, ShowBSpline: Boolean;
      clCP, clPolygon, clBSpline: TColor);
    procedure Deform(EdgePot: TByteList;
      DeltaX, DeltaY: TExtendedList; Rows, Cols: Integer;
      df: Integer);
    function Deformation(Variance: Extended): Extended;
    property FileName: string read FFileName;
    property Bitmap: TBitmap read FBitmap;
  end;
```

Sedangkan untuk menyimpan suatu data citra masukan dibuat suatu kelas *TImageData*.

- ***TImageData***

*TImageData* digunakan untuk menyatukan informasi citra masukan yaitu antara data citra dengan nama file citra tersebut. Dalam kelas ini disediakan

fasilitas untuk baca/tulis file dan mengcopy data dari kelas yang sama. Berikut ini adalah abstraksi dari kelas *TImageData*:

```
type
  { Abstraksi Kelas TImageData }
  TImageData = class(TObject)
  private
    FFileName: string;
    FBitmap: TBitmap;
  public
    constructor Create;
    destructor Destroy; override;
    procedure LoadFromFile(const FileName: string);
    procedure SaveToFile(const FileName: string);
    procedure CopyFrom(Source: TImageData);
    property FileName: string read FFileName
      write FFileName;
    property Bitmap: TBitmap read FBitmap write FBitmap;
  end;
```

Selanjutnya, untuk menyimpan data internal dibuat beberapa *array* dalam bentuk kelas seperti berikut ini:

- ***TZuList***

  *TZuList* merupakan suatu kelas yang dapat digunakan sebagai penyimpan data yang berupa *array pointer*. Dimana ukuran dari *item* tersebut dapat ditentukan sesuai kebutuhan. Dalam kelas ini juga disediakan fasilitas untuk baca/tulis file, dan mengcopy data dari kelas yang sama. Disamping itu juga disediakan fasilitas untuk menukar dua *item* dengan indek yang berbeda. Berikut ini adalah abstraksi dari kelas *TZuList*:

```
type
  { Abstraksi Kelas TZuList }
  TZuList = class(TObject)
  protected
    FMemory: Pointer;
    FCapacity: LongInt;
    FItemSize: Integer;
    FStart, FFinish: LongInt;
    PItem: Pointer;
    function GetPtr(Idx: LongInt): Pointer; virtual;
    procedure SetPtr(Idx: LongInt; Item: Pointer); virtual;
    procedure Error; virtual;
```

```
public
  constructor Create(Capacity: LongInt; ItemSize: Integer;
    Start: LongInt);
  destructor Destroy; override;
  procedure Exchange(Idx1, Idx2: LongInt);
  procedure LoadFromFile(const FileName: string);
  procedure SaveToFile(const FileName: string);
  procedure CopyFrom(Source: TZulist);
  property Memory: Pointer read FMemory;
  property Capacity: LongInt read FCapacity;
  property ItemSize: Integer read FItemSize;
  property Start: LongInt read FStart;
  property Finish: LongInt read FFinish;
  property Ptrs[Idx: LongInt]: Pointer read GetPtr
    write SetPtr; default;
end;
```

- ***TByteList***

*TByteList* merupakan kelas turunan *TZuList* dengan *item byte*. Berikut ini adalah abstraksi dari kelas *TByteList*:

```
type
  { Abstraksi Kelas TByteList }
  TByteList = class(TZuList)
  private
    FGetPByte, FSetPByte: ^Byte;
    function GetItem(Idx: LongInt): Byte;
    procedure SetItem(Idx: LongInt; Item: Byte);
  public
    constructor Create(Capacity, Start: LongInt);
    destructor Destroy; override;
    procedure MinMax(var Min, Max: Byte;
      var iMin, iMax: LongInt);
    property Items[Idx: LongInt]: Byte read GetItem
      write SetItem; default;
  end;
```

- ***TIntegerList***

*TIntegerList* merupakan kelas turunan *TZuList* dengan *item integer*. Berikut ini adalah abstraksi dari kelas *TIntegerList*:

```
type
  { Abstraksi Kelas TIntegerList }
  TIntegerList = class(TZuList)
  private
    FGetPInteger, FSetPInteger: ^Integer;
    function GetItem(Idx: LongInt): Integer;
    procedure SetItem(Idx: LongInt; Item: Integer);
```

```
public
  constructor Create(Capacity, Start: LongInt);
  destructor Destroy; override;
  procedure MinMax(var Min, Max: Integer;
    var iMin, iMax: LongInt);
  property Items[Idx: LongInt]: Integer read GetItem
    write SetItem; default;
end;
```

- ***TExtendedList***

*TExtendedList* merupakan kelas turunan *TZuList* dengan *item extended* (pecahan). Berikut ini adalah abstraksi dari kelas *TExtendedList*:

```
type
  { Abstraksi Kelas TExtendedList }
  TExtendedList = class(TZuList)
  private
    FGetPExtended, FSetPExtended: ^Extended;
    function GetItem(Idx: LongInt): Extended;
    procedure SetItem(Idx: LongInt; Item: Extended);
  public
    constructor Create(Capacity, Start: LongInt);
    destructor Destroy; override;
    procedure MinMax(var Min, Max: Extended;
      var iMin, iMax: LongInt);
    property Items[Idx: LongInt]: Extended read GetItem
      write SetItem; default;
  end;
```

Dalam kelas *TByteList*, *TIntegerList*, dan *TExtendedList* selain dapat menyimpan data dengan tipe masing-masing tersebut, juga disediakan fasilitas untuk mencari nilai minimum dan maksimum, dan indek dari nilai minimum dan maksimum dari data yang ada dalam *array*.

- ***TPointList***

*TPointList* merupakan kelas turunan *TZuList* dengan *item TPoint* (Titik (X, Y)). Berikut ini adalah abstraksi dari kelas *TPointList*:

```
type
  { Abstraksi Kelas TPointList }
  TPointList = class(TZuList)
```

```
private
  FGetPPoint, FSetPPoint: ^TPoint;
protected
  procedure SetItem(Idx: LongInt; Item: TPoint); virtual;
  function GetItem(Idx: LongInt): TPoint; virtual;
public
  constructor Create(Capacity: LongInt);
  destructor Destroy; override;
  property Items[Idx: LongInt]: TPoint read GetItem
    write SetItem; default;
end;
```

- ***TCurves***

*TCurves* merupakan kelas turunan *TPointList* dengan *item TPoint*. Dalam kelas ini *item* hanya dapat dimasukkan ke dalam *array* pada posisi setelah data terakhir dimasukkan dan jumlah *item* yang ada dalam *array* dicatat. Kelas ini berguna untuk menyimpan titik-titik kurva yang dihasilkan oleh kelas *TBSpline*. Berikut ini adalah abstraksi dari kelas *TCurves*:

```
type
  { Abstraksi Kelas TCurves }
  TCurves = class(TPointList)
  private
    FCount: LongInt;
    function GetItem(Idx: LongInt): TPoint; override;
  public
    procedure Add(const Item: TPoint);
    procedure CopyFrom(Source: TCurves);
    property Items[Idx: LongInt]: TPoint read GetItem;
      default;
    property Count: LongInt read FCount;
  end;
```

Selain tipe data *array* juga digunakan suatu *record*. *Record* yang dibuat adalah *TImageInfo* yang digunakan untuk menyimpan informasi yang dihasilkan oleh praproses citra.

- ***TImageInfo***

*TImageInfo* digunakan untuk menyimpan data internal yaitu data pada saat praproses citra. Informasi yang disimpan meliputi data: *tepi*, *gambar tepi*,

*delta-x*, *delta-y*, *magnitude*, *energi potensial*, *arah tepi*, dan *ukuran citra*.

Berikut ini adalah tipe *record TImageInfo* yang digunakan:

```
type
  TImageInfo = record
    EdgeImg: TByteList;
    EdgeImgBmp: TBitmap;
    DeltaX, DeltaY: TExtendedList;
    Mags: TExtendedList;
    EdgePot: TIntegerList;
    DirEdgePot: TExtendedList;
    Rows, Cols: Integer
  end;
```

Selain struktur data untuk data masukan dan data selama proses juga terdapat struktur data untuk data keluaran. Berikut ini beberapa kelas yang digunakan untuk menyimpan data keluaran tersebut, yaitu: *TMatchInfo*, *TMatchRes*, dan *TMultiRes*.

- ***TMatchInfo***

Untuk menyimpan informasi kecocokan antara *deformable template* dengan obyek dalam citra masukan digunakan suatu kelas *TMatchInfo*. *TMatchInfo* merupakan kelas turunan *TObject*. Dimana dalam kelas *TMatchInfo*, disamping dapat menyimpan informasi kecocokan juga menyediakan fasilitas untuk mengcopy data dari kelas yang sama. Berikut ini adalah abstraksi dari kelas *TMatchInfo*:

```
type
  { Abstraksi Kelas TMatchInfo }
  TMatchInfo = class(TObject)
  private
    FResTemplate: TTemplate;
    FObjective, FEnergy, FDeviation: Extended;
    FRotated, FScaled: Extended;
    FIteration: Integer;
  public
    constructor Create;
    destructor Destroy; override;
    procedure CopyFrom(Source: TMatchInfo);
```

```
    property ResTemplate: TTemplate read FResTemplate
      write FResTemplate;
    property Objective: Extended read FObjective
      write FObjective;
    property Energy: Extended read FEnergy write FEnergy;
    property Deviation: Extended read FDeviation
      write FDeviation;
    property Rotated: Extended read FRotated write FRotated;
    property Scaled: Extended read FScaled write FScaled;
    property Iteration: Integer read FIteration write
      FIteration;
  end;
```

- ***TMatchRes***

Kelas *TMatchRes* merupakan kelas turunan *TObject* yang digunakan untuk menyatukan informasi antara data citra masukan, data *deformable template*, dan informasi hasil pencocokan antara citra masukan dengan *deformable template*. Dalam kelas ini juga disediakan fasilitas untuk mengcopy data dari kelas yang sama. Berikut ini adalah abstraksi dari kelas *TMatchRes*:

```
type
  { Abstraksi Kelas TMatchRes }
  TMatchRes = class(TObject)
  private
    FImageData: TImageData;
    FOrgTemplate: TTemplate;
    FMatchingInfo: TMatchInfo;
  public
    constructor Create;
    destructor Destroy; override;
    procedure CopyFrom(Source: TMatchRes);
    property ImageData: TImageData read FImageData
      write FImageData;
    property OrgTemplate: TTemplate read FOrgTemplate
      Write FOrgTemplate;
    property MatchInfo: TMatchInfo read FMatchingInfo
      write FMatchingInfo;
  end;
```

- ***TMultiRes***

*TMultiRes* merupakan kelas turunan dari *TZuList*. Kelas ini digunakan untuk menyimpan hasil akhir semua *pointer TMatchRes* yang dihasilkan oleh proses

*Multiresolution Coarse-to-Fine*. Dalam kelas *TMultiRes*, data hanya dapat dimasukkan pada posisi terakhir saja dan data yang masuk tersebut dicatat jumlahnya. Berikut ini adalah abstraksi dari kelas *TMultiRes*:

```
type
  { Abstraksi Kelas TMultiRes }
  TMultiRes = class(TZuList)
  private
    FCount: Integer;
    function GetPtr(Idx: LongInt): Pointer; override;
  public
    constructor Create(Capacity: LongInt);
    procedure Add(Item: Pointer);
    procedure ResetCount;
    property Count: Integer read FCount;
  end;
```

Setelah pembuatan struktur data, maka pembuatan perangkat lunak dilanjutkan dengan pembuatan beberapa prosedur dan fungsi yang digunakan untuk proses pencocokan obyek dalam citra masukan dengan *deformable template*. Suatu prosedur atau fungsi dibuat agar proses yang dilakukan akan lebih terstruktur. Dalam satu prosedur atau fungsi yang baik biasanya hanya terdiri dari sebuah proses yang unik dimana proses ini berbeda dengan prosedur atau fungsi yang lain. Berikut ini adalah beberapa prosedur dan fungsi utama yang digunakan untuk melakukan proses pencocokan *deformable template* dengan obyek dalam citra masukan:

- ***Procedure ImagePreprocessing***

  Prosedur ini digunakan untuk menghitung *tepi*, *delta-x*, *delta-y*, *magnitude*, *energi potensial*, dan *arah tepi* dalam tiga tingkatan resolusi *Coarse-to-Fine*. Dalam prosedur ini terdapat prosedur *CannyEdge* untuk proses deteksi tepi dan prosedur *DirectionalEdgePotential* untuk menghitung *delta-x*, *delta-y*,

*magnitude*, *energi potensial*, dan *arah tepi* dalam medan potensial. Urutan proses dalam prosedur *ImagePreprocessing* adalah sebagai berikut:

- Deteksi tepi dari citra masukan dengan prosedur *CannyEdge*.
- Hitung *delta-x*, *delta-y*, *magnitude*, *energi potensial*, dan *arah tepi* dalam medan potensial (menggunakan informasi tepi pada masing-masing resolusi) dengan prosedur *DirectionalEdgePotential*.

Berikut ini adalah prototipe dari prosedur *CannyEdge*, prosedur *DirectionalEdgePotential*, dan prosedur *ImagePreprocessing*:

```
//-------------------------------------------------------------//
procedure CannyEdge(ImageData: TByteList; Rows, Cols: Integer;
  Sigma: Extended; WindowSize: Integer; LowTh, HighTh: Extended;
  var Smoothed: TByteList; var DeltaX, DeltaY: TExtendedList;
  var Mags: TIntegerList; var DirDegs: TExtendedList;
  var NMS: TByteList; var EdgeImg: TByteList);
//-------------------------------------------------------------//
//-------------------------------------------------------------//
procedure DirectionalEdgePotential(EdgeImg: TByteList;
  DirEdge: TExtendedList; Rows, Cols: Integer;
  CrsRho, CrsDim: Integer;
  FnrRho, FnrDim: Integer;
  FnsRho, FnsDim: Integer;
  var CrsImgInfo, FnrImgInfo, FnsImgInfo: TImageInfo);
//-------------------------------------------------------------//
//-------------------------------------------------------------//
procedure ImagePreprocessing(imageBitmap: TBitmap;
  Sigma: Extended; WindowSize: Integer; LowTh, HighTh: Extended;
  SaveToFile: Boolean; const Dir: string;
  var CrsImgInfo, FnrImgInfo, FnsImgInfo: TImageInfo);
//-------------------------------------------------------------//
```

Data hasil proses dari prosedur *ImagePreprocessing* diatas disimpan dalam suatu *record TImageInfo*.

- ***Procedure RunGHT***

Prosedur ini digunakan untuk menghitung kemungkinan posisi, orientasi, dan ukuran dari obyek yang ditempati oleh *deformable template*. Dimana

informasi yang dihasilkan akan digunakan untuk menyesuaikan *deformable template* dengan obyek yang ditempati dengan proses translasi, rotasi dan perubahan skala. Dalam prosedur ini terdapat fungsi *BuildRTableTemplate* yang digunakan untuk membuat *R-table template*.

Urutan proses dalam prosedur *RunGHT* ini adalah sebagai berikut:

- Buat *R-table* yang berdasarkan pada *template* dengan menggunakan fungsi *BuildRTableTemplate*.
- Untuk setiap titik dalam *template*, dihitung kemungkinan posisi pusat, skala, dan perbedaan sudutnya dengan tepi yang diamati.
- Buat *histogram* dari masing-masing posisi pusat, skala, dan perbedaan skala yang berhasil dideteksi.
- Cari jumlah frekuensi terbanyak dari masing-masing *histogram* posisi pusat, skala, dan perbedaan sudutnya dengan tepi yang diamati.

Berikut ini adalah prototipe dari fungsi *BuildRTableTemplate* dan prosedur *RunGHT*:

```
//------------------------------------------------------------//
function BuildRTableTemplate(Template: TTemplate): TRTable;
//------------------------------------------------------------//

//------------------------------------------------------------//
procedure RunGHT(Template: TTemplate; EdgeImg: TByteList;
  DirEdgePot: TExtendedList; Rows, Cols: Integer;
  var PossCX, PossCY: Integer;
  var PossScale, PossTheta: Extended);
//------------------------------------------------------------//
```

- ***Function RunMatching***

Dalam fungsi ini proses yang dilakukan adalah pencocokan *deformable template* dengan tepi obyek dengan cara menghitung energi yang dihasilkan.

ITS



Energi yang dihasilkan dihitung dengan menggunakan fungsi *CalcEnergy*. Proses deformasi *deformable template* secara lokal dengan mengupdate titik-titik kontrol juga dikerjakan dalam fungsi ini. Untuk deformasi lokal tersebut digunakan prosedur *Deform* yang dimiliki oleh kelas *TTemplate*. Dalam fungsi *RunMatching* ini informasi yang dihasilkan adalah nilai kecocokan antara *deformable template* dengan obyek yang ditempatinya. Urutan proses dalam fungsi ini adalah sebagai berikut:

*Loop*:

- Hitung energi yang dihasilkan oleh *deformable template* dengan obyek yang ditempatinya.
- Hitung nilai deviasi yang terjadi pada *deformable template* dengan menggunakan fungsi *Deformation* pada kelas *TTemplate*.
- Nilai obyektif dihitung dengan cara menjumlahkan kedua nilai diatas.
- Jika nilai obyektifnya lebih kecil dari *threshold*, maka proses selesai.
- Jika nilai obyektifnya tidak lebih kecil dari *threshold* dan masih dalam batas iterasi yang diijinkan, maka *deformable template* dideformasi dan proses diulangi ke *loop*.

Berikut ini adalah prototipe fungsi *CalcEnergy* dan *RunMatching* yang digunakan:

```
//-------------------------------------------------------------//
function CalcEnergy(Template: TTemplate;
  EdgePot: TIntegerList; DirEdgePot: TExtendedList;
  Rows, Cols: Integer): Extended;
//-------------------------------------------------------------//
```

```
//-------------------------------------------------------------//
function RunMatching(Template: TTemplate; ImageInfo: TImageInfo;
  df: Integer; Variance, Objective: Extended; MaxLoop: Integer;
  ShowMatching: Boolean; ptbox: TPaintBox;
  var MatchInfo: TMatchInfo): Boolean;
//-------------------------------------------------------------//
```

- ***Procedure MultiresolutionCoarseToFine***

  Prosedur ini merupakan prosedur utama untuk proses pencocokan dengan tiga tingkatan resolusi, yaitu: *coarsest*, *finer*, dan *finest*. Hasil proses dari prosedur ini adalah berupa obyek yang terlokalisasi oleh *deformable template* (jika ada obyek yang dikenali), baik dalam tingkatan *coarsest*, *finer*, maupun *finest*. Adapun proses dalam prosedur *MultiresolutionCoarseToFine* adalah sebagai berikut:

  - Praproses citra dengan prosedur *ImagePreprocessing*.
  - Inisialisasi *default deformable template*. Inisialisasi dilakukan hanya pada tingkatan *coarsest* yaitu dengan cara membuat *deformable template* dalam ukuran tingkatan *coarsest* dan merotasinya dalam 12 orientasi yang mewakili setiap orientasi $30^0$ dari orientasi $[0\text{-}360]^0$. Kemudian semua *default deformable template* ini diletakkan pada posisi kiri-atas (x:0, y:0) dari citra masukan.
  - Proses pencocokan pada tingkatan *coarsest* dilakukan dengan proses translasi *default deformable template* pada semua daerah dalam citra masukan.
  - Jika *deformable template* berada dalam medan potensial tepi, maka dilakukan deformasi global dengan menggunakan prosedur *RunGHT*.

Langkah selanjutnya adalah proses pencocokan dengan menggunakan fungsi *RunMatching*. Jika ada obyek yang dikenali oleh *deformable template* pada tingkatan ini maka dicatat informasinya untuk diproses dalam tingkatan *finer*.

- Proses pencocokan pada tingkatan *finer* menggunakan fungsi *RunMatching* dan *deformable template* yang digunakan adalah hasil pencocokan pada tingkatan *coarsest*. Dan *deformable template* disini diperbesar sesuai ukuran tingkatan *finer*.
- Sedangkan proses pencocokan pada tingkatan *finest* menggunakan fungsi *RunMatching* dan *deformable template* yang digunakan adalah hasil pencocokan pada tingkatan *finer*. Dan *deformable template* disini juga diperbesar sesuai ukuran tingkatan *finest*.

Berikut ini adalah prototipe dari prosedur *MultiresolutionCoarseToFine* yang digunakan:

```
//-------------------------------------------------------------//
procedure MultiresolutionCoarseToFine(Template: TTemplate;
  ImageData: TImageData; Sigma: Extended; WindowSize: Integer;
  LowTh, HighTh: Extended; Objective: Extended;
  Variance, MaxLoop: Integer; SaveIntOutput: Boolean;
  const OutputDir: string; isRunGHT, isShowMatching: Boolean;
  var CrsMultiRes, FnrMultiRes, FnsMultiRes: TMultiRes);
//-------------------------------------------------------------//
```

Dalam prosedur *MultiresolutionCoarseToFine* ini, hasil proses yang dikerjakan disimpan dalam *TMultiRes*.

# BAB V

# UJI COBA DAN ANALISA HASIL PERANGKAT LUNAK

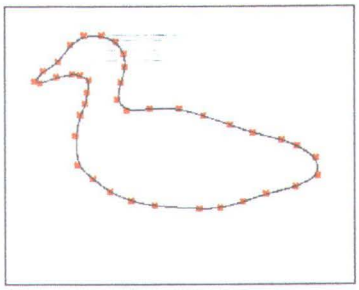
# BAB V

# UJI COBA DAN ANALISA HASIL PERANGKAT LUNAK

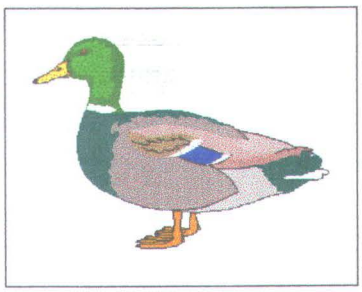
Perangkat yang dibutuhkan untuk menjalankan perangkat lunak yang dibuat dalam tugas akhir ini dapat dikelompokkan menjadi dua bagian, yaitu: perangkat lunak sistem operasi dan perangkat keras. Perangkat lunak sistem operasi yang digunakan adalah Microsoft Windows '95 / NT 4.0 atau versi sesudahnya. Sedangkan perangkat keras yang digunakan setidaknya harus mendukung kebutuhan minimal dari sistem operasi tersebut. Perangkat keras yang digunakan oleh penulis dalam membuat perangkat lunak ini adalah prosesor Pentium-166 dengan memory (RAM) sebesar 32 MB. Meskipun demikian, Penulis menganjurkan untuk menggunakan perangkat keras yang lebih baik daripada konfigurasi tersebut agar menghasilkan performa kecepatan yang lebih baik pula. Hal ini disebabkan karena perangkat lunak yang dibuat membutuhkan operasi perkalian yang cukup banyak sehingga waktu komputasinya tinggi. Selanjutnya, berikut ini adalah beberapa uji coba yang telah dilakukan dan analisanya dengan menggunakan beberapa tipe *deformable template* dan beberapa citra masukan.

## 5.1 Uji Coba dengan Obyek yang Orientasi dan Ukurannya Tidak Jauh Berbeda

Sebelum proses pencocokan antara *deformable template* (Gambar 5.1) dengan obyek yang ada di dalam citra masukan (Gambar 5.2), maka terlebih dahulu dilakukan praproses citra (seperti dijelaskan pada bab sebelumnya).

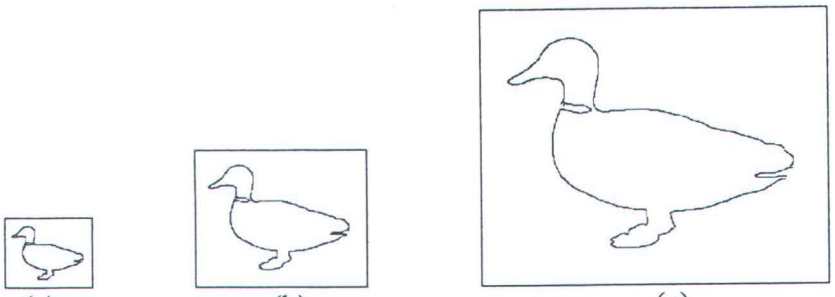


Gambar 5.1: Data *deformable template* dengan parameter-parameternya.



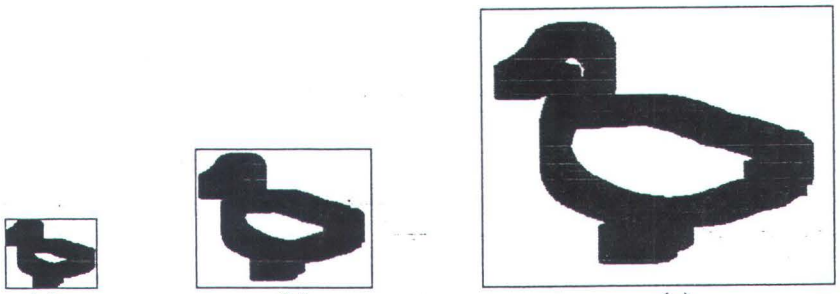
Gambar 5.2: Citra asal (Duck.bmp – 350x275).

Berikut ini adalah hasil dari praproses citra (*Coarse-to-Fine*) yang diperoleh dimana citra masukan yang digunakan sengaja citra dengan bentuk obyek yang sederhana dan jelas didalamnya:



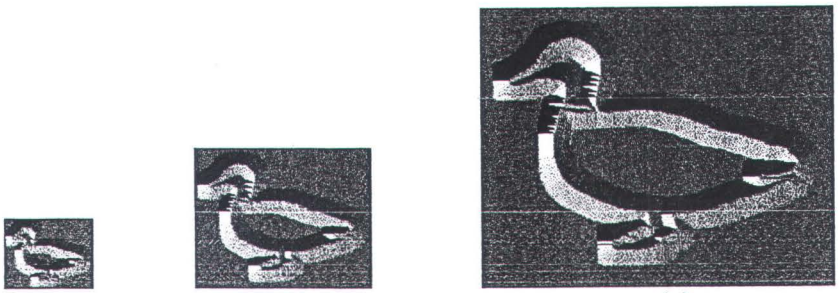
(a) (b) (c)

Gambar 5.3: Citra tepi dari Gambar 5.2 ($\sigma = 1$, *HighTh* = 0,9, dan *LowTh* = 0,5). (a) Tingkat *coarsest*. (b) Tingkat *finer*. (c) Tingkat *finest*.



(a) (b) (c)

Gambar 5.4: Medan potensial tepi dari Gambar 5.3: (a) Tingkat *coarsest* ($\rho = 12$). (b) Tingkat *finer* ($\rho = 8$). (c) Tingkat *finest* ($\rho = 3$).

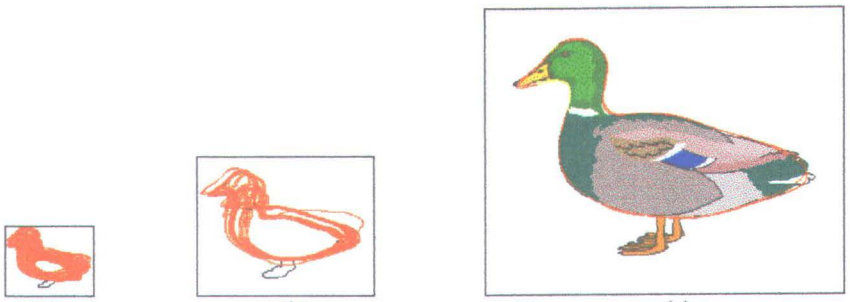


(a) (b) (c)

Gambar 5.5: Arah tepi dalam medan potensial tepi dari Gambar 5.4: (a) Tingkat *coarsest*. (b) Tingkat *finer*. (c) Tingkat *finest*.

Setelah praproses citra, maka proses selanjutnya adalah pencocokan obyek dengan menggunakan algoritma *Multiresolution Coarse-to-Fine*. Pada percobaan ini, orientasi dan ukuran *deformable template* sengaja dibuat tidak jauh berbeda dengan obyek yang ada di dalam citra masukan. Hal ini dimaksudkan untuk

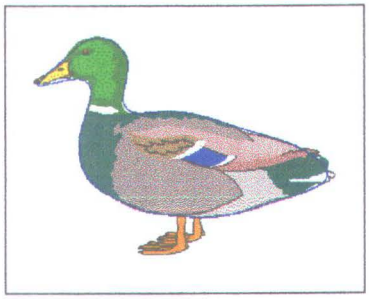
mengetahui kebenaran dari perangkat lunak yang telah dibuat apakah bisa mengenali obyek tersebut atau tidak. Dari uji coba yang telah dilakukan dengan menggunakan data tersebut diatas diperoleh hasil pencocokan pada tingkat *coarsest* sebanyak 80 obyek, pada tingkat *finer* sebanyak 28 obyek, dan pada tingkat *finest* sebanyak 3 obyek seperti Gambar 5.6 berikut ini:



(a) (b) (c)

Gambar 5.6: Hasil proses pencocokan dengan algoritma *Multiresolution Coarse-to-Fine*. (a) Tingkat *coarsest*. (b) Tingkat *finer*. (c) Tingkat *finest*.

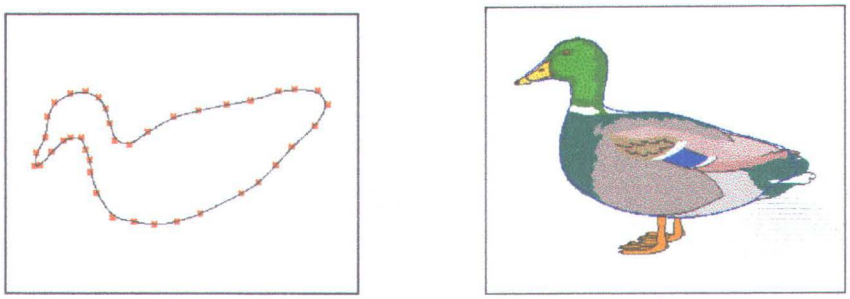
Hasil pencocokan pada tingkat *finest* merupakan hasil akhir proses pencocokan. Dalam tingkat ini obyek yang dapat dikeluarkan dapat dipilih jumlahnya sesuai keinginan pengguna. Berikut ini adalah hasil terbaik yang berhasil dikenali oleh *deformable template* (dari Gambar 5.1) terhadap citra masukan (dari Gambar 5.2):



Gambar 5.7: Hasil proses pencocokan dengan algoritma *Multiresolution Coarse-to-Fine* pada tingkat *finest* yang terbaik

## 5.2 Uji Coba dengan Obyek yang Berbeda Orientasinya

Tujuan dari uji coba ini adalah untuk mengetahui keberhasilan dari penerapan algoritma *Generalized Hough Transform* (GHT) dari segi perhitungan kemungkinan orientasinya. Berikut ini adalah *deformable template* yang digunakan dan hasil pencocokannya terhadap Gambar 5.2:



(a) (b)

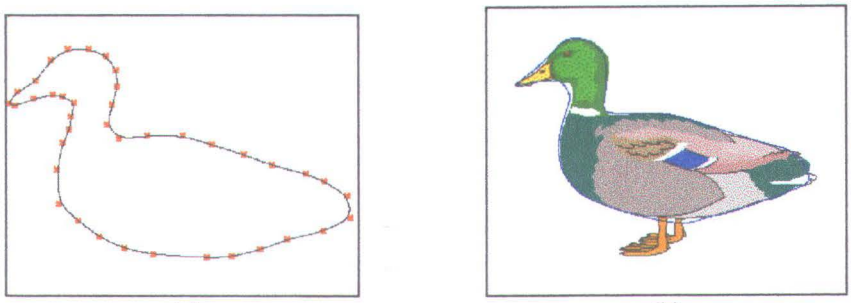
Gambar 5.8: Pencocokan obyek dengan *deformable template* yang berbeda orientasinya. (a) *Deformable template*. (b) Hasil pencocokan ($L$ = 0,232).

Dari uji coba diatas diperoleh informasi bahwa *deformable template* mengalami perubahan orientasi. Informasi yang diperoleh adalah *deformable template* harus menyesuaikan orientasinya dengan perotasian sebesar $30^0$ (searah jarum jam). Informasi ini menunjukkan bahwa proses perhitungan selisih sudut ($\theta$) dalam penerapan algoritma *Generalized Hough Transform* (GHT) mungkin kurang berhasil.

## 5.3 Uji Coba dengan Obyek yang Berbeda Ukurannya

Uji coba dengan menggunakan *deformable template* yang berbeda ukurannya dengan obyek yang dicocokkan bertujuan untuk mengetahui keberhasilan dari penerapan algoritma *Generalized Hough Transform* (GHT)

khususnya dari segi perhitungan kemungkinan skalanya. Berikut ini adalah *deformable template* yang digunakan dan hasil pencocokannya terhadap Gambar 5.2:



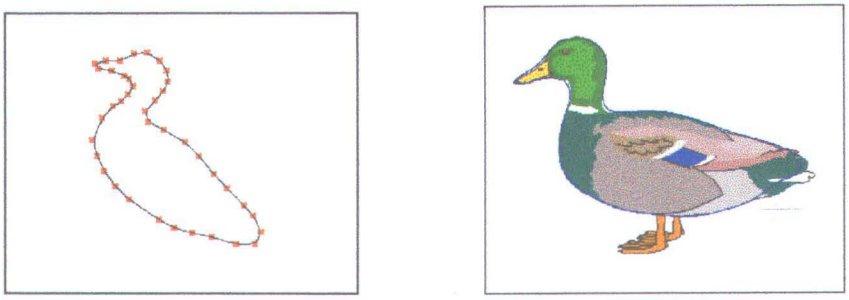
(a) (b)

Gambar 5.9: Pencocokan obyek dengan *deformable template* yang berbeda ukurannya. (a) *deformable template*. (b) Hasil pencocokan ($L = 0{,}229$).

Informasi yang diperoleh dari hasil pencocokan tersebut adalah perubahan skala yang terjadi pada *deformable template*. Untuk dapat mengenali obyek pada Gambar 5.2 maka *deformable template* harus mengalami perubahan skala sebesar 0,85.

## 5.4 Uji Coba dengan Obyek yang Berbeda Orientasi dan Ukurannya

Uji coba berikut ini adalah dengan menggunakan *deformable template* yang orientasi dan ukurannya berbeda sekaligus terhadap obyek Gambar 5.2. Berikut ini adalah hasil pencocokannya terhadap Gambar 5.2:



(a) (b)

Gambar 5.10: Pencocokan obyek dengan *deformable template* yang berbeda orientasi dan ukurannya. (a) *Deformable template*. (b) Hasil pencocokan ($L = 0{,}223$).

Informasi hasil pencocokan menunjukkan bahwa orientasi dan ukuran yang dihasilkan berbeda dengan aslinya. Berdasarkan informasi tersebut, untuk dapat mengenali obyek pada Gambar 5.2 maka *deformable template* harus diputar sebesar $330^0$ (searah jarum jam) dan kemudian diperbesar ukurannya dengan faktor skala 1,25.

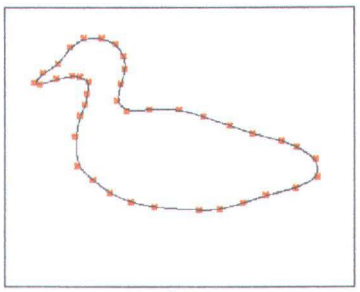
## 5.5 Uji Coba dengan Banyak Obyek

Dari uji coba **5.1-5.4** menunjukkan bahwa suatu obyek dapat dideteksi dengan menggunakan beberapa *deformable template* yang berbeda. Dengan menggunakan asumsi tersebut, maka dapat juga dilakukan pendeteksian obyek yang berbeda dengan sebuah *deformable template.* Bahkan dapat dilakukan pendeteksian semua obyek yang mirip dengan semua *deformable template* yang digunakan tersebut dimana bentuk, orientasi, dan ukurannya masih dalam batas yang diijinkan.

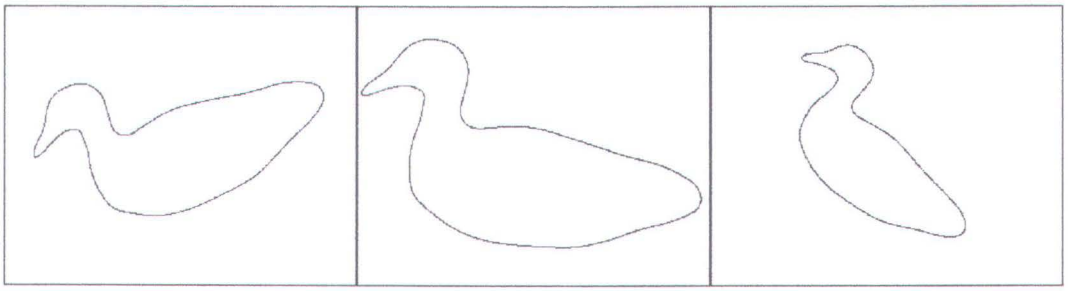
Jika pencocokan dengan menggunakan *deformable template* pada Gambar 5.1, maka dapat juga dideteksi beberapa obyek seperti Gambar 5.11 berikut ini:

ITS





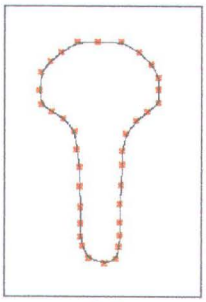
(a)



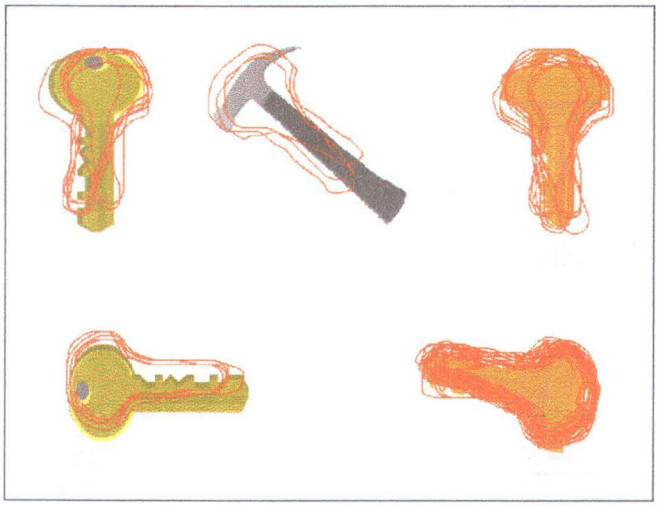
(b) (c) (d)

Gambar 5.11: *Deformable template* dengan obyek variasinya. (a) *Deformable template*. (b) Obyek dengan orientasi yang berbeda. (c) Obyek dengan ukuran yang berbeda. (d) Obyek dengan orientasi dan ukuran yang berbeda.

Berikut ini adalah contoh uji coba pencocokan *deformable template* dengan menggunakan beberapa obyek yang ada di dalam suatu citra masukan:



(a)



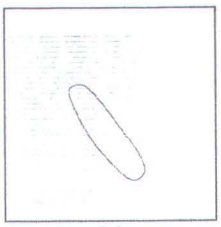
(b)

Gambar 5.12: Pencocokan *deformable template* dengan banyak obyek dalam sebuah citra masukan. (a) *Deformable template*. (b) Hasil pencocokan.

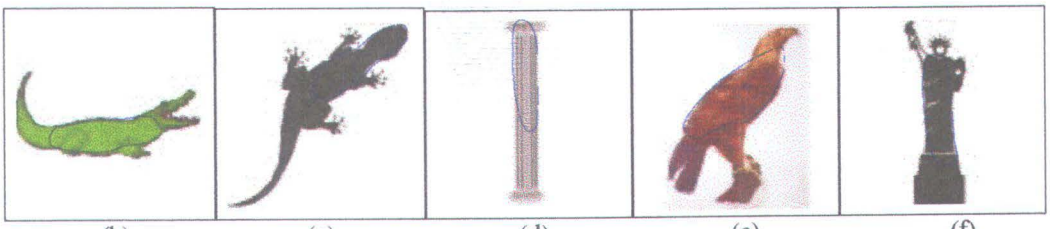
Dalam uji coba tersebut obyek yang dapat dideteksi tidak terlalu baik bentuknya. Hal ini dikarenakan setiap obyek memiliki bentuk yang berbeda dan nilai batas obyektif yang digunakan cukup besar yaitu 0,5. Tujuan digunakannya nilai batas obyektif yang cukup besar ini adalah agar dapat mengenali bentuk obyek yang lebih banyak.

## 5.6 Uji Coba dengan Banyak Citra

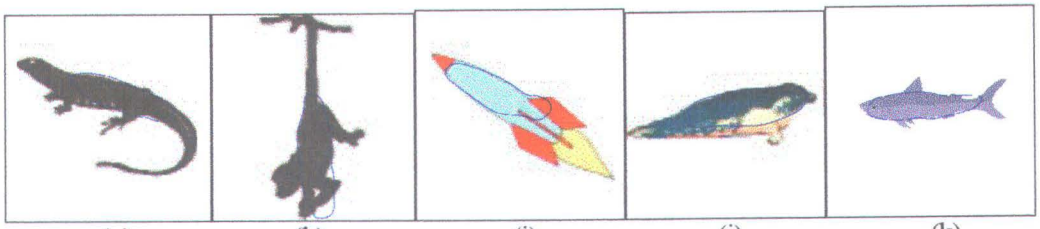
Dalam uji coba dengan menggunakan banyak citra, batas nilai obyektif dari hasil pencocokan sengaja diset lebih besar dibandingkan dengan menggunakan satu obyek saja. Hal ini bertujuan agar dapat mengidentifikasi semua obyek yang tidak terlalu mirip bentuknnya. Uji coba dilakukan dengan menggunakan banyak data citra masukan dan sebuah *deformable template*. Berikut ini adalah beberapa hasil pencocokan obyek (Gambar 5.13(b)-(k)) dari banyak data citra masukan dengan menggunakan sebuah *deformable template* Gambar 5.13(a).



(a)



(b) (c) (d) (e) (f)



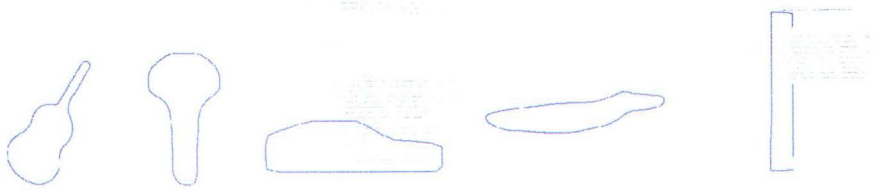
(g) (h) (i) (j) (k)

Gambar 5.13: Pencocokan sebuah *deformable template* dengan beberapa citra masukan. (a) *Template*. (b) Alligator.bmp ($L = 0{,}223$). (c) Chameleon.bmp ($L = 0{,}239$). (d) Column.bmp ($L = 0{,}228$). (e) Eagle.bmp ($L = 0{,}242$). (f) Liberty.bmp ($L = 0{,}238$). (g) Lizard.bmp ($L = 0{,}229$). (h) Monkey.bmp ($L = 0{,}249$). (i) Rocket.bmp ($L = 0{,}231$). (j) Seal.bmp ($L = 0{,}248$). (k) Shark.bmp ($L = 0{,}235$).

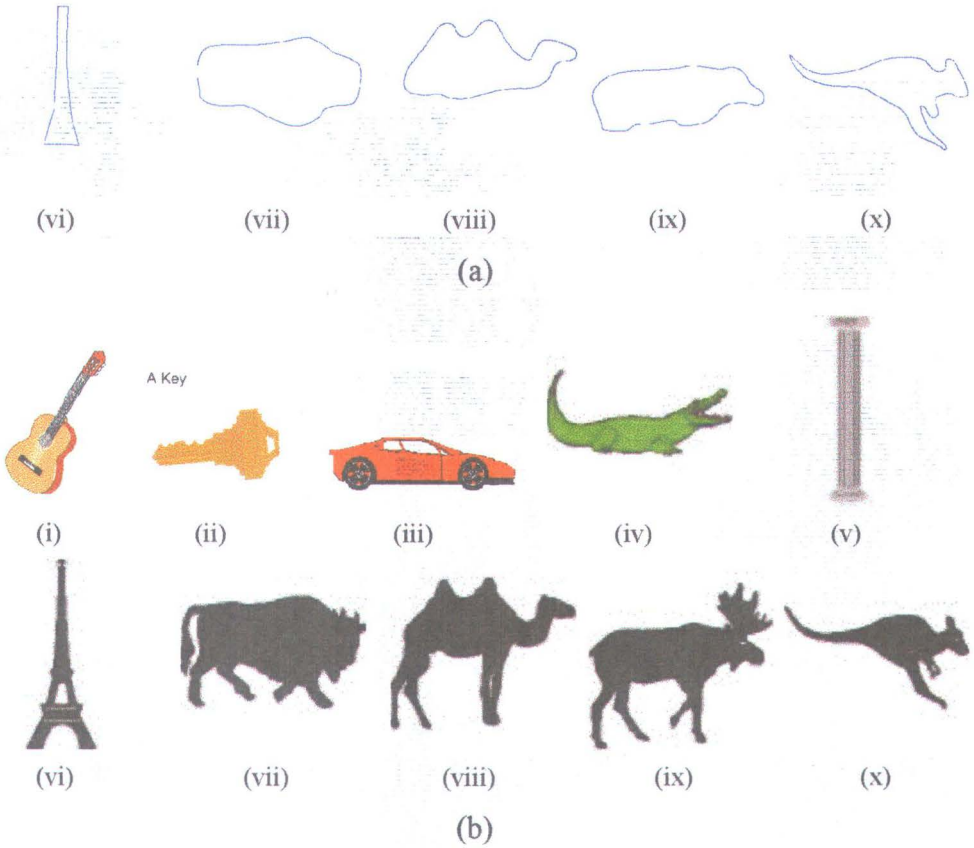
Dari hasil pencocokan diatas terlihat bahwa semakin besar batas nilai obyektif yang digunakan, maka kemungkinan untuk mendapatkan obyek yang tidak diharapkan dapat terjadi (misalnya: Monkey.bmp). Oleh karena itu, pengguna harus dapat menentukan batas nilai obyektif sesuai dengan keperluannya.

## 5.7 Uji Coba Perbandingan

Pada uji coba kali ini, pencocokan *deformable template* tidak saja dilakukan terhadap obyek yang sama melainkan juga dilakukan terhadap obyek yang berbeda. Berikut ini adalah beberapa *deformable template* dan citra masukan yang digunakan untuk proses perbandingan:



(i) (ii) (iii) (iv) (v)




Gambar 5.14: Pencocokan perbandingan beberapa *deformable template* dengan beberapa obyek. (a) *Deformable template*. (b) Citra masukan.

Dari hasil pencocokan perbandingan dengan data diatas diperoleh hasil seperti ditunjukkan dalam tabel 5.1:

| | Citra masukan | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *Template* | I | II | III | IV | V | VI | VII | VIII | IX | X |
| Guitar(I) | 0,231 | 0,246 | X | X | X | 0,513 | X | X | X | 0,415 |
| Key(II) | 0,313 | 0,227 | X | X | X | 0,375 | X | X | X | 0,465 |
| Car(III) | X | X | 0,213 | X | X | X | X | X | 0,437 | X |
| Alligator(IV) | X | X | X | 0,235 | X | X | X | X | 0,382 | X |
| Column(V) | X | X | X | X | 0,225 | 0,247 | X | X | X | X |
| Eiffel(VI) | X | X | X | X | 0,236 | 0,218 | X | X | X | X |
| Bison(VII) | X | X | X | X | X | X | 0,229 | X | 0,316 | X |
| Camel(VIII) | X | X | X | X | X | X | X | 0,228 | X | X |
| Moose(IX) | X | X | X | X | X | X | 0,33 | X | 0,234 | X |
| Kangaroo(X) | X | X | X | X | X | X | X | X | X | 0,232 |

Tabel 5.1: Hasil pencocokan perbandingan.

Dari tabel di atas terlihat bahwa pencocokan *deformable template* sesuai dengan obyek yang diharapkan dalam citra masukan. Hal ini ditunjukkan oleh nilai obyektif yang paling kecil (garis bawah) yang dihasilkannya. Untuk tanda (X) menyatakan bahwa obyek tidak cocok sama sekali. Hal ini disebabkan oleh perbedaan bentuk dan ukuran yang tidak diijinkan atau *deformable template* tidak berada di dalam medan potensial tepi obyek tersebut.

# BAB VI

# PENUTUP

# BAB VI

# PENUTUP

Sebagai penutup, diberikan kesimpulan dari hasil perangkat lunak yang telah dibuat berdasarkan uji coba yang telah dilakukan oleh Penulis. Di samping itu juga diberikan catatan untuk pengembangan perangkat lunak selanjutnya.

## 6.1 Kesimpulan

Dari beberapa hasil uji coba perangkat lunak dan analisa yang telah dilakukan, maka dapat diambil kesimpulan bahwa:

a. Dengan *deformable template* obyek yang dapat dikenali lebih banyak variasinya. Meskipun demikian hasil obyek yang dapat dikenali sangat dipengaruhi oleh dua faktor, yaitu:

- *Informasi Prior*

  Pemilihan nilai *variance* yang tepat pada distribusi *prior* sebagai faktor penalti terhadap deformasi *deformable template* yang tidak diharapkan adalah sangat sulit. Meskipun pada dasarnya semakin besar nilai yang digunakan, maka *deformable template* akan semakin elastis dan sebaliknya.

- *Informasi Likelihood*

  *Likelihood* merupakan informasi yang menunjukkan ukuran kecocokan antara *deformable template* dengan obyek yang ditempatinya. Nilai

kecocokan ini dapat dipengaruhi oleh beberapa faktor dalam praproses citra, yaitu:

- *Segmentasi citra*

  Deteksi tepi dengan menggunakan *Canny Edge Detector* dapat menghasilkan obyek yang berbeda karena faktor *Sigma*, *LowTh*, *HighTh*, dan *ukuran jendela* yang berbeda akan menghasilkan informasi tepi yang berbeda.

- *Pemilihan nilai Rho* ($\rho$)

  Penggunaan medan potensial tepi dengan faktor kehalusan ($\rho$) yang berbeda akan menghasilkan nilai energi potensial yang berbeda pula.

b. Penggunaan algoritma Hough yang dibakukan (GHT) pada dasarnya dapat digunakan untuk mendeteksi obyek dengan perbedaan skala yang cukup besar dan sembarang orientasi obyek, akan tetapi akan memerlukan waktu komputasi yang cukup tinggi. Dalam percobaan yang telah dilakukan, penerapan algoritma GHT tersebut kurang memenuhi harapan khususnya dalam perhitungan kemungkinan orientasi obyek yang ditempatinya. Hal ini disebabkan oleh faktor arah tepi dalam medan potensial yang digunakan kurang tepat.

## 6.2 Kemungkinan Pengembangan

Beberapa pengembangan yang memungkinkan dari perangkat lunak yang telah dibuat ini diantaranya adalah:

a. Karena dalam pembentukan *deformable template* masih bersifat manual, yaitu dengan menempatkan *control point* pada sekitar kontur dalam *bitmap template* maka akan memerlukan kepandaian pengguna untuk menempatkannya sehingga faktor ketelitian masih dibutuhkan. Oleh karena itu, pengembangan yang memungkinkan adalah dengan pembuatan *deformable template* yang menggunakan proses pembelajaran dari beberapa obyek secara otomatis sehingga diperoleh bentuk umum suatu *deformable template* yang lebih baik.

b. Hal yang juga memungkinkan untuk pengembangan perangkat lunak selanjutnya adalah dengan menggunakan data citra masukan yang bersifat *multi frame*, dimana perubahan bentuk antar *frame* sangat kecil dan *deformable template* diinisialisasi pada *frame* pertama saja sedangkan untuk *frame* selanjutnya digunakan *deformable template* yang dihasilkan pada *frame* sebelumnya sehingga dapat digunakan untuk mengikuti jejak obyek. Aplikasi ini dapat diterapkan untuk format gambar bergerak seperti *MPEG*. Dengan demikian dapat diterapkan untuk otomasi gerak kamera, misalnya dalam bidang kecerdasan buatan untuk sistem keamanan.

# DAFTAR PUSTAKA

# DAFTAR PUSTAKA


1. Y. Zhong, *Object Matching Using Deformable Templates,* PhD. Thesis, Department of Computer Science, Michigan State University, May 1997.

2. A. K. Jain, Y. Zhong, and S. Lakshmanan, "Object Matching Using Deformable Templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 3, March 1996.

3. J. F. Canny, *Finding Edges and Lines in Images*, MSc. Thesis, Massachusetts Institute of Technology, June 1983.

4. J. F. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, November 1986.

5. M. E. Mortenson, *Geometric Modelling*, Second Edition, Wiley Computer Publishing, 1997, 113-137.

6. D. Hearn and M. P. Baker, *Computer Graphics*, Second Edition, Prentice-Hall International, Inc., 1994, 334-344.

7. T. E. Shoup, *A Practical Guide to Computer Methods for Engineers*, Prentice-Hall International, Inc., 1979, 172-174.

8. W. K. Pratt, *Digital Image Processing*, Second Edition, John Wiley & Sons, Inc., 1991.

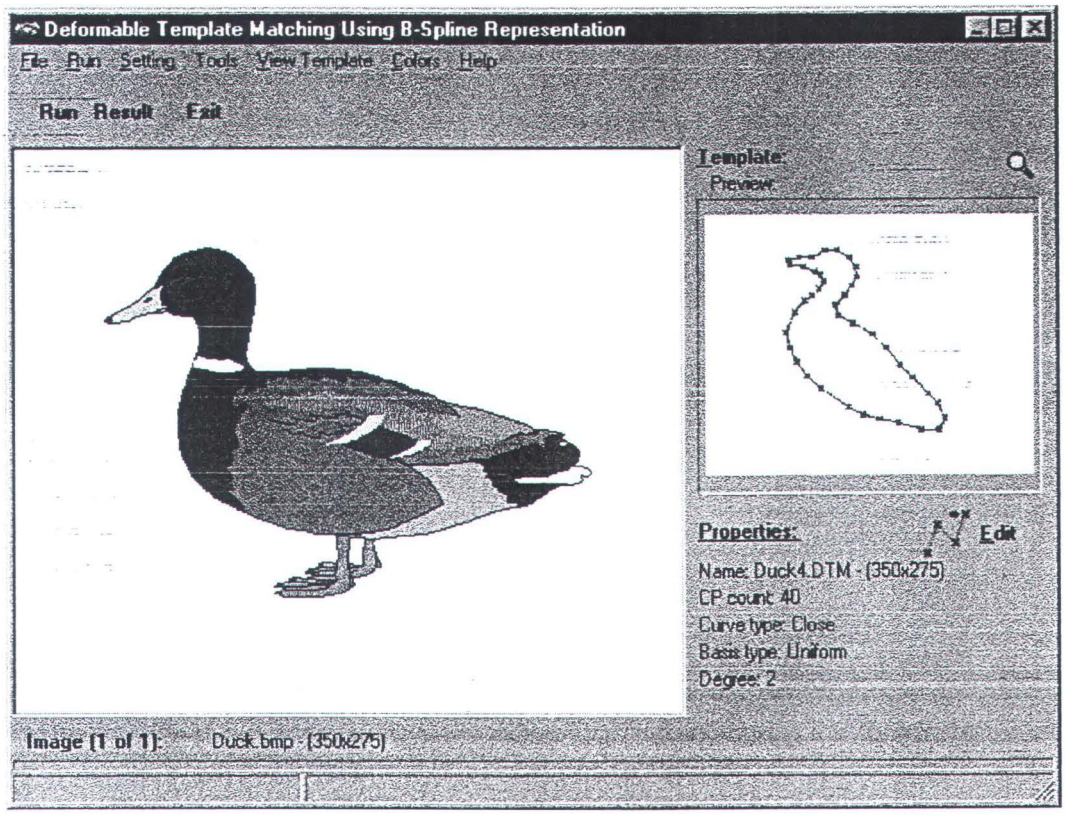
# LAMPIRAN

# LAMPIRAN

# PANDUAN PENGGUNAAN PERANGKAT LUNAK

Panduan penggunaan perangkat lunak yang dibuat bertujuan untuk memberikan informasi kepada pengguna tentang cara-cara menggunakan perangkat lunak yang terdiri dari beberapa bagian seperti: *input-output*, *proses*, dan *tools*. Berikut ini akan dijelaskan tentang beberapa bagian yang berkaitan dengan bagian tersebut:

## L.1 Tampilan Utama




Gambar L.1: Tampilan antarmuka utama perangkat lunak.

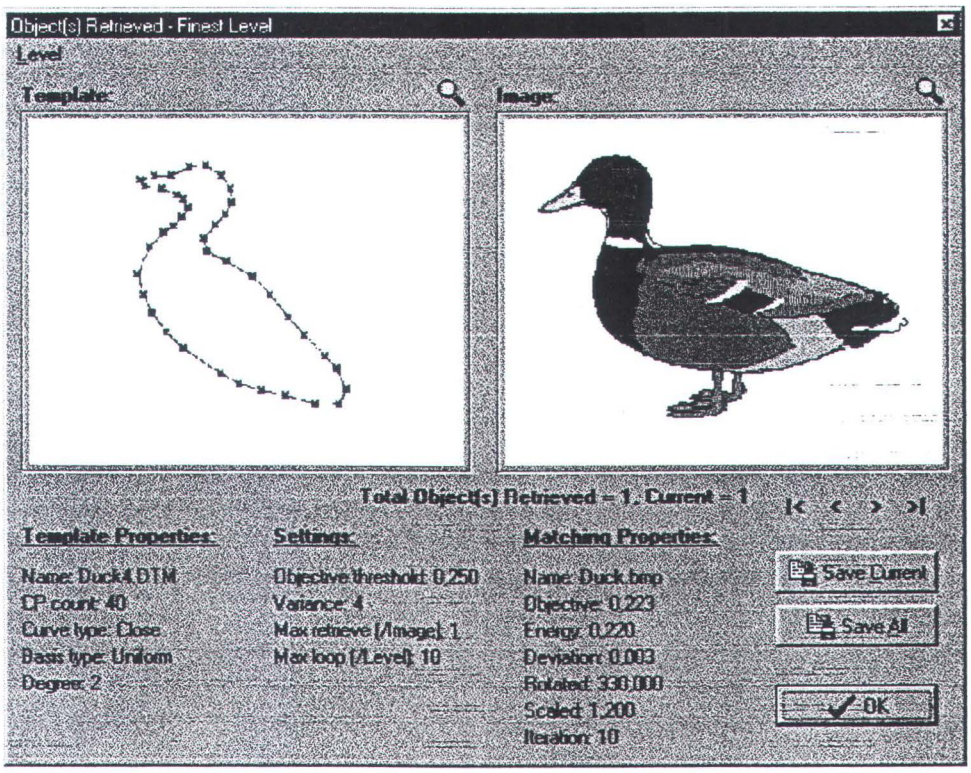
Perangkat lunak yang dibuat memiliki tampilan utama seperti Gambar L.1. Dimana dalam tampilan tersebut terdapat dua daerah yang merupakan daerah tampilan untuk citra masukan yaitu daerah sebelah kiri dan daerah tampilan template yaitu daerah sebelah kanan dengan ukuran yang lebih kecil (*stretch*). Disamping itu perangkat lunak ini memiliki beberapa menu dan tombol yang berguna untuk mempermudah pengguna dalam menggunakan perangkat lunak ini. Untuk beberapa fasilitas menu pilihan akan dijelaskan selanjutnya. Sedangkan beberapa tombol yang ada adalah: *Run*, *Result*, *Exit*, *Edit*, dan *Full View Template*. Fungsi dari masing-masing tombol tersebut adalah sebagai berikut:

- **Tombol *Run***: Digunakan untuk menjalankan perangkat lunak ini, yaitu proses pencocokan antara template dengan citra masukan.
- **Tombol *Result***: Digunakan untuk menampilkan antarmuka hasil dari proses pencocokan antara template dengan citra masukan (Lihat Gambar L.2). Dimana daerah sebelah kiri adalah daerah untuk menampilkan template yang digunakan dan daerah sebelah kanan adalah daerah untuk menampilkan hasil proses pencocokan. Hasil proses pencocokan yang dapat dilihat sesuai dengan menu *Level* (*Coarsest*, *Finer*, atau *Finest*) yang dipilih (misalnya: jika dipilih menu *Level Finest*, maka hasil pencocokan yang ditampilkan adalah pada tingkatan *finest*). Selain fasilitas menu, juga terdapat fasilitas tombol *Full View* untuk template maupun hasil pencocokan yang terletak di sebelah kanan-atas masing-masing daerah tampilan template dan hasil pencocokan. Hasil pencocokan dapat disimpan kedalam bentuk

ITS

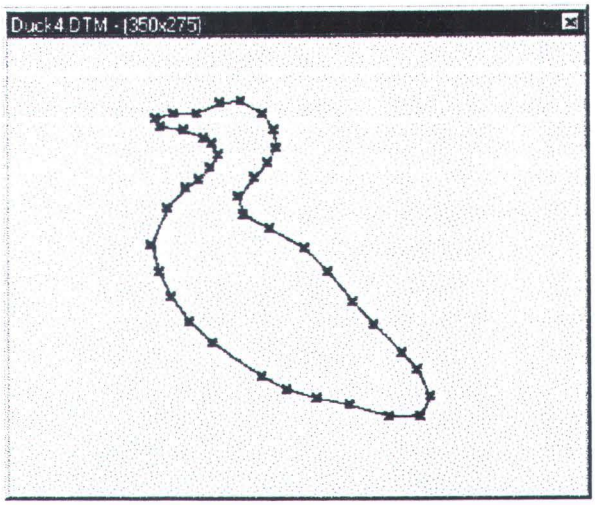


bitmap dengan menggunakan tombol *Save Current* atau *Save All*. Tombol *Save Current* digunakan untuk menyimpan hasil pencocokan pada indek sekarang yang sedang ditunjuk. Sedangkan tombol *Save All* digunakan untuk menyimpan semua hasil pencocokan pada sebuah citra masukan.




Gambar L.2: Tampilan antarmuka hasil proses pencocokan antara citra masukan dengan template yang dihasilkan oleh perangkat lunak ini.

- **Tombol *Exit***: Digunakan untuk keluar dari perangkat lunak ini.
- **Tombol *Edit***: Digunakan untuk mengedit template.
- **Tombol *Full View Template***: Digunakan untuk melihat template dengan ukuran yang sebenarnya. Tombol *Full View Template* terletak disebelah kanan atas daerah tampilan template. Gambar L.3 menunjukkan template dengan ukuran yang sebenarnya.

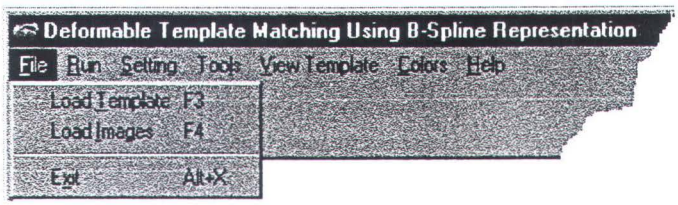



Gambar L.3: Tampilan template dengan ukuran yang sebenarnya.

## L.2 Fasilitas Menu

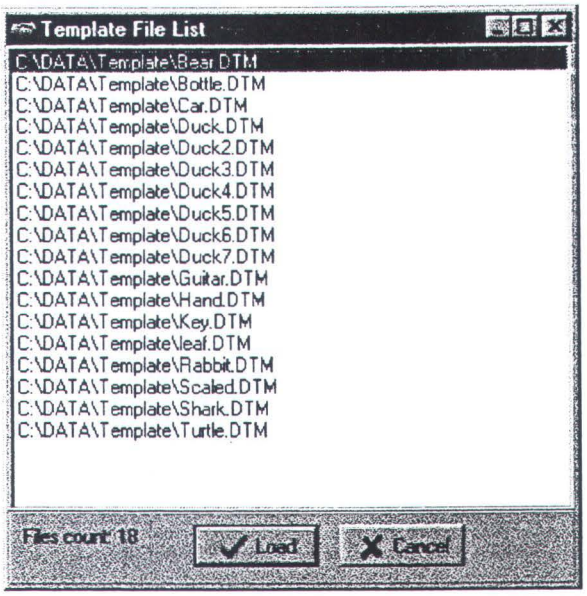
Fasilitas menu yang dimiliki oleh perangkat lunak ini adalah menu: *File*, *Run*, *Setting*, *Tools*, *View Template*, *Colors*, dan *Help*.

- **Menu *File***: Dalam menu *File* terdapat tiga item menu (Lihat Gambar L.4), yaitu: *Load Template*, *Load Images*, dan *Exit*.



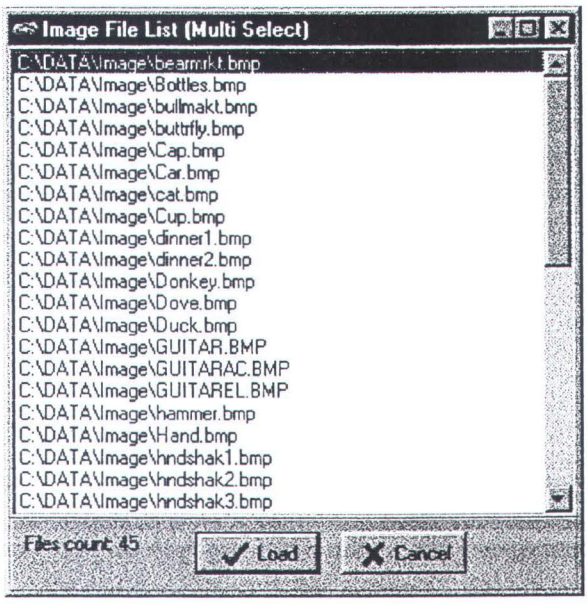

Gambar L.4: Tampilan menu *File*.

- **Item *Load Template***: Digunakan untuk menampilkan antarmuka pemilihan template yang akan digunakan untuk proses pencocokan dari suatu *file list template* (Lihat Gambar L.5). *ShortCut* F3.




Gambar L.5: Tampilan antarmuka *file list template*.

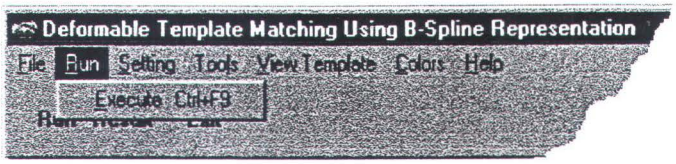
- **Item *Load Images***: Digunakan untuk menampilkan antarmuka pemilihan citra masukan (bisa lebih dari satu) yang akan digunakan untuk proses pencocokan dari suatu *file list image* (Lihat Gambar L.6). *ShortCut* F3.




Gambar L.6: Tampilan antarmuka *file list image*.

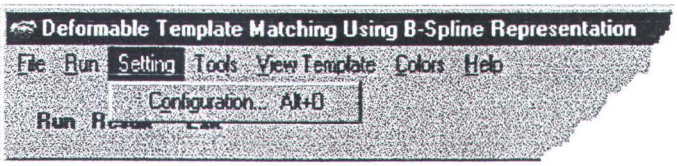
- **Item *Exit***: Fungsinya sama dengan tombol *Exit*. *ShortCut* Alt+X.

❖ **Menu *Run***: Menu *Run* hanya memiliki satu item menu *Execute*. Fungsi dari item *Execute* ini sama dengan tombol *Run* (Lihat Gambar L.7). *ShortCut* Ctrl+F9.

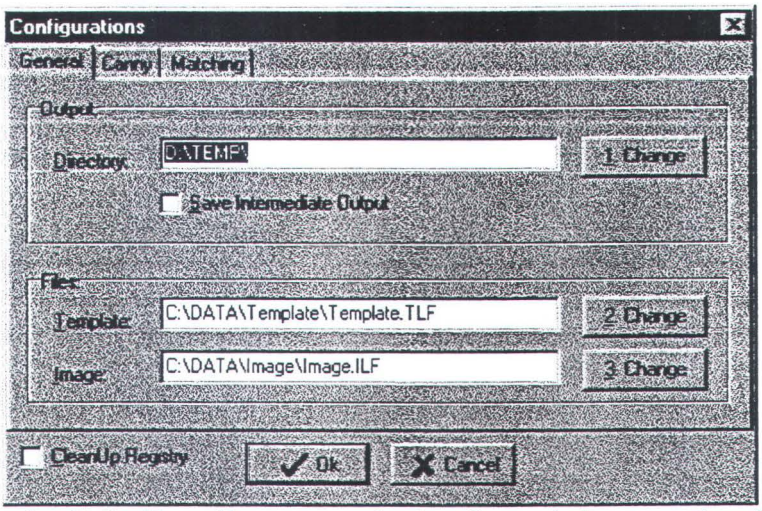



Gambar L.7: Tampilan menu *Run*.

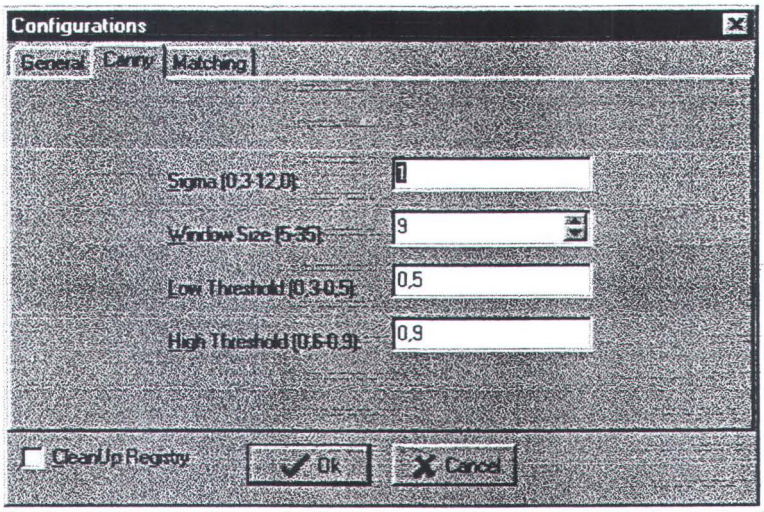
❖ **Menu *Setting***: Menu *Setting* juga hanya memiliki satu item menu yaitu *Configuration* (Lihat Gambar L.8). Fungsi dari item *Configuration* ini adalah untuk menampilkan antarmuka pengaturan konfigurasi *directory output*, *file list template*, dan *file list image* (Lihat Gambar L.9). Juga digunakan untuk menampilkan antarmuka pengesetan parameter-parameter *Canny* (Lihat Gambar L.10) dan antarmuka pengesetan parameter-parameter pencocokan (Lihat Gambar L.11).



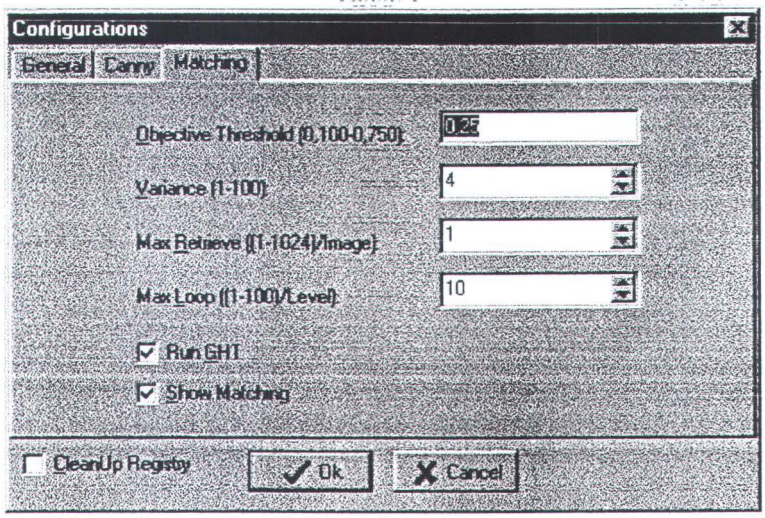

Gambar L.8: Tampilan menu *Setting*.




Gambar L.9: Tampilan antarmuka konfigurasi *directory* dan *file*.

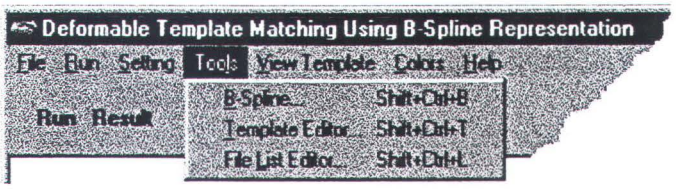



Gambar L.10: Tampilan antarmuka konfigurasi parameter-parameter *Canny*.



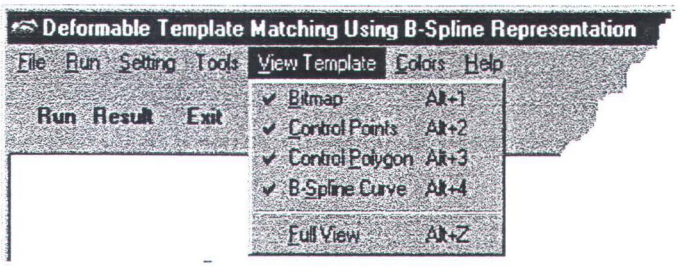

Gambar L.11: Tampilan antarmuka konfigurasi parameter-parameter pencocokan.

❖ **Menu *Tools***: Menu *Tools* memiliki tiga item menu (Lihat Gambar L.12), yaitu: *B-Spline*, *Template Editor*, dan *File List Editor*. Fungsi dari masing-masing tool tersebut akan dijelaskan pada penjelasan khusus selanjutnya.




Gambar L.12: Tampilan menu *Tools*.

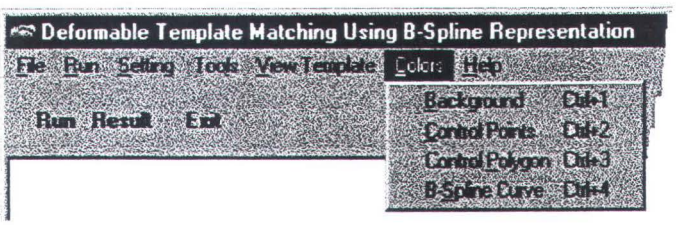
❖ **Menu *View Template***: Menu *View Template* memiliki lima item menu (Lihat Gambar L.13), yaitu: *Bitmap*, *Control Points*, *Control Polygon*, *B-Spline Curve*, dan *Full View*. Fungsi dari masing-masing item menu (selain item *Full View*) adalah untuk menampilkan atau menyembunyikan tampilan template sesuai dengan nama item menu tersebut (*ShortCut* Alt+1 sampai Alt+4). Sedangkan item menu *Full View* (*ShortCut* Alt+Z) fungsinya sama dengan tombol *Full View Template*.




Gambar L.13: Tampilan menu *View Template*.

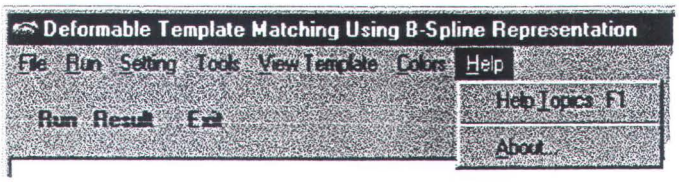
❖ **Menu *Colors***: Menu *Colors* memiliki empat item menu (Lihat Gambar L.14), yaitu: *Background*, *Control Points*, *Control Polygon*,

dan *B-Spline Curve*. Fungsi dari masing-masing item menu tersebut adalah untuk mengganti warna template sesuai dengan nama item menu tersebut. *ShortCut* Ctrl+1 sampai Ctrl+4.

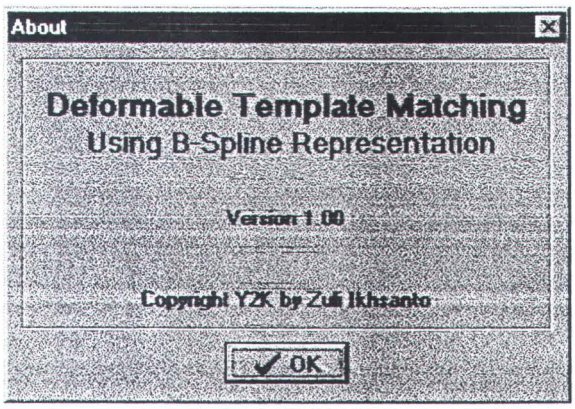



Gambar L.14: Tampilan menu *Colors*.

❖ **Menu *Help***: Menu *Help* memiliki dua item menu (Lihat Gambar L.15), yaitu: *Help Topics*, dan *About*. Fungsi dari item menu *Help Topics* (*ShortCut* F1) adalah untuk menampilkan bantuan dari perangkat lunak ini. Sedangkan fungsi dari item menu *About* adalah untuk menampilkan tentang identitas perangkat lunak ini (Lihat Gambar L.16).




Gambar L.15: Tampilan menu *Help*.

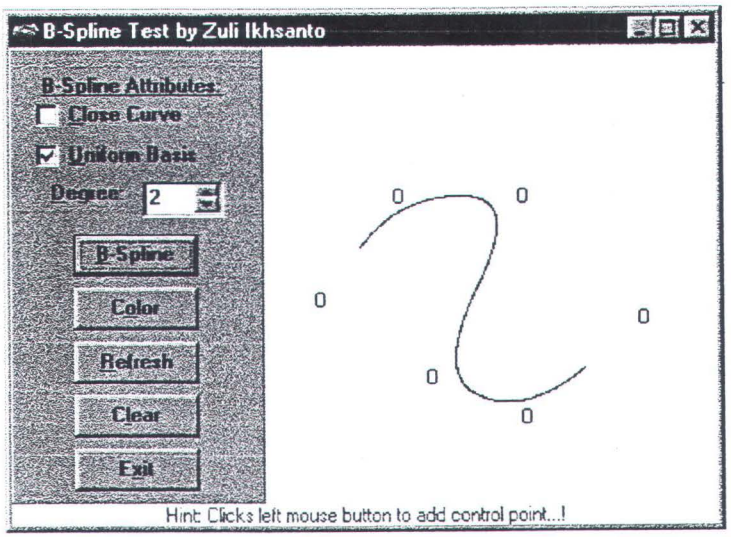



Gambar L.16: Tampilan *About*.

Selanjutnya akan diberikan panduan tentang cara penggunaan item menu *Tools*, yaitu: *B-Spline*, *Template Editor*, dan *File List Editor*.

## L.3 B-Spline

Item menu *B-Spline* digunakan untuk menampilkan antarmuka proses uji coba pembentukan kurva *B-Spline* (Lihat Gambar L.17). Dimana pengguna dapat melakukan uji coba pembentukan kurva *B-Spline* dengan menggunakan parameter-parameter yang berbeda. Parameter-parameter yang dimaksud adalah *tipe kurva*, *tipe basis*, dan *derajat* kurva *B-Spline* yang digunakan.



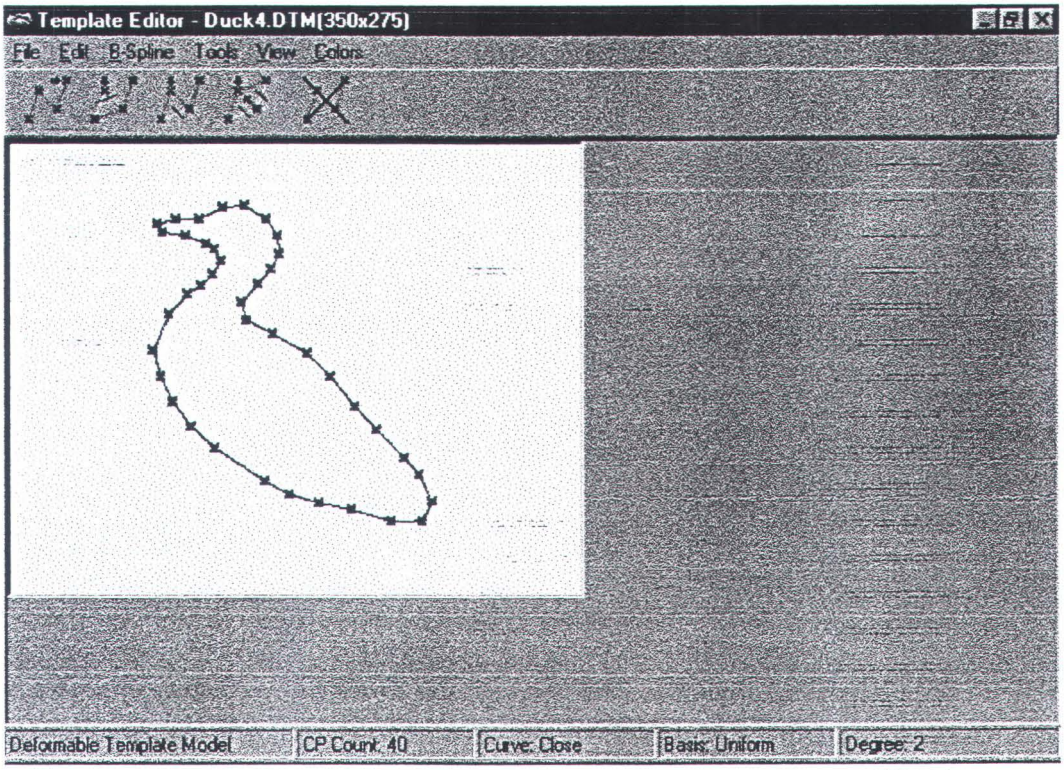

Gambar L.17: Tampilan antarmuka untuk uji coba pembentukan kurva *B-Spline*.

Cara membuat kurva *B-Spline* adalah dengan cara meletakkan beberapa *control point* pada daerah yang berwarna putih (kanan) dengan menggunakan tombol kiri *mouse*. Untuk menggambar kurva *B-Spline* dari set *control point* tersebut dilakukan dengan cara menekan tombol *B-Spline* yang terdapat di sebelah kiri daerah penggambaran. Warna kurva *B-Spline* yang diinginkan dapat diset dengan menggunakan tombol *Color*. Sedangkan tombol *Refresh* berguna untuk

menampilkan kurva *B-Spline* yang terakhir saja. Tombol *Clear* digunakan untuk menghapus set *control point*. Dan untuk keluar dari uji coba pembuatan kurva *B-Spline* dapat dilakukan dengan menekan tombol *Exit*.

## L.4 Editor Template

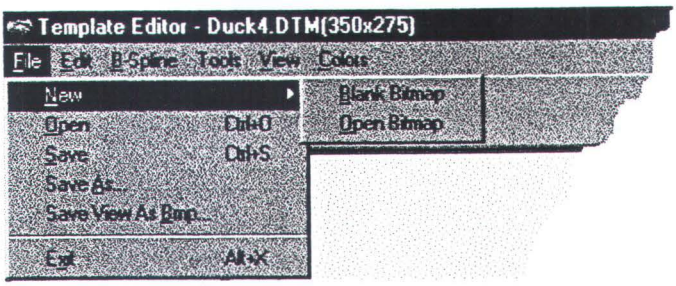
Item menu *Template Editor* digunakan untuk menampilkan antarmuka Editor Template (Lihat Gambar L.18).




Gambar L.18: Tampilan antarmuka Editor Template.

Beberapa fasilitas menu yang dimiliki oleh Editor Template ini yaitu: *File*, *Edit*, *B-Spline*, *Tools*, *View*, dan *Colors*. Fungsi dari masing-masing menu tersebut adalah sebagai berikut:

❖ **Menu *File***: Dalam menu *File* terdapat enam item menu (Lihat Gambar L.19), yaitu: *New*, *Open*, *Save*, *Save As*, *Save View As Bmp*, dan *Exit*.

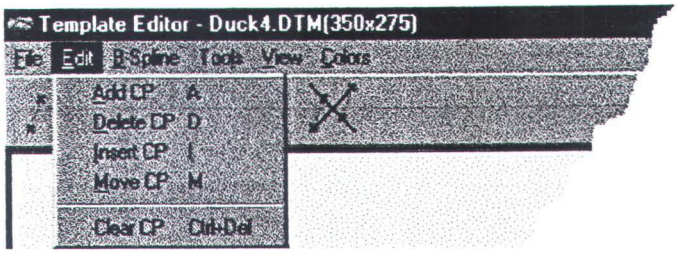



Gambar L.19: Tampilan menu *File* dalam Editor Template.

- **Item *New***: Dalam item menu *New* memiliki dua sub item menu, yaitu: *Blank Bitmap* dan *Open Bitmap*. Sub item menu *Blank Bitmap* digunakan untuk membuat template yang baru dengan bitmap kosong dan ukuran bitmap yang dapat ditentukan. Sedangkan sub item menu *Open Bitmap* digunakan untuk membuat template yang baru dengan menggunakan prototipe bitmap yang sudah ada dari suatu file bitmap (.BMP).
- **Item *Open***: Digunakan untuk membuka file template yang sudah ada. Dimana file template terdiri dari dua file, yaitu: file prototipe bitmap (.BMP) dan file attribut kurva *B-Spline* (.DTM). *ShortCut* Ctrl+O.
- **Item *Save***: Digunakan untuk menyimpan template. *ShortCut* Ctrl+S.
- **Item *Save As***: Digunakan untuk menyimpan template dengan nama yang baru.

- **Item *Save View As Bmp***: Digunakan untuk menyimpan tampilan template ke dalam bentuk file bitmap (.BMP).
- **Item *Exit***: Digunakan untuk keluar dari Editor Template. *ShortCut* Alt+X.

❖ **Menu *Edit***: Dalam menu *Edit* terdapat lima item menu (Gambar L.20), yaitu: *Add CP*, *Delete CP*, *Insert CP*, *Move CP*, dan *Clear CP*.



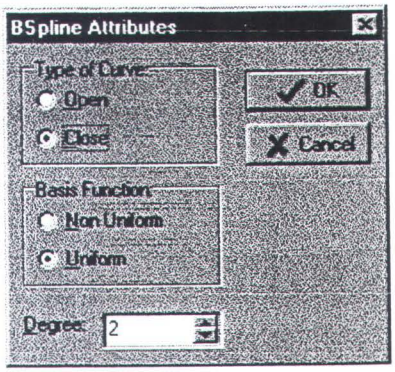

Gambar L.20: Tampilan menu *Edit* dalam Editor Template.

- **Item *Add CP***: Digunakan untuk menambah *control point* pada posisi terakhir. Fungsinya sama dengan tombol *Add CP*. *ShortCut* A.
- **Item *Delete CP***: Digunakan untuk menghapus sebuah *control point*. Fungsinya sama dengan tombol *Delete CP*. *ShortCut* D.
- **Item *Insert CP***: Digunakan untuk menyisipkan *control point* diantara dua *control point* yang sudah ada. Fungsinya sama dengan tombol *Insert CP*. *ShortCut* I.
- **Item *Move CP***: Digunakan untuk memindahkan sebuah *control point* ke posisi yang baru. Fungsinya sama dengan tombol *Move CP*. *ShortCut* M.

- **Item *Clear CP***: Digunakan untuk menghapus semua *control point*. Fungsinya sama dengan tombol *Clear CP*. *ShortCut* Ctrl+Del.

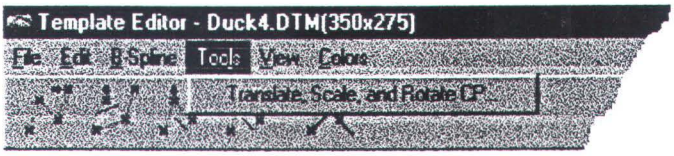
❖ **Menu *B-Spline***: Menu *B-Spline* hanya memiliki satu item menu (Lihat Gambar L.21), yaitu: *Attributes* (*ShortCut* Ctrl+B). Fungsi dari item *Attributes* ini adalah untuk mengatur attribut kurva *B-Spline*, yaitu: *tipe kurva*, *tipe basis*, dan *derajat* yang digunakan (Lihat Gambar L.22).

Gambar L.21: Tampilan menu *B-Spline* dalam Editor Template.

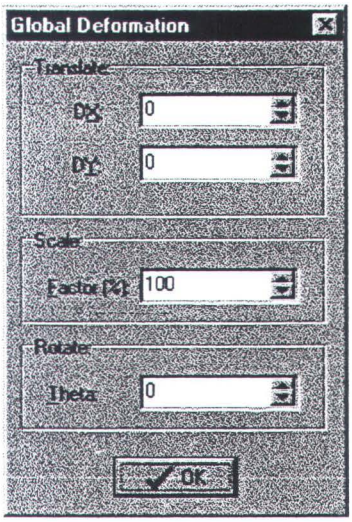



Gambar L.22: Tampilan antarmuka attribut kurva *B-Spline*.

❖ **Menu *Tools***: Menu *Tools* hanya memiliki satu item menu (Lihat Gambar L.23), yaitu: *Translate, Scale, and Rotate CP*. Fungsi dari tool ini adalah untuk mengubah set *control point* secara global dengan proses translasi, perubahan skala, dan rotasi (Lihat Gambar L.24).

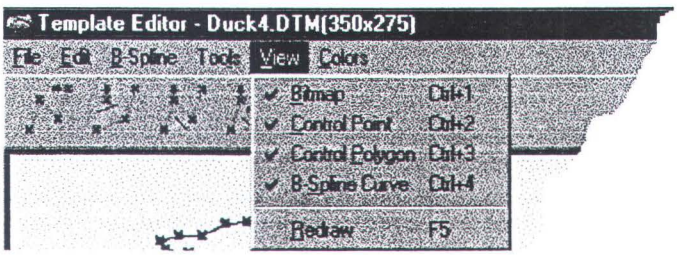



Gambar L.23: Tampilan menu *Tools* dalam Editor Template.



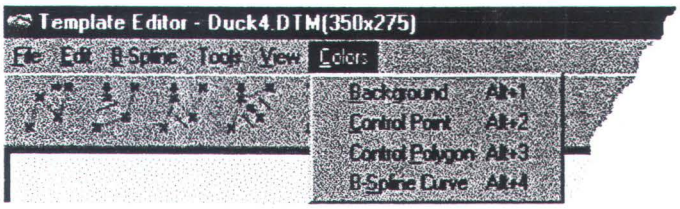

Gambar L.24: Tampilan antarmuka masukan untuk proses deformasi template secara global.

❖ **Menu *View***: Dalam menu *View* terdapat lima item menu (Lihat Gambar L.25), yaitu: *Bitmap*, *Control Point*, *Control Polygon*, *B-Spline Curve*, dan *Redraw*. Fungsi item menu satu sampai empat sama dengan item menu *View Template* satu sampai empat dalam antarmuka utama. Sedangkan item menu *Redraw* digunakan untuk mengupdate tampilan template (*ShortCut* F5).




Gambar L.25: Tampilan menu *View* dalam Editor Template.

❖ **Menu *Colors***: Fungsinya sama dengan menu *Colors* dalam antarmuka utama (Lihat Gambar L.26).



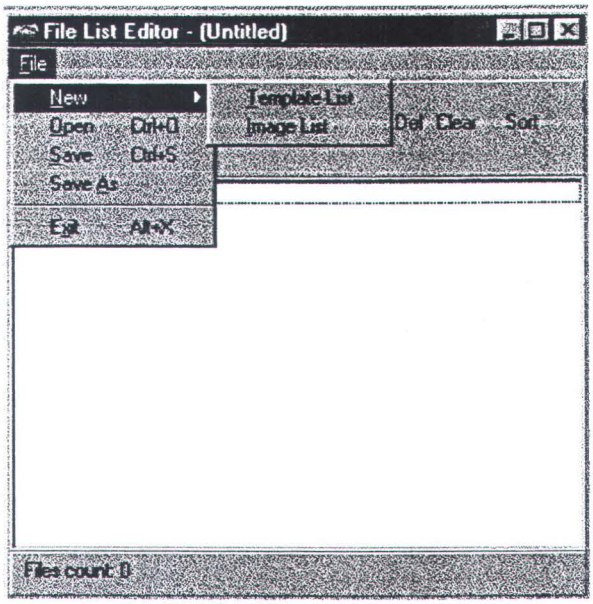

Gambar L.26: Tampilan menu *Colors* dalam Editor Template.

## L.5 Editor File List

Editor File List digunakan untuk mengedit *file list template* dan *file list image*. Dalam Editor File List ini terdapat sebuah menu *File* dengan beberapa itemnya (Lihat Gambar L.27), yaitu: *New*, *Open*, *Save*, *Save As*, dan *Exit*. Fungsi dari masing-masing item menu tersebut adalah sebagai berikut:

❖ **Item *New***: Dalam menu *New* terdapat dua sub item menu, yaitu: *Template List* dan *Image List*.

- Sub item *Template List*: Digunakan untuk membuat *file list template* baru (.TLF).
- Sub item *Image List*: Digunakan untuk membuat *file list image* baru (.ILF).




Gambar L.27: Tampilan menu *File* dalam Editor File List.

- **Item *Open***: Digunakan untuk membuka *file list* baik *file list template* (.TLF) maupun *file list image* (.ILF).
- **Item *Save***: Digunakan untuk menyimpan *file list* baik *file list template* (.TLF) maupun *file list image* (.ILF).
- **Item *Save As***: Digunakan untuk menyimpan *file list* baik *file list template* (.TLF) maupun *file list image* (.ILF) dengan nama yang baru.
- **Item *Exit***: Digunakan untuk keluar dari Editor File List.

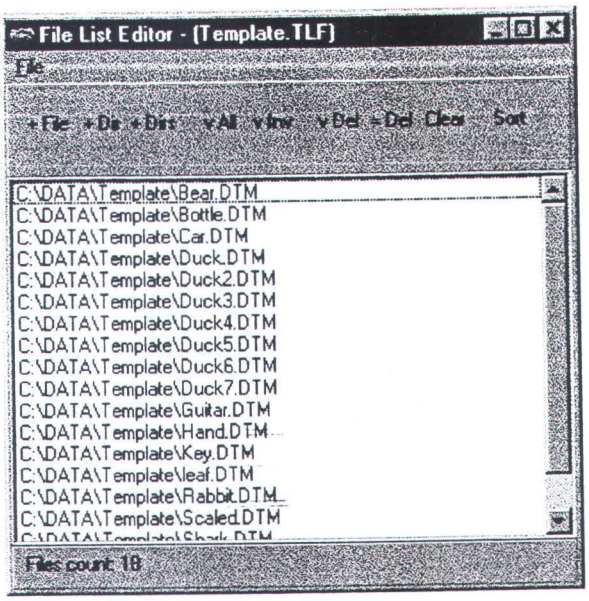
Disamping menu *File* tersebut dalam Editor File List juga disediakan fasilitas berupa tombol-tombol (Lihat Gambar L.28), yaitu: + *File*, + *Dir*, + *Dirs*, *v All*, *v Inv*, *v Del*, = *Del*, *Clear*, dan *Sort*. Fungsi dari masing-masing tombol tersebut adalah sebagai berikut:

- **Tombol + *File***: Digunakan untuk menambah nama-nama *file template* atau *file image* ke dalam *file list template* atau *file list image*.

ITS



- **Tombol** + ***Dir***: Digunakan untuk menambah nama-nama *file template* atau *file image* dari suatu *directory* ke dalam *file list template* atau *file list image*.
- **Tombol** + ***Dirs***: Digunakan untuk menambah nama-nama *file template* atau *file image* dari suatu *directory* termasuk *directory-directory* didalamnya ke dalam *file list template* atau *file list image*.
- **Tombol** ***v All***: Digunakan untuk memilih semua nama-nama *file list*.
- **Tombol** ***v Inv***: Digunakan untuk memilih semua nama-nama *file list* yang tidak dipilih sebelumnya.
- **Tombol** ***v Del***: Digunakan untuk menghapus semua nama-nama *file list* yang dipilih.
- **Tombol** = ***Del***: Digunakan untuk menghapus nama-nama *file list* yang sama dengan nama-nama *file list* sebelumnya.
- **Tombol** ***Clear***: Digunakan untuk menghapus semua nama-nama *file list*.
- **Tombol** ***Sort***: Digunakan untuk mengurutkan semua nama-nama *file list* dengan urutan menaik.




Gambar L.28: Tampilan antarmuka Editor File List.