IMPLEMENTASI SEGMENTASI WARNA PADA CITRA DIGITAL MENGGUNAKAN METODE GRAPH COLORING UNTUK PENGENALAN POLA

SKRIPSI

Disusun Oleh :

HENDRY KUSNOYO
NIM. 05.12.723

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNIK INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

2011

LEMBAR PERSETUJUAN

IMPLEMENTASI SEGMENTASI WARNA PADA CITRA DIGITAL MENGGUNAKAN METODE GRAPH COLORING UNTUK PENGENALAN POLA

SKRIPSI

Disusun dan diajukan untuk melengkapi dan memenuhi persyaratan guna mencapai gelar Sarjana Teknik

Disusun Oleh :

Hendry Kusnoyo

NIM. 05.12.723

Mengetahui,	Diperiksa dan Disetujui,
Ketua Jurusan Teknik Elektro S-1	Dosen Pembimbing
Ir. Yusuf Ismail Nakhoda, MT	Dr. Eng. Aryuanto Soetedjo, ST, MT
NIP.Y.1018800189	NIP.P.1030800417

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

ABSTRAK

IMPLEMENTASI SEGMENTASI WARNA PADA CITRA DIGITAL MENGGUNAKAN METODE GRAPH COLORING UNTUK PENGENALAN POLA

Hendry Kusnoyo, Nim. 05.12.723

Dosen Pembimbing : Dr. Eng. Aryuanto Soetedjo, ST, MT

Saat ini,pengolahan citra dapat lebih mudah dilakukan dengan menggunakan pengolahan citra digital. Salah satu operasi dari pengolahan citra digital adalah proses segmentasi citra. Segmentasi citra merupakan proses memisahkan objek-objek dari citra ke dalam bagian-bagian,sehingga informasi yang ada pada citra dapat dengan mudah dibaca. Graph coloring adalah salah satu metode dalam segmentasi citra. Metode graph coloring ini mendefinisikan predikat dari sebuah batas antara dua region yang bersinggungan pada sebuah citra dengan menggunakan representasi berbasis graph coloring. Kemudian mengembangkan algoritma segmentasi berbasis graph coloring dengan menggunakan dua model ketetanggaan yang berbeda. Preprocessing sebaiknya dilakukan sebelum melakukan proses segmentasi citra.

Pengenalan pola (pattern recognition) merupakan salah satu cabang ilmu computer yang dapat diartikan sebagai pengumpulan data-data mentah untuk dapat diklasifikasikan dengan maksud dan tujuan tertentu. Pengenalan pola ini bersifat conceptually driven processing yang berarti bahwa proses dimulai dari pembentukan konsep pada objek yang dijumpai (informasi dari memori). Ada banyak aplikasi yang bisa dijadi implementasi dari pengenalan pola, diantaranya adalah pengenalan wajah pada manusia, pengenalan tulisan tangan, pengenalan penyakit berdasarkan gejala-gejala yang ditemukan pada sebuah objek. Deteksi wajah (face detection) adalah salah satu tahap awal yang sangat penting dalam sistem pengenalan wajah (face recognition) yang digunakan dalam identifikasi biometric. Deteksi wajah juga dapat digunakan untuk pencarian atau pengideksan data wajah dari citra atau video yang berisikan wajah dengan berbagi ukuran, posisi, dan latar belakang. Informasi warna kulit dengan metode template metching.

Kata Kunci: Segmentasi, Metode Graph Coloring, Pengenalan Pola

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Segmentasi citra (image segmentation) adalah suatu tahap pada proses analisis citra yang bertujuan untuk memperoleh informasi yang ada pada citra tersebut dengan membagi citra ke dalam daerah-daerah terpisah dimana setiap daerah adalah homogen dan mengacu pada sebuah kriteria keseragaman yang jelas. Proses segmentasi citra merupakan proses dasar dan penting di dalam komputer visi. Terdapat banyak sekali metode dalam melakukan segmentasi pada citra. Salah satu metode yang dapat digunakan dalam proses segmentasi citra adalah metode graph coloring. Dimana metode graph coloring ini mendefinisikan predikat dari sebuah batas antara dua region yang bersinggungan pada sebuah citra dengan menggunakan representasi barbasis graph coloring. Kemudian mengembangkan algoritma segmentasi berbasis graph coloring dengan menggunakan dua model ketetanggaan yang berbeda.

Ada banyak aplikasi yang bisa dijadi implementasi dari pengenalan pola, diantaranya adalah pengenalan wajah pada manusia, pengenalan tulisan tangan, pengenalan penyakit berdasarkan gejala-gejala yang ditemukan pada sebuah objek. Pengenalan wajah semakin banyak diaplikasikan dalam sistem pengenalan biometrik, pencarian dan pengindeksian database citra dan video digital, sitem keamanan, konferansi video, dan interaksi manusia dengan komputer. Pendeteksian wajah (face detection) merupakan salah satu tahapan awal yang sangat penting sebelum dilakukan proses pengenalan wajah (fece recognition).

Masalah deteksi wajah dapat dirumuskan sebagai berikut : diberikan masukan sebuah citra digital sembarang, maka sistem akan mendeteksi apakah ada wajah manusia atau tidak di dalam citra tersebut. Jika ada maka sistem akan memberitahu berapa jumlah wajah yang ditemukan dan dimana lokasi wajah-wajah tersebut di dalam citra. Keluaran dari sistem adalah posisi subcitra berisi wajah-wajah yang berhasil dideteksi.

Berdasarkan uraian diatas penulis ingin mengimplementasikan segmentasi warna menggunakan metode graph coloring untuk pengenalan pola. Yang mana hasil dari segmentasi menggunakan metode graph coloring tersebut akan digunakan untuk melakukan pengenalan pola. Dimana pengenalan pola hanya sebatas pengenalan wajah manusia.

1.2 RUMUSAN MASALAH

Berdasarkan latar belakang diatas, maka rumusan masalah yang dapat diambil dalam skripsi ini adalah bagaimana menerapkan metode graph coloring dalam proses segmentasi yang akan digunakan untuk pengenalan suatu pola pada citra digital.

1.3 BATASAN MASALAH

Batasan masalah yang digunakan diharapkan mampu membatasi pembahasan. Agar dalam pembahasan dan pemecahan permasalahan skripsi ini tidak melebar, maka pembahasan skripsi ini dibatasi oleh hal – hal berikut:

1. Pengenalan pola yang dibahas hanya untuk pengenalan wajah manusia

2. Gambar yang digunakan adalah citra digital dengan format .jpg dan berwarna

3. Model wajah yang digunakan sebagai tamplate diambil dengan posisi menghadap kedepan/frontal dan tidak terhalang oleh objek lain.

1.4 TUJUAN PENELITIAN

Tujuan pembuatan skripsi ini adalah untuk menerapkan metode graph pada proses segmentasi yang digunakan untuk pengenalan pola.

1.5 METODOLOGI

1.5.1 Studi Literatur

Kajian tentang pengolahan citra, segmentasi citra, algoritma berbasis graph coloring, analisis dan perancangan system segmentasi untuk pengenalan pola pada citra digital dengan metode graph coloring.

1.5.2 Metode Perancangan Sistem

a) *Analisa.* Pada tahap ini dilakukan pengumpulan fakta-fakta yang mendukung perancangan system.

b) *Perancangan dan implementasi.* Proses perancangan dan implementasi segmentasi untuk pengenalan pola pada citra digital didasarkan pada metode graph coloring.

c) *Uji Analisa.* Bertujuan untuk mengetahui hasil segmentasi untuk pengenalan pola pada citra digital yang didasarkan pada metode graph coloring.

1.6 SISTEMATIKA PENULISAN

Disusun Sistematika penulisan Tugas Akhir yang terbagi atas 5 bab, yakni:

BAB I.PENDAHULUAN

Berisikan tentang Latar Belakang, Rumusan Masalah, Batasan Masalah, Tujuan Penulisan, Manfaat Tugas Akhir dan Sistematika Penulisan.

BAB II.LANDASAN TEORI

Berisikan tinjauan umum teori – teori yang mendasari *sementasi citra, metode graph coloring* dan *pengenalan pola*

BAB III.METODE DAN PERANCANGAN

Berisikan penjelasan metode dan algoritma *graph coloring* yang dipakai untuk *segmentasi* dan *pengenalan pola citra*

BAB IV.IMPLEMENTASI DAN UJI ANALISA

Berisikan pembahasan tentang penerapan *segmentasi* dan *pengenalan pola citra* dan uji analisa yang dilakukan.

BAB V.PENUTUP

Merupakan Bab terakhir yang berisi kesimpulan dan saran.

BAB II

LANDASAN TEORI

2.1 Pengertian Citra Digital

Citra Digital adalah gambaran dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinu menjadi gambar diskrit melalui proses digitalisasi. Citra yang terlihat merupakan cahaya yang direfleksikan oleh sebuah objek.

Citra dapat dikelompokkan menjadi dua yaitu: citra tampak seperti foto/gambar,lukisan dan apa saja yang tampak dilayar monitor/televise, hologram dan sebagainya serta citra yang tidak tampak seperti data foto/gambar dalam file, dan citra yang dipresentasikan dalam fungsi matematis. Ada beberapa macam tipe citra didasarkan dari format penyimpanan warnanya yaitu:

Citra biner adalah citra yang pikselnya hanya bernilai 0(waran hitam) dan 1 (warna putih).

Citra skala keabuan adalah citra yang setiap pikselnya menpunyai kemungkinan waran antara hitam (minimal) dan putih (maksimal).

Citra warna *(true color)* adalah citra yang setiap pikselnya memilki waran yang merupakan kombinasi dari tiga warna dasar, yaitu merah kuning dan hijau *(RGB)*.

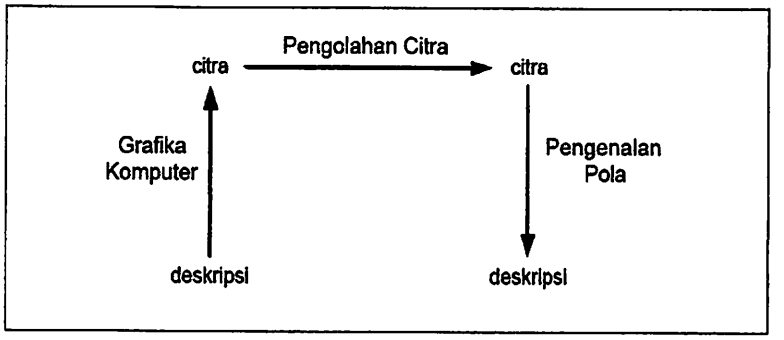
Citra warna berindeks adalah citra yang setiap pikselnya memilki indeks dan suatu table warna yang tersedia (biasanya disebut palet warna).

2.1.1 Pengertian Pengolahan Citra

Meskipun sebuah citra kaya informasi, namun seringkali citra yang kita miliki mengalami penurunan mutu *(degradasi)*, misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, kabur (*blurring*), dan sebagainya. Tentu saja citra semacam ini menjadi lebih sulit diinterpretasi karena informasi yang disampaikan oleh citra tersebut menjadi berkurang. Agar citra yang mengalami gangguan mudah diinterpretasi (baik oleh manusia maupun mesin), maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik. Bidang studi yang menyangkut hal ini adalah pengolahan citra (image processing). Pengolahan citra adalah suatu metode yang digunakan untuk mengolah gambar sehingga menghasilkan gambar lain yang sesuai dengan keinginan kita. Pengambilan gambar biasanya dilakukan dengan kamera *video digital* atau alat lain yang biasanyan digunakan untuk mentransfer gambar (*scanner*, kamera *digital*).

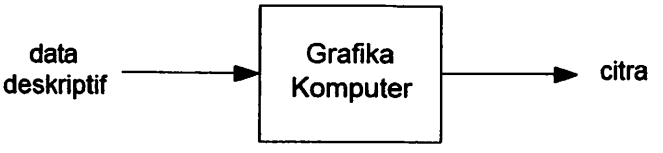
Di dalam bidang komputer, sebenarnya ada tiga bidang studi yang berkaitan dengan data citra, namun tujuan ketiganya berbeda, yaitu:

1. Grafika Komputer (*computer graphics*).
2. Pengolahan Citra (*image processing*).
3. Pengenalan Pola (*pattern recognition/image interpretation*).



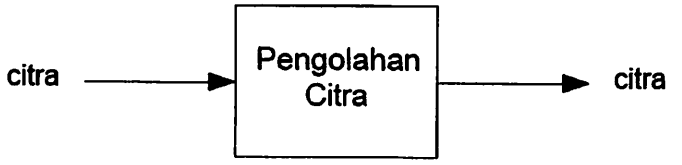
Gambar 2.1 Tiga bidang yang berkaitan dengan Citra

Grafika Komputer bertujuan menghasilkan citra (lebih tepat disebut grafik atau picture) dengan primitif-primitif geometri seperti garis, lingkaran, dan sebagainya. Primitif-primitif geometri tersebut memerlukan data deskriptif untuk melukis elemen-elemen gambar. Contoh data deskriptif adalah koordinat titik, panjang garis, jari-jari lingkaran, tebal garis, warna, dan sebagainya. Grafika computer memainkan peranan penting dalam visualisasi dan virtual reality.



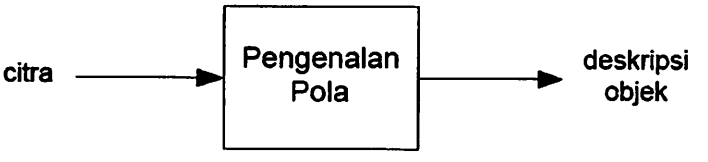
Gambar 2.2 Grafika Komputer

Pengolahan Citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih baik daripada citra masukan. Termasuk ke dalam bidang ini juga adalah pemampatan citra (image compression).



Gambar 2.3 Prngolahan Citra

Pengenalan Pola mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini komputer). Tujuan pengelompokan adalah untuk mengenali suatu objek di dalam citra. Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek-objek di alam sehingga mampu membedakan suatu objek dengan objek lainnya. Kemampuan sistem visual manusia inilah yang dicoba ditiru oleh mesin. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut, dan memberikan keluaran berupa deskripsi objek di dalam citra.



Gambar 2.4 Pengenalan Pola

2.1.2 Operasi Pengolahan Citra

Operasi-operasi yang dilakukan di dalam pengolahan citra banyak ragamnya. Namun, secara umum,operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut:

1. Perbaikan kualitas citra *(image enhancement).*

Contoh-contoh operasi perbaikan citra:

a. perbaikan kontras gelap/terang

b. perbaikan tepian objek *(edge enhancement)*

c. penajaman *(sharpening)*

d. pembrian warna semu *(pseudocoloring)*

e. penapisan derau *(noise filtering)*

2. Pemugaran citra *(image restoration).*

Operasi ini bertujuan menghilangkan/meminimumkan cacat pada citra. Tujuan pemugaran citra hampir sama dengan operasi perbaikan citra. Bedanya, pada pemugaran citra penyebab degradasi gambar diketahui.

Contoh-contoh operasi pemugaran citra:

a. penghilangan kesamaran *(deblurring).*

b. penghilangan derau *(noise)*

3. Kompresi citra *(image compression).*

Jenis operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam kompresi adalah citra yang telah dikompresikan harus tetap mempunyai kualitas gambar yang bagus.

4. Segmentasi citra *(image segmentation).*

Jenis operasi ini bertujuan untuk memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengenalan pola.

5. Analisis citra *(image analysis)*

Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Teknik pengorakan citra mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek. Proses segmentasi kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya.

Contoh-contoh operasi pengorakan citra:

a. Pendeteksian tepi objek *(edge detection)*

b. Ekstraksi batas *(boundary)*

c. Representasi daerah *(region)*

6. Rekonstruksi citra *(image reconstruction)*

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

2.1.3 Proses digitalisasi citra

Agar suatu citra dapat diolah dengan computer digital, maka citra harus direpresentasikan secara numeric dengan nilai-nilai diskrit melalui proses digitalisasi citra.digitalisasi citra merupakan representasi citra dari fungsi kontinu menjadi nilai-nilai diskrit. Dimana citra kontinu dihasilkan dari sumber cahaya yang menyinari objek, kemudian objek memantulkan kembali sebagian dari berkas cahaya tersebut dan pantulan cahaya ini ditangkap oleh sensor visual pada

system optic, misalnya manusia dan kamera analog. Adapun proses digitalisasi citra terdiri atas dua tahap yaitu:

Digitalisasi *spasial (x,y)*, sering disebut sebagai sampling, merupakan proses pengambilan nilai diskrit koordinat ruang *(x,y)* dengan melewatkan citra melalui grid (celah)

Digitalisasi intensitas *f(x,y)*, sering disebut sebagai kuantisasi, merupakan proses pengelompokkan nilai tingkat keabuan citra kontinu ke dalam beberapa level atau merupakan proses membagi skala keabuan *(0,L)* menjadi *G* buah level yang dinyatakan dengan suatu harga bilangan bulat *(integer)*, dinyatakan sebagai

$$G = 2^m$$

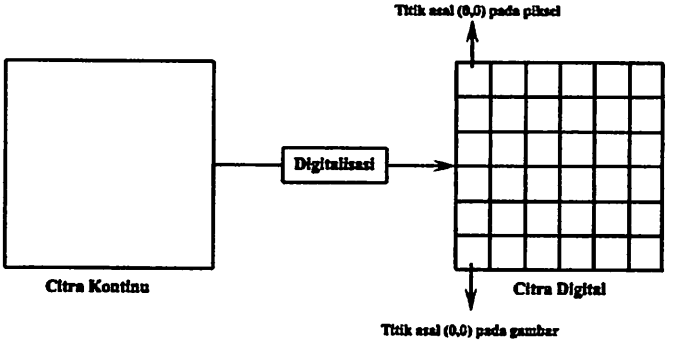
Nilai maksimun (dalam indeks) $G = 2^m - 1$

G : derajat keabuan,

m : bil bulat positif

dari skala keabuan (0'L). Nilai intensitas keabuan 0 menyatakan hitam, nilai intensitas L menyatakan putih, dan nilai intensitas antara 0 sampai L bergeser dari hitam sampai putih. Hitam dinyatakan dengan nilai skala keabuan terendah, sedangkan putih dinyatakan dengan nilai dklala keabuan tertinggi. Jumlah bit yang dibutuhkan untuk mempresentasikan nilai skala keabuan piksel disebut kefgalaman piksel (pixel deph). Citra sering diasosiasiak dengan kedalaman pikselnya. Jadi citra dengan kedalaman 8 bit disebut juga citra 256 warna skala keabuan. Biasanya daerah dkala keabuan yang digunakan menentukan resolusi deverahan dari gambat yang diperoleh. Semakin banyak jumlah skala keabuan,

berarti jumlah bit intensitasnya semakin banyak dan gambar yang diperoleh semakin bagus karena skala keabuannya akan semakin tinggi sehingga mendekati citra asli.



Gambar 2.5 Digitalisasi

2.2 Metode Graph Coloring

2.2.1 Pengertian graph Coloring

Graph adalah salah satu pokok bahsan matematika diskrit yang telah lama dikenal dan diaplikasikan pada berbagai bidang. Secara umum graph *G* di definisikan sebagai pasangan himpunan *(V,E)* di tulis dengan notasi *G=(V,E)* yang dalam hal ini *V* adalah himpunan tidak kosong dari simpul-simpul *(nodes)* dan *E* adalah himpunan sisi/busur *(edges)* yang menghubungkan sepasang simpul.

Kegunaan graph coloring sangat banyak. Umumnya graph coloring digunakan untuk memodelkan suatu masalah sehingga menjadi lebih mudah, yaitu dengan cara mempresentasikan objek-objek tersebut.

2.2.2 Macam-macam graph coloring

Graph coloring dapat dikelompokkan menjadi beberapa jenis tergantung dari sudut pandang pengelompokannya. Pengelompokan graph coloring dapat

dipandang berdasarkan dari ada tidaknya sisi ganda, berdasarkan jumlah simpul, atau berdasarkan orientasi arah pada sisi.

Berdasarkan ada tidaknya loop atau sisi ganda pada suatu graph, maka secara umum graph digolongkan menjadi dua jenis.yaitu:

1. Graph Coloring sederhana *(simple graph coloring)* adalah graph coloring yang tidak mengandung loop maupun sisi ganda. Contohnya graph yang mempresentasikan jaringan computer.
2. Graph Coloring tidak sederhana *(unsimple graph coloring)* adalah graph coloring yang mengandung sisi ganda ataupun loop. Ada dua macam graph coloring tidak sederhana yaitu graph coloring ganda *(multi graph coloring)* yaitu graph yang mengandung sisi ganda dan graph coloring semu *(pseudo graph coloring)* yaitu graph yang mengandung loop.

Berdasakan sisi pada graph coloring dapat mempunyai orientasi arah. Orientasi arah pada sisi, maka secara umum graph coloring dibedakan atas dua jenis,yaitu:

1. **Graph Coloring tak berarah** ***(undirected graph coloring)*** adalah graph coloring yang sisinya tidak mengandung orientasi arah.
2. **Graph Coloring berarah** ***(directed graph coloring atau digraph coloring)*** adalah graph coloring yang setiap sisinya mempunyai orientasi arah.

Berdasarkan pada sifat Graph Coloring,ada beberapa sifat yang berkaitan dengan graph coloring. Berikut adalah sifat-sifat yang sering digunakan, yaitu:

1. **Bartetangga *(adjacent)***

   Dua buah simpul pada graph coloring tak berarah G dikatakan bertetanggaan bila keduanya terhubung langsung dengan sebuah sisi.

2. **Bersisian *(incident)***

   Untuk sembarang sisi $e=(u,v)$, sisi r dikatakan bersisian dengan simpul u dan simpul v.

3. **Derajat *(degree)***

   Derajat suatu simpul pada graph coloring tak berarah adalah jumlah sisi yang bersisian denga simpul tersebut.

4. **Lintasan *(path)***

   Lintas yang panjangnya n dari simpul awal v_o ke simpul tujuan vn didalam graph G adalah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk $v0,e1,v1,e2,v2,...vn\text{-}1,en,vn$ sedemikian sehingga $e2 = (v0,v1)$, $e2 = (v1,v2)$,.., $en = (vn\text{-}1,vn)$

5. **Sirkuit *(circuit)* atau siklus *(cycle)***

   Lintasan yang berawal dan barakhir pada dimpul yang sama disebut sirkuit atau siklus.

6. **Terhubung *(connected)***

   Graph coloring tak berarah G disebut graph coloring terhubung *(connected graph coloring)* jika untuk setiap pasang simpul u dan v didalam himpunan V terdapat lintasan dari u ke v.

7. Graph Coloring berbobot *(weighted graph coloring)*

Graph Coloring berbobot adalah graph coloring yang setiap sisinya diberikan suatu bobot atau nilai.

Graph coloring terhubung yang tidak mengandung sirkuit disebut pohon *(tree)*. Diantara sekian banyak konsep dalam graph coloring, konsep pohon *(tree)* merupakan konsep yang paling penting, kerena terapannya yang luas daln berbagai bidang ilmu, baik dalm bidang computer maupun diluar bidang computer.

Yang dimaksud dengan pohon merentang (spanning tree) yaitu misalkan $G=(V,E)$merupakan graph coloring tak berarah terhubung yang bukan pohon, yang berarti di G terdapat beberapa sirkuit. G dapat diubah menjadi pohon $T = (V_1, V_2)$dengan cara memutuskan sirkuit-sirkuit yang ada, yakni memilih sebuah sirkuit, kemudian menghapus sebuah sisi dari sirkuit tersebut.

Jika G adalah graph coloring berbobot, maka bobot pohon merentang T dari G didefinisikan sebagai jumlah bobot semua sisi di T. Pohon merentang yang berbeda mempunyai bobot yang berbeda pula. Diantara semua pohon merentang di G, pohon merentang yang berbobot minimum dinamakan Minimum spanning tree (MST). MST memiliki terapan yang cukup luas dalam prakteknya. Misalkan pemerintah akan membangun jalur rel kereta api yang menghubungkan sejumlah kota. Membangun jalur rel kereta api memerlukan banyak biaya. Karena itu pembangunan jalur itu tidak perlu menghubungkan langsung dua buah kota, tetapi cukup membangun jalur kereta seperti pohon merentang. Karena di dalam sebuah graph coloring mungkin saja terdapat lebih dari satu pohon

merentang, maka harus dicari pohon merentang dengan jumlah jarak terpendek, dengan kata lain harus dicari Minimum spanning tree-nya.

2.2.3 Representasi Graph Coloring dalam proses Segmentasi Citra

Teknik graph coloring yang didasarkan segmentasi citra pada umumnya mempresentasikan masalah graph coloring $G = (V,E)$ dimana setiap simpul $v1 \epsilon V.i$=1,2,3,....,n dianggap piksel-piksel dari citra dan sisi E merupakan pasangan-pasangan dari piksel-piksel yang bertetanggaan. Setiap sisi yang menghubungkan piksel-piksel pada citra memiliki bobot *(weight)* yaitu nilai intensitas keabuan pada citra itu sendiri. Edge yang menghubungkan piksel (simpul) p dan q pada graph coloring memiliki bobot *(weighted graph coloring)* yaitu ditunjukkan pada rumus berikut:

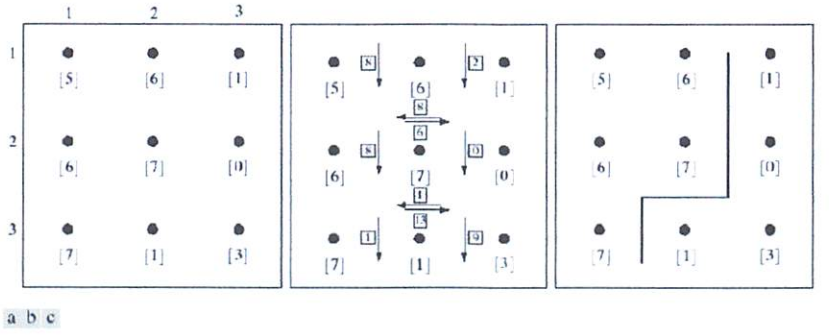
$$W((p,q))=H-[f(p)-f(q)]$$

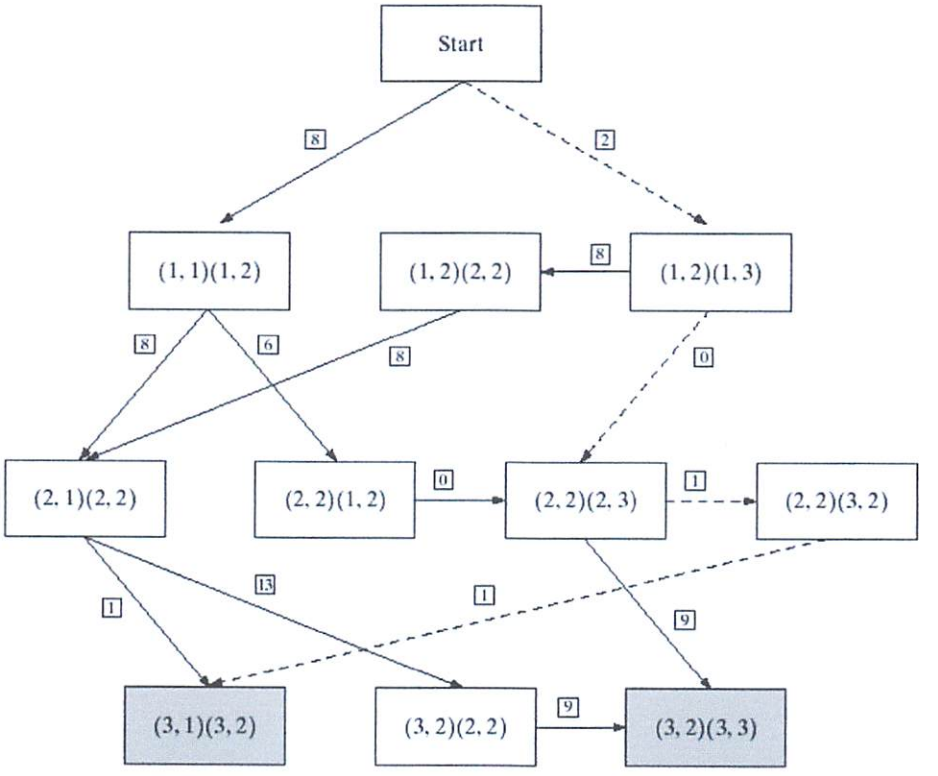
Dimana:

H = nilai intensitas keabuan tertinggi pada citra

f(p) dan f(q) = nilai keabuan pikel p dan q

pada contoh berikut akan dilakukan proses segmentasi dasar dengan menggunakan teori graph coloring pada citra berukutran 3x3. Untuk memudahkan diasumsikan bahwa sisi di mulai sari baris teratas dan berakhir di baris terakhir. Drngan kata lain, sisi hanya berawal dari (1,1),(1,2) atau (1,2),(1,3) dan berakhir di (3,1)(3,2) atau (3,2)(3,3)





Gambar 2.6 Representasi Graph Coloring

Dalam hal proses segmentasi citra, graph coloring G=(V,E) merupakan graph coloring tidak berarah (undirected graph), dengan simpul-simpul $v_i \in V$ merupakan himpunan elemen-elemen yang akan disegmentasi dan sisi $(v_i, v_j) \in E$ merupakan pasangan-pasangan dari piksel-piksel ketetanggaan. Setiap sisi $(v_i, v_j) \in E$ memiliki bobot $w((v_i, v_j))$ atau w(e), dimana bobot-bobotnya negatif dan tidak sama diantara simpul v_i dan v_j yang bertetanggaan.

Proses segmentasi dalam pendekatan pada graph coloring dilakukan melalui proses partisi graph coloring yaitu dilakukan pengelompokan dengan cara mempartisi himpunan dari simpul-simpulnya. Segmentasi S merupakan partisi dari V ke dalam komponen-komponen C sehingga setiap komponen $C \in S$ bersesuaian dengan komponen yang terhubung dalam sebuah graph $G^I = (V, E^I)$ dimana $E^I \geq E$. Atau dengan kata lain segmentasi yang dilakukan diindukasi oleh sebuah subset dari sisi E.

Ada beberapa cara dalam menentukan kualitas dari segmentasi, tetapi pada umumnya elemen yang diharapkan dalam sebuah komponen adalah sama dengan elemen pada komponen yang sama haris mempunyai bobot yang relatif rendah dan sisi diantara simpul-simpul di beberapa komponen harus mempunyai bobot yang tinggi. Tujuan mempartisi citra ke dalam himpunan yang berbeda daerah adalah untuk memperoleh struktur dari citra tersebut, untuk mempresentasikan citra sehingga menjadi lebih kompak, serta untuk mengetahui perbedaan struktur citra yang memiliki level tinggi maupun level rendah.

2.2.4. Daerah pasangan berurutan dari perbandingan predikatnya

Didefinisikan D sebagai predikat untuk mengevaluasi apakah ada atau tidaknya bukti untuk sebuah batasan antara dua komponen pada segmentasi (dua daerah dari citra). Predikat ini didasarkan pada ukuran ketidaksamaan antar elemen sepanjang batas dua relatif komponen kesebuah ukuran ketidaksamaan antara elemen ketetanggan dalam setiap dua komponen. Hasil predikat membandingkan perbedaan inter-komponen dengan perbedaan antar-komponen dan ini sesuai dengan data local karakteristik yang berlaku.

Didefinisikan *internal difference (Int)* pada komponen $C \geq V$ adalah bobot terbesar dalam komponen *minimum spaning tree (MST(C,E))* yaitu:

$$Int(C) = \max_{e \in MST(C,E)} W(e)$$

Salah satu intuisi yang mendasari ukuran ini adalah komponen C yang diberikan hanya terhubung ketika setidaknya bobot dari satu sisi *int (C)* diperhitungkan.

Didefinisikan *difference between (Dif)* antara dua komponen $C1, C2 \geq V$ adalah bobot minimum dari sisi yang menghubungkan dua komponen yaitu:

$$Dif(C_1, C_2) - \min_{v_i \in C_2, v_j \in C_2, (v_i, v_j) \in E} W\left((v_i, v_j)\right)$$

Jika tidak ada sisi yang terhubung antara C1 dan C2 maka dianggap

$$Dif(C1, C2) = \infty \quad .$$

Sebuah fungsi batas digunakan untuk mengontrol derajat *(degree)* dimana perbedaan antara komponen-komponen harus lebih besar dari pada minimum *internal difference*. Didefinisikan pasangan berurutan perbandingan predikat sebagai berikut:

$$D(C_1, C_2) = \begin{cases} benar & \text{jika } Dif(C_1, C_2) > M\ Int(C_1, C_2) \\ salah & \text{untuk yang lainnya} \end{cases}$$

Dimana minimum *internal difference (M Int)* didefinisikan sebagai berikut:

$$M\ Int(C_2, C_1) - \min\left(int(C_1) + \pi(C_1), int(C_2) + \pi(C_2)\right)$$

Fungsi ambang *(threshold)* π berguna untuk mengontrol derajat, dimana perbedaan antar dua komponen harus lebih besar dari internal difference-nya. Hal

ini menujukkan adanya bukti batasan antara dua komponen. Untuk komponen yang kecil, *Int(C)* bukanlah ukuran yang baik untuk karakteristik lokal data. Dalam kasus yang lebih besar, ketika $|C| = 1$, $int(C) = 0$. Oleh karena itu digunakan fungsi ambang berdasarkan ukuran komponennya, yaitu:

$$\pi(C) = \frac{k}{|C|}$$

Dimana $|C|$ dinotasikan sebagai ukuran C dan k sebagai parameter konstanta. Jelasnya untuk komponen yang kecil dibutuhkan bukti yang kuat sebagai batasan. Dalam prakteknya k menetapkan skala pengamatan, dalam k yang lebih besar menyebabkan k menjadi lebih besar. Bagaimanpun k bukan ukuran komponen minimum. Komponen yang lebih kecil diperbolehkan bila ada perbedaan yang cukup besar antar komponen bertetangga.

2.3 Pengenalan Pola

Pengenalan pola (pattern recognition) merupakan salah satu cabang ilmu computer yang dapat diartikan sebagai pengumpulan data-data mentah untuk dapat diklasifikasikan dengan maksud dan tujuan tertentu. Pengenalan pola ini bersifat conceptually driven processing yang berarti bahwa proses dimulai dari pembentukan konsep pada objek yang dijumpai (informasi dari memori).

Ada banyak aplikasi yang bisa dijadi implementasi dari pengenalan pola, diantaranya adalah pengenalan wajah pada manusia, pengenalan tulisan tangan, pengenalan penyakit berdasarkan gejala-gejala yang ditemukan pada sebuah objek. Pengenalan wajah dan tulisan tangan manusia bisa dibuat melalui pendekatan pemrosesan citra yang mana tujuan akhir dari pemrosesan citra tersebut digunakan untuk pengelompokan objek sehingga menghasilkan output

yang diinginkan. Sedangkan pengenalan penyakit berdasarkan gejala-gejala yang ada pada seorang pasien bisa dilakukan dengan pendekatan analisa data, kemudian mengambil kesimpulan dari data-data yang sudah dikelompokkan. Atau dengan kata lain, kita membuat sebuah aplikasi yang secara otomatis akan bercerita dan menjelaskan secara terperinci tentang informasi yang dikandung sebuah objek. Dan bagaimana kelengkapan informasi yang dikandung dari objek tersebut, bergantung kepada kualitas dan kuantitas dari data statistik yang kita punya.

Beberapa metode yang bisa digunakan untuk pengenalan pola adalah JST (jaringan syaraf tiruan), metode statistik, metode terstruktur dan lain sebagainya. Dengan JST, menganalogikan cara berfikir pada otak manusia. Jadi, informasi di proses sebagaimana otak manusia memproses sebuah informasi yang di dapat. Misalnya cara pengenalan wajah pada manusia. Sedangkan metode statistik berdasarkan hasil analisa data yang sudah terkumpul. Misalnya pengenalan dan analisa penyakit pada manusia.

Salah satu contoh pengenalan pola adalah memprediksi makanan yang sesuai untuk seseorang dilihat dari sudut pandang tempat tinggal, suhu maupun kepekaan badan orang yang bersangkutan terhadap penyakit. Pengenalan pola di atas dapat kita selesaikan dengan menggunakan metode statistik. Pada tahap awal, dilakukan pengumpulan data tentang makanan serta zat yang dikandung oleh makan tersebut. Kemudian, pengumpulan data tentang zat-zat yang diperlukan tubuh untuk kondisi daerah dan suhu tertentu. Kemudian, dilakukan pemrosesan dan anlisa sehingga menghasilkan informasi yang kita inginkan.

2.3.1 Deteksi Wajah

Deteksi wajah dapat dipandang sebagai masalah klasifikasi pola dimana inputnya adalah citra masukan dan akan ditentukan output yang berupa label kelas dari citra tersebut. Dalam hal ini terdapat dua label kelas,yaitu wajah dan nonwajah.

Teknik-teknik pengenalan wajah yang dilakukan selama ini banyak yang menggunakan asumsi bahwa data wajah yang tersedia memiliki ukuran yang sama dan latar belakang yang seragam. Di dunia nyata, asumsi ini tidak selalu berlaku karena wajah dapat muncul dengan berbagai ukuran dan posisi di dalam citra dan dengan latar belakang yang bervariasi.

Pendeteksian wajah (face detection) adalah salah satu tahap awal yang sangat penting sebelum dilakukan proses pengenalan wajah (face recognition). Bidang-bidang penelitian yang berkaitan dengan pemrosesan wajah (face processing) adalah pengenalan wajah (face recognition) yaitu membandingkan citra wajah masukan dengan suatu database wajah dan menemukan wajah yang paling cocok dengan citra masukan tersebut.

Autentikasi wajah (face authentication) yaitu menguji keaslian/kesamaan suatu wajah dengan data wajah yang telah diinputkan sebelumnya. Lokalisasi wajah (face localization) yaitu pendeteksian wajah namun dengan asumsi hanya ada satu wajah di dalam citra. Penjejakan wajah (face tracking) yaitu memperkirakan lokasi suatu wajah di dalam video secara real time.

Pengenalan ekspresi wajah (facial expression recognition) untuk mengenali kondisi emosi manusia. Tantangan yang dihadapi pada masalah deteksi wajah disebabkan oleh adanya faktor-faktor berikut :

- Posisi wajah. Posisi wajah di dalam citra dapat bervariasi karena posisinya bisa tegak, miring, menoleh, atau dilihat dari samping. Komponen-komponen pada wajah yang bisa ada atau tidak ada, misalnya kumis, jenggot, dan kacamata.
- Ekspresi wajah. Penampilan wajah sangat dipengaruhi oleh ekspresi wajah seseorang, misalnya tersenyum, tertawa, sedih, berbicara, dan sebagainya.
- Terhalang objek lain. Citra wajah dapat terhalangi sebagian oleh objek atau wajah lain, misalnya pada citra berisi sekelompok orang. Kondisi pengambilan citra. Citra yang diperoleh sangat dipengaruhi oleh faktor-faktor seperti intensitas cahaya ruangan, arah sumber cahaya, dan karakteristik sensor dan lensa kamera.

Pengelompokkan metode deteksi wajah menjadi empat kategori, yaitu:

1. Knowledge-based method. Metode ini kebanyakan digunakan untuk lokalisasi wajah.
2. Feature invariant approach. Metode ini kebanyakan digunakan untuk lokalisasi wajah.
3. Template matching method. Metode ini digunakan untuk lokalisasi wajah maupun deteksi wajah.
4. Appearance-based method. Metode ini kebanyakan digunakan untuk deteksi wajah.

2.3.2. Template Matching

Pada metode ini akan disimpan beberapa pola wajah standar untuk mendeskripsikan wajah secara keseluruhan maupun bagian-bagiannya. Pada saat pendeteksian akan dihitung korelasi antara citra input dengan citra pola wajah yang tersimpan sebelumnya. Pada pendekatan ini, para peneliti mencoba menemukan fitur-fitur yang tidak berubah (invariant) pada wajah. Asumsi ini didasarkan pada observasi bahwa manusia dapat dengan mudah mendeteksi wajah dengan berbagai pose dan kondisi cahaya, sehingga tentunya ada sifat-sifat atau fitur-fitur yang bersifat invariant. Fitur wajah seperti alis, mata, hidung, mulut, biasanya diekstraksi dengan edge detector. Selanjutnya dibentuk suatu model statistik yang mendeskripsikan hubungan antara fitur-fitur tersebut untuk menentukan ada tidaknya wajah.

Template matching merupakan proses pencocokan sebuah obyek dari citra dengan obyek citra lain yang sudah ditentukan.

Aplikasi alamiah dari template matching digunakan untuk membangun gambar template yang diminta untuk membagi kategori semantik. Template ini dapat dibangun on-line, dan digunakan untuk menyederhanakan query dengan menggunakan suatu template yang ada dan bukan mennyusun suatu query.

Menemukan wajah merupakan bagian template matching dengan tampilan fontal dari wajah, kemudian mencari bagian yang gelap yang khas dari mulut, hidung dan mata. Jika wajah dapat ditemukan, maka tidak terikat pada identitas dari seseorang tertentu yakni hanya mencari polanya.

Pemberian gambar dengan piksel W x H dan template dengan piksel w x h, dihasilkan gambar dengan piksel W-w+1xH-h+1, dan nilai piksel pada tiap

lokasi (x,y) yang memiliki karakteristik kesamaan antara template dan gambar dengan sudut atas kiri pada (x,y) dan sudut kanan bawah pada (x+w-1,y+h-1).

Persamaannya dapat dirumuskan sebagai berikut :

$$R(x,y) = \sum_{y'=0}^{h-1} \sum_{x'=0}^{w-1} T'(x',y') I'(x+x', y+y')$$

dimana T=template dan I=image sedangkan T' dan I' merupakan nilai masing-masing piksel dikurangi dengan rata-rata pada suatu blok.

$$T'(x',y') = T(x',y') - \overline{T}$$

$$I'(x+x', y+y') = I(x+X', y+y') - \overline{I}(x,y)$$

Dalam template *matching*, pola wajah bakuan (biasanya tampak depan/*frontal*) ditetapkan terlebih dahulu secara manual oleh suatu fungsi. Kemudian diberi suatu citra masukan, nilai korelasi dengan pola bakuan dihitung untuk kontur wajah, mata, hidung, dan mulut secara bebas[1]. Keberadaan sebuah wajah ditentukan berdasarkan nilai-nilai korelasi.

Untuk memperoleh nilai korelasasinya, maka digunakan rumus seperti dibawah ini :

$$Corr(x,y) = \frac{R(x,y)}{\sqrt{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1} T'(x',y')^2 \sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1} I'(x+x', y+y')^2}}$$

2.4 Matlab 7.0.4

Matlab 7.0.4 merupakan software program aplikasi yang digunakan untuk komputasi teknik. Matlab singkatan dari Matrix Laboratory yang merupakan salah satu bahasa pemograman yang dikembangkan oleh MathWorks. Matlab mampu mengintegrasikan komputasi, visualisasi, dan pemrograman untuk dapat digunakan secara mudah[11]. Penggunaan Matlab diantaranya adalah pada:

- Matematika dan Komputansi
- Pengembangan algoritma
- Pemodelan, simulasi, dan prototipe
- Analisa, eksplorasi, dan visualisasi data
- Pengolahan grafik untuk sains dan teknik
- Pengembangan Aplikasi berbasis GUI (Graphical User Interface)

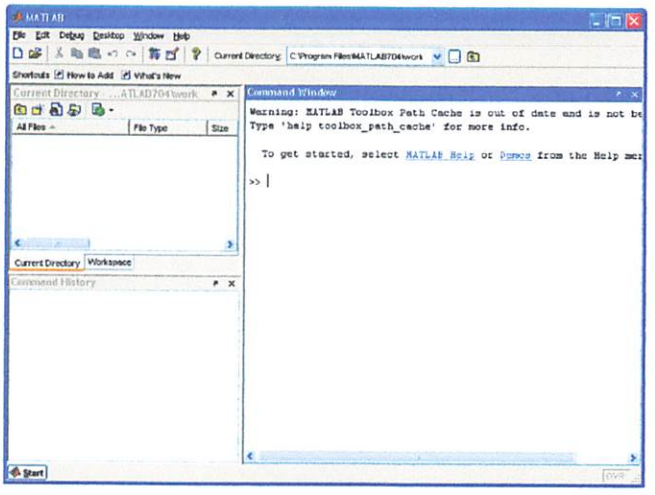
Matlab muncul di dunia bahasa pemrograman yang cenderung dikuasai oleh bahasa yang telah mapan. Logikanya, sebagai pemain baru tentu saja Matlab akan sukar mendapat hati dari pemakai. Namun Matlab hadir tidak dengan fungsi dan karakteristik yang ditawarkan bahasa pemrograman lain yang biasanya hampir seragam. Matlab dikembangkan sebagai bahasa pemrograman sekaligus alat visualisasi, yang menawarkan banyak kemampuan untuk menyelesaikan berbagai kasus yang berhubungan langsung dengan disiplin keilmuan Matematika, seperti bidang rekayasa teknik, fisika, statistika, komputasi dan modeling. Matlab dibangun dari bahasa induknya yaitu bahasa C, namun tidak dapat dikatakan sebagai varian dari C, karema dalam sintak maupun cara kerjanya sama sekali berbeda dengan C. Namun dengan hubungan langsungnya terhadap bahasa C,

Matlab mempunyai kelebihan-kelebihan bahasa C bahkan mampu berjalan pada semua *platform* Sistem Operasi tanpa mengubah sintak sama sekali.

Matlab adalah bahasa pemrograman level tinggi yang dikhususkan untuk komputasi teknis[10]. Bahasa ini mengintegrasikan kemampuan komputasi, visualisasi dan pemrograman dalam sebuah lingkungan yang tunggal dan mudah digunakan. Matlab memberikan sistem interaktif yang menggunakan konsep *array*/matrik sebagai standar variabel elemennya tanpa membutuhkan pen-deklarasi-an array seperti pada bahasa lainnya.

Matlab dikembangkan oleh MathWorks, yang pada awalnya dibuat untuk memberikan kemudahan mengakses data matrik pada program LINPACK dan EISPACK. Selanjutnya menjadi sebuah aplikasi untuk komputasi matrik. Dari sejak awal dipergunakan, Matlab memperoleh masukan ribuan pemakai. Kehadiran Matlab memberikan jawaban sekaligus tantangan. Matlab menyediakan beberapa pilihan untuk dipelajari, mempelajari metoda visualisasi saja, pemrograman saja atau kedua-duanya. Kemudahan yang lain, karena bahasa pemrograman yang lain memang tidak menawarkan kemudahan serupa. Selain itu Matlab juga memberikan keuntungan bagi programmer-developer program yaitu untuk menjadi program pembanding yang sangat handal, hal tersebut dapat dilakukan karena kekayaannya akan fungsi matematika, fisika, statistik dan visualisasi.

Pada skripsi ini Matlab 7.0.4 digunakan untuk proses pengolahan data, yakni proses yang berkaitan dengan analisa, visualisasi data, dan pengembangan aplikasi berbasis GUI.

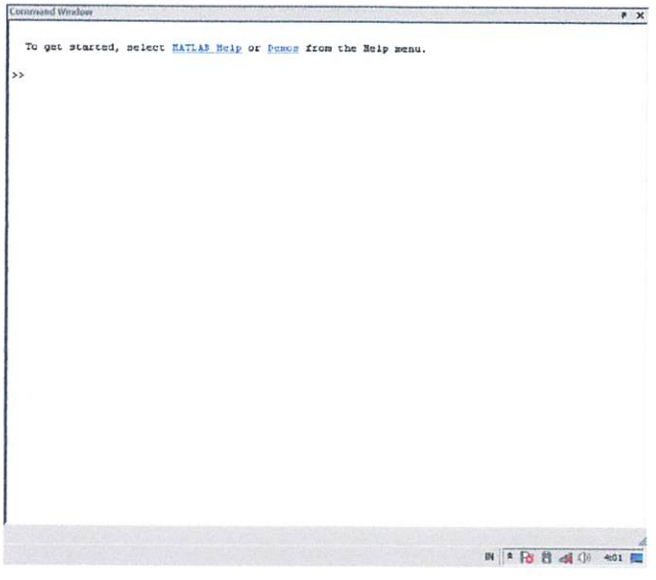


Gambar 2.7. Tampilan Utama Matlab

2.4.1. Lingkup Matlab

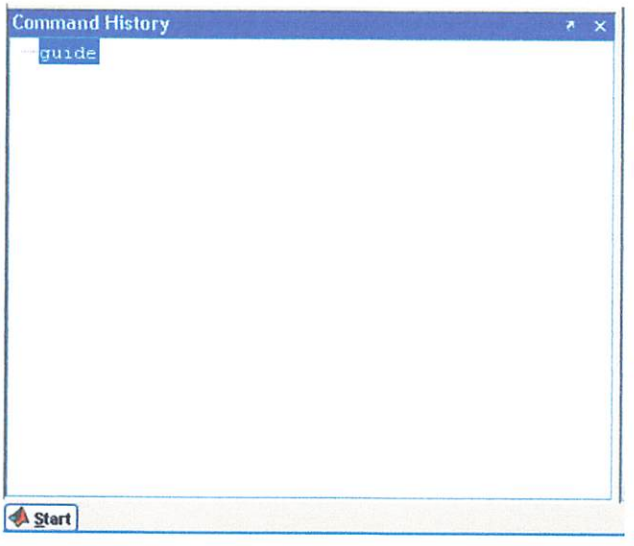
Sebagaimana bahasa pemrograman lainnya, Matlab juga menyediakan lingkungan kerja terpadu yang sangat mendukung dalam pembangunan aplikasi. Tampilan jendela Matlab dapat dibagi menjadi beberapa bagian, yaitu :

- Command Window, yang berfungsi untuk tempat memasukkan dan menjalankan variabel (fungsi) dari Matlab dan M File.



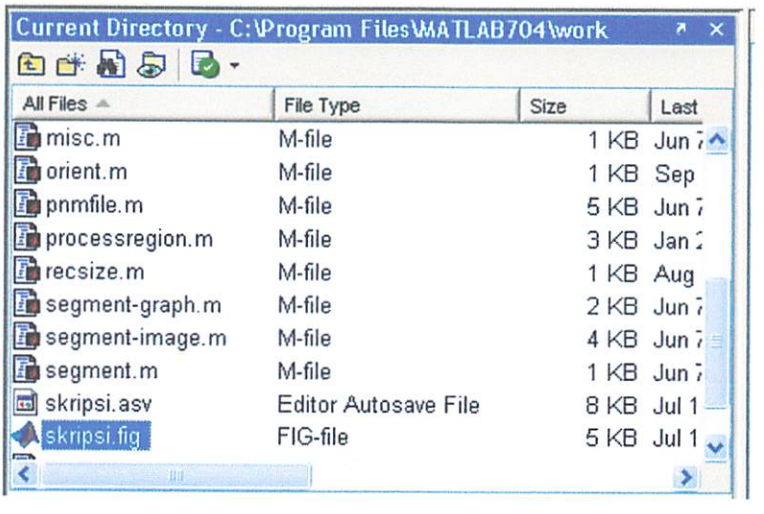
Gambar 2.8. Tampilan *Workspace*

- Command History, yang berfungsi menampilkan fungsi- fungsi yang telah dikerjakan pada command window.



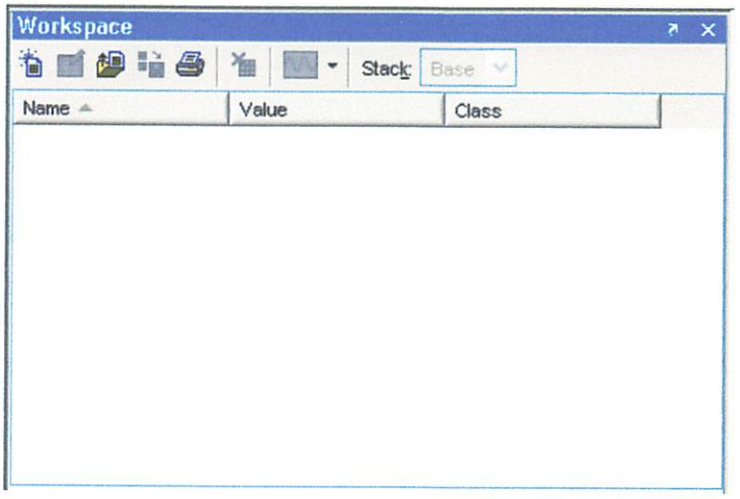
Gambar 2.9 Tampilan *Command History*

- Launch Pad, yang berfungsi untuk akses tools, demo, dan dokumentasi semua produk Math Works.
- Current Directory Browser, yang berfungsi menampilkan file-file Matlab dan file yang terkait serta mengerjakan operasi file seperti membuka dan mencari isi file.



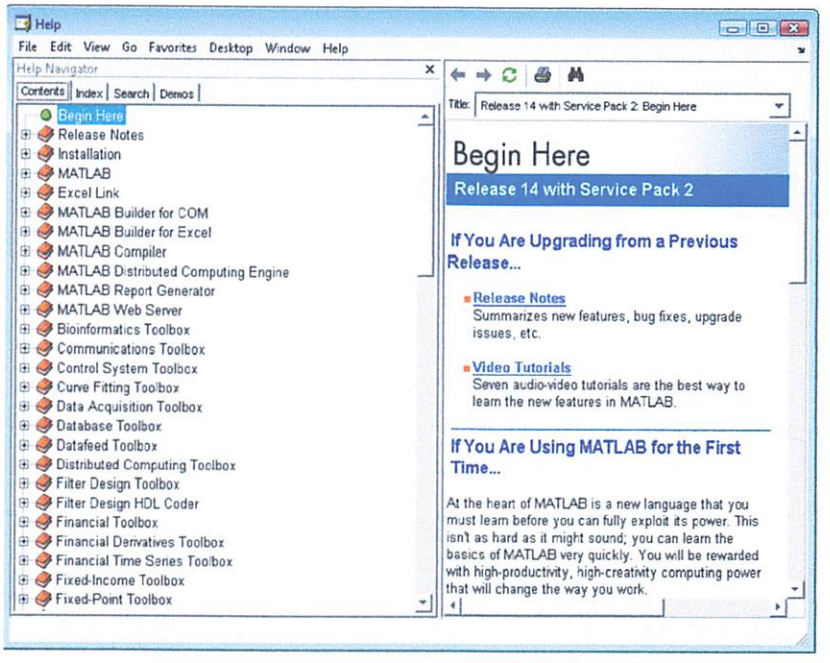
Gambar 2.10. Tampilan *Current Directory*

- Workspace Browser, yang memuat variabel-variabel yang dibuat dan yang disimpan dalam memori saat penggunaan Matlab.



Gambar 2.11. Tampilan *Workspace Browser*

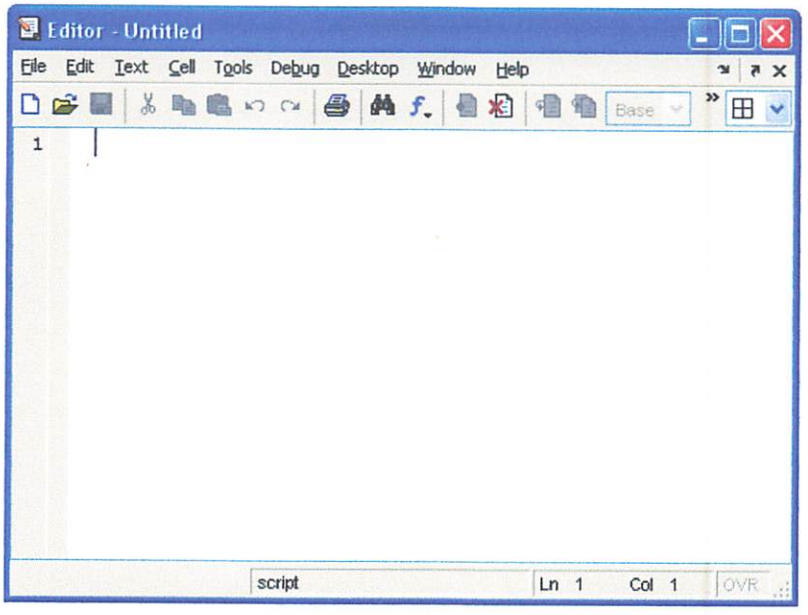
- Editor / Debugger, yang berfungsi untuk membuat dan memeriksa M File
- Help Browser, yang berfungsi untuk menampilkan dan mencari dokumentasi yang ada pada Matlab.



Gambar 2.12. Tampilan *Help*

2.4.2. M File Editor

M File merupakan file teks yang memuat variabel- variabel dan fungsi yang ada pada Matlab. M File berupa nama file script dalam Matlab yang disimpan dengan ekstensi '.m'. M File memudahkan dalam penulisan (pembuatan) program dalam Matlab. Dimana fungsi-fungsi yang ada pada M File tersebut dapat mengakses semua variabel Matlab dan menjadi bagian dari ruang kerja Matlab.



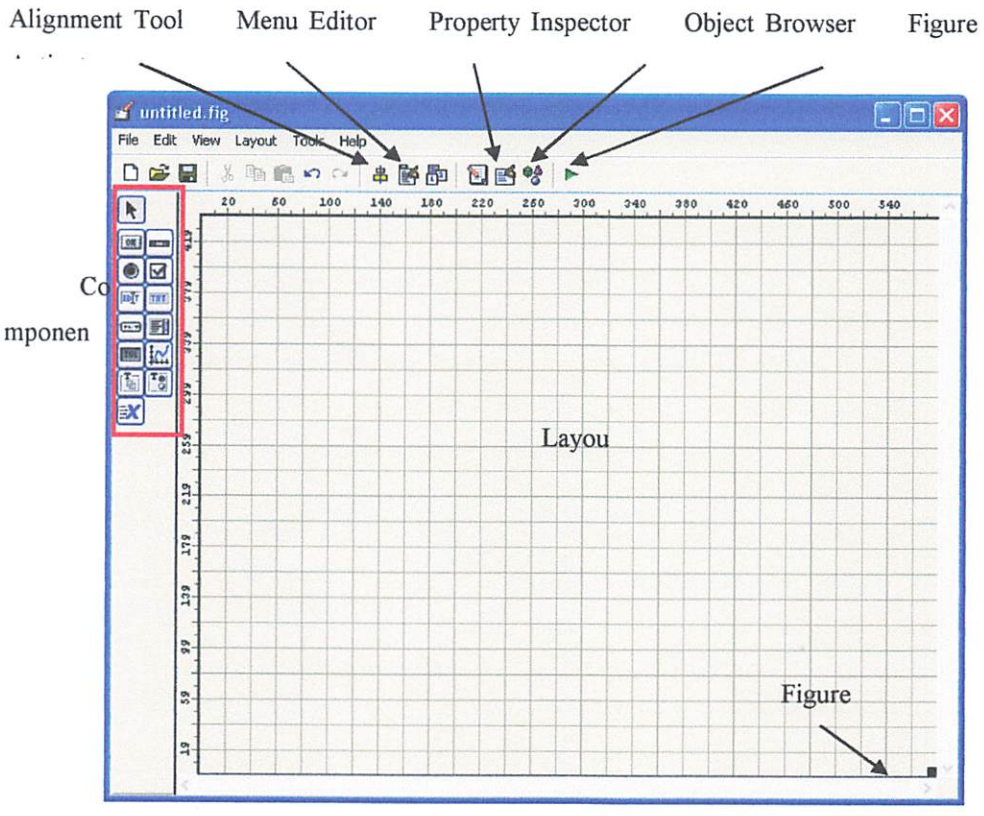
Gambar 2.13. Tampilan Layout M File

2.4.3. Matlab GUI (Graphical User Inerface)

GUI (Graphical User Interface) merupakan software aplikasi dari Matlab yang mampu menampilkan secara visualisasi program yang telah dibuat pada Matlab (M-File), dengan melalui bantuan komponen- komponen yang ada seperti icons, pushbutton, radio button, dan sebagainya.

GUIDE (GUI Development Environtment) merupakan tools Matlab yang diaplikasikan untuk pembuatan Gui. Guide menyediakan seperangkat tools yang digunakan untuk mendesain dan menampilkan GUI. Salah satunya adalah tools

Layout Editor, yang berfungsi sebagai tempat peletakan komponen-komponen yang dibutuhkan. Dimana ukuran, jarak antar komponen dan align dari komponen tersebut dapat diatur. Tools ini secara otomatis tampil, pada saat pertama kali menjalankan software aplikasi GUI pada Matlab 7.0.4.



Gambar 2.14. Layout Editor Dari Guide

Programer dapat memanfaatkan tools dan komponen Guide lainnya yang merupakan bagian dari user interface control (uicontrols) dan user interface menus (uimenus) untuk memudahkan dalam pembuatan GUI. Beberapa tools dasar dari Guide, antara lain:

- Layout Area, digunakan untuk menambah atau mengatur object pada figure window.
- Aligment Tool, digunakan untuk mengatur jarak antar object

- Property Inspector, digunakan untuk mengatur properti dari object
- Object Browser, digunakan untuk menampilkan secara hirarki object yang sedang digunakan dalam layout.
- Menu Editor, digunakan untuk menambahkan menu dan contect yang ada didalamnya pada layout.
- Figure Activator atau yang biasa disebut Run, digunakan untuk menjalankan program.

Sementara dalam component pallete terdapat beberapa object yang biasa digunakan untuk tampilan pada figure yang akan dibuat. Object – object tersebut adalah :

- Push Button  , berfungsi untuk menjalankan eksekusi seketika jika ditekan.
- Slider  , digunakan untuk menampilkan range suatu nilai dan kita dapat memilih nilai yang diinginkan dengan melakukan drag.
- Radio Button  , berfungsi untuk memilih satu pilihan dari beberapa pilihan.
- Check Box  , sama seperti radio button namun dapat berfungsi untuk memilih lebih dari satu pilihan dari beberapa pilihan.
- Editable Text  berfungsi untuk menampilkan teks dan teks ini dapat sewaktu-waktu diedit.
- Static Text  , berfungsi untuk menampilkan teks secara statis.
- Pop Up Menu  , untuk memilih satu list dari beberapa list yang ada (ditampilkan).

- List Box  , berfungsi menampilkan keseluruhan list.
- Toggle Button  , Button berfungsi untuk menjalankan eksekusi secara on ,off.
- Axes  , berfungsi untuk menampilkan gambar atau grafik.
- Panel  , digunakan untuk menandai atau mengelompokkan daerah tertentu pada figure.
- Button Grup  , digunakan untuk mengelompokkan button grup dan toggle button.
- Frame, berguna untuk menampilkan dan mengelompokkan beberapa kontrol fungsi yang masih berkaitan.
- Figure merupakan tempat untuk meletakkan komponen Gui yang telah didesain dengan Layout Editor.

Semua object diatas dikendalikan lewat command dalam fungsi callback untuk setiap browser yang berada pada file tipe ".m" dari figure yang dibuat. Sementara data yang digunakan dalam mendesign figure disimpan dalam struktur handles.

BAB III

PERANCANGAN SISTEM

3.1. Perancangan Sistem

Secara garis besar aktivitas dari desain sistem pada skripsi ini adalah mensegmentasi warna pada citra digital menggunakan metode graph coloring untuk pengenalan pola. Dimana untuk pengenalan polanya lebih difokuskan untuk deteksi wajah menggunakan metode template matching.

Segmentasi citra *(image segmentation)* merupakan salah satu proses penting yang banyak aplikasinya dalam pengolahan citra, namun tekniknya cukup rumit. Segmentasi citra diperlukan karena proses ini merupakan proses identifikasi objek yang dinyatakan dalam bentuk dasar, dimana objek dalam suatu citra akan lebih mudah dikenal apabila strukturnya jelas. Melalui proses segmentasi citra diharapkan objek yang terkandung pada gambar dapat dideskripsikan dan dikenali. Proses-proses yang termasuk dalam segmentasi citra, yaitu:

1. Preprocessing, merupakan kumpulan dari proses yang digunakan untuk dapat menghasilkan segmentasi yang baik.
   a. Konversi yaitu proses mengubah citra menjadi citra lain. Dalam hal ini konversi yang dilakukan adalah konversi citra RGB menjadi citra grayscale.
   b. Konvolusi yaitu proses menghilangkan komponen noise yang variasinya mempunyai pola acak dan bersifat jangka pendek serta tidak mempunyai korelasi dengan noise yang terjadi di titik-titik

sekitarnya dan mempunyai karakter pemerataan atau pengaburan.

c. Filterisasi yaitu proses meningkatkan mutu citra yang mengalami gangguan pada saat citra awal diproses oleh perekam citra digital. Dalam hal ini digunakan jenis filter tipe gaussian, karena filter ini sangat baik untuk menghilangkan noise yang bersifat sebaran normal, yang banyak dijumpai pada citra hasil proses digitalisasi menggunakan kamera.

2. Segmentasi citra dengan metode graph coloring yaitu melakukan proses pengelompokan komponen-komponen yang terdapat didalam citra.

Pengenalan Pola difokuskan untuk deteksi wajah menggunakan metode tamplate mathching yaitu mendeteksi wajah manusia dengan template wajah yang sudah ditentukan jika ada wajah di dalam citra yang di inputkan, program akan mandeteksi bahwa didalam citra digital tersebut terdapat wajah manusia dengan sebuah template wajah diatasnya yang akan menunjukkan letak dari masing-masing wajah yang berhasil di deteksi.

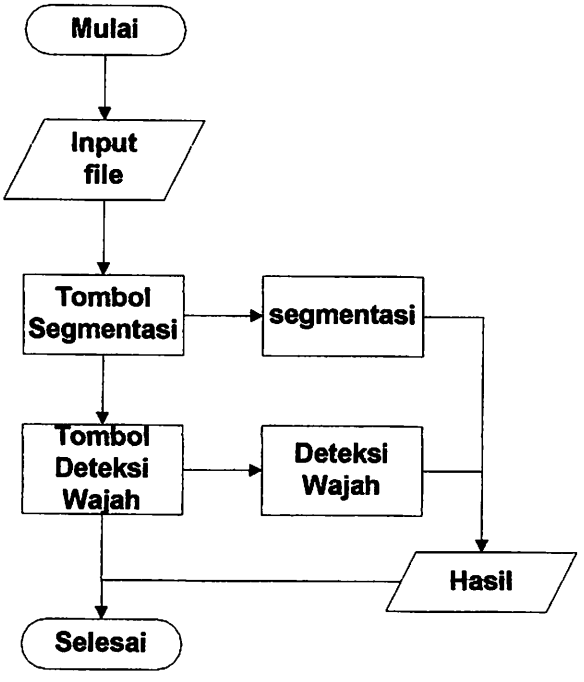
3.2 Flowchart Segmentasi Warna pada Citra Digital Menggunakan Metode Graph Coloring untuk Pengenalan Pola

Suatu program yang baik haruslah bebas dari kesalahan dan kemungkinan kesalahan serta memiliki urutan sistematis dan terarah agar prosesnya dapat terlaksana secara efisien. Untuk itu sebelum menyusun suatu program, perlu disusun

urutan-urutan proses yang logis dan sistematis. Maka akan ditentukan dahulu bentuk flowchart dalam mendukung proses segmentasi citra, diantaranya:

3.2.1 Flowchart Secara Garis Besar

Secara garis besar flowchart segmentasi warna pada citra digital menggunakan metode graph coloring untuk pengenalan pola ditunjukkan sebagai berikut.



Gambar 3.1 Flowchart secara garis besar

Keterangan:

(i) Input file yang nantinya akan disegmentasi, setelah nama file di-input, gambar di-load dan dipersiapkan untuk segmentasi dan deteksi wajah.

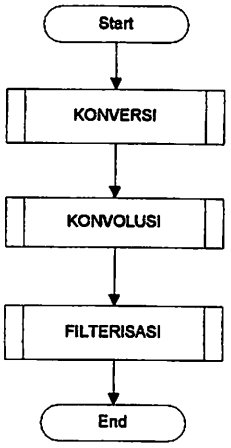
(ii) Tombol Segmentasi di sini terdiri dari beberapa proses dan akan dibahas lebih lanjut setelah ini.

(iii) Tombol Deteksi Wajah di sini terdiri dari beberapa proses dan akan dibahas lebih lanjut setelah ini.

(iv) Proses berhenti di sini

3.2.2 Flowchart Preprocesing

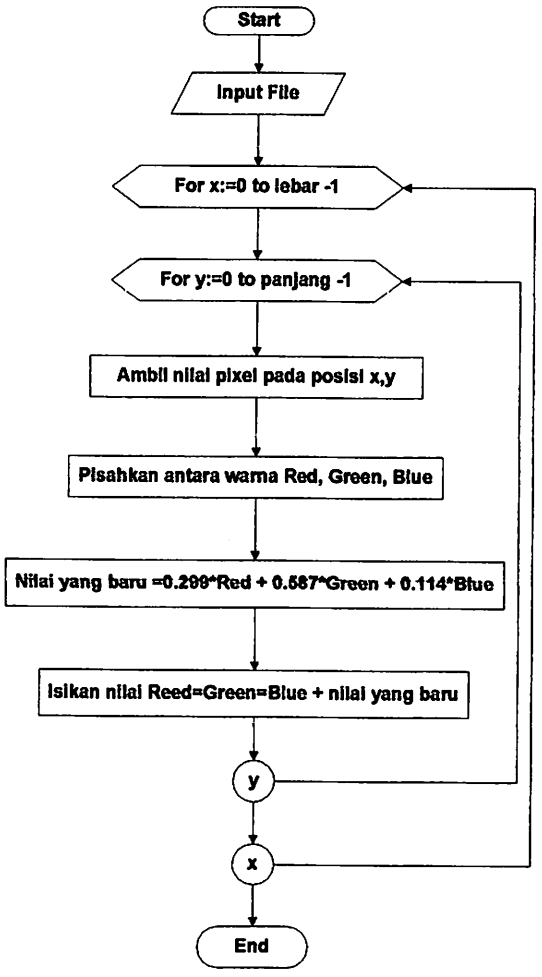
Preprocessing merupakan kumpulan dari proses yang digunakan untuk dapat menghasilkan segmentasi yang terbaik. Preprocessing yang digunakan adalah konversi, konvolusi, dan filterisasi. Flowchart ditunjukkan sebagai berikut.



Gambar 3.2 Flowchart Preprocesing

Dari proses preprocessing yang terlihat pada flowchart sebelumnya, dikatakan bahwa salah satu proses awal dari proses preprocessing ini adalah proses konversi. Dalam hal ini, proses konversi yang dilakukan adalah proses konversi

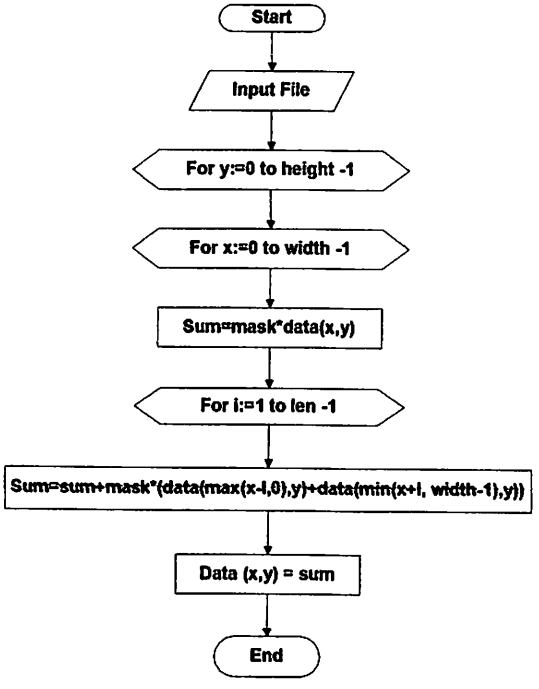
citra RGB menjadi citra grayscale (gambar yang memiliki tingkat warna abu-abu). Pada gambar 3.3 dijelaskan bagaimana cara kerja proses ini dalam bentuk flowchart.



Gambar 3.3 Flowchart Konversi

Nilai tiap titik citra yang akan di konversi akan disamakan nilai red, green, dan blue-nya sehingga untuk tiap titik hanya memiliki satu nilai saja yang disebut nilai gray level. Proses ini mengambil persentasi tertentu dari masing-masing warna kemudian dijumlahkan untuk mendapatkan nilai yang baru.

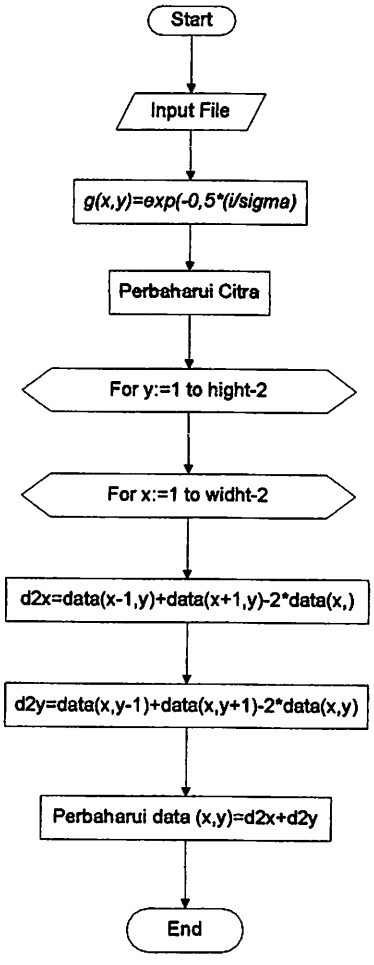
Proses kedua setelah proses konversi dari proses preprocessing adalah proses konvolusi. Proses konvolusi sangat berguna pada proses pengolahan citra seperti perbaikan kualitas citra (image enhancement), penghilangan derau, penghalusan/ pelembutan citra, deteksi tepi, dan penajaman tepi. Dalam hal ini, proses konvolusi digunakan untuk menghilangkan derau dan menajamkan tepi. Pada Gambar 3.6 dijelaskan bagaimana cara kerja proses ini dalam bentuk flowchart secara sederhana.



Gambar 3.4 Flowchart konvolusi

Proses terakhir dari proses preprocessing adalah proses filterisasi, dimana proses yang dilakukan disini adalah untuk menghilangkan noise dan

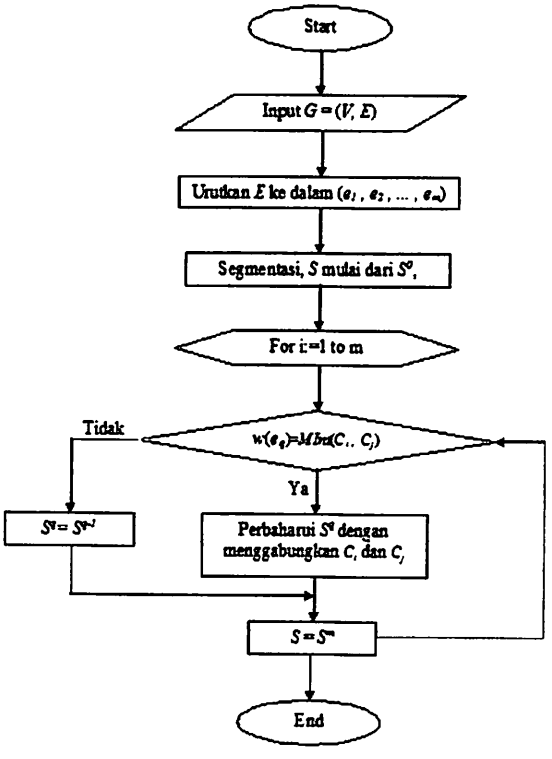
menghaluskan citra kemudian mendeteksi tepi dari citra. Proses ditunjukkan pada flowchart berikut.



Gambar 3.5 Flowchart Filterisasi

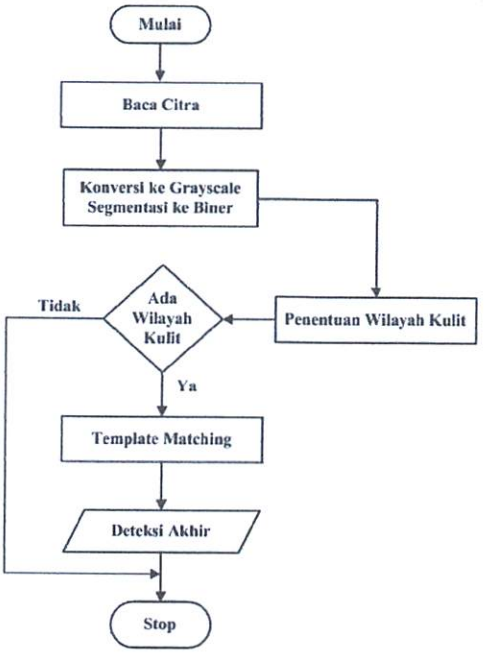
3.2.3 Flowchart Segmentasi

Berikut flowchart dari segmentsi:



Gambar 3.6 Flowchart Segmentasi

Dalam proses pengenalan pola yang difokuskan pada deteksi wajah ini, citra inputan yang dipakai adalah citra berwarna dengan format .jpg. Pembacaan citra inputan diperlukan untuk mendapatkan data masukan yang selanjutnya akan diproses untuk mendeteksi wajah sehingga didapatkan keluaran berupa citra wajah yang berhasil terdeteksi berdasarkan template wajah yang ada diatasnya. Berikut diagram alir dalam proses pengenalan pola yang difokuskan untuk deteksi wajah :



Gambar 3.7 Flowchart Proses Pengenalan Pola

3.3 Algoritma Segmentasi Citra menggunakan Metode Graph Coloring

Untuk sampai pada perancangan program, akan ditentukan dahulu bentuk algoritma dalam mendukung proses segmentasi citra menggunakan metode graph yang efisien, yaitu: Inputnya adalah sebuah graph $G = V, E$ dengan n simpul dan m sisi. Outputnya adalah sebuah segmentasi dari V ke komponen-komponen $S = (C_1, \ldots, C_n)$

1. Urutkan sisi E ke dalam $(e_1, e_2, \ldots, e_m)$ dengan bobot sisi tidak menurun.

2. Mulai dengan segmentasi S^0, dimana setiap simpul v_1 ada didalam komponen itu sendiri.

3. Ulangi langkah 4 untuk setiap e_q dari $(e_1, e_2, \ldots, e_m)$

4. jika bobot e_q relatif lebih kecil dari internal difference komponen yang terhubung, maka gabungkan komponen, selain itu tidak tidak berlaku apapun. Atau lebih formalnya, jika $w(e_q) \leq MInt(C_i, C_j)$ dimana $(C_i, C_j) \in S$ adalah komponen yang jelas terhubung dengan e_q, maka perbaharui S dengan menggabungkan C_i dan C_j.

5. Ulangi $S = S^m$.

3.4. Pengenalan Pola

Dalam proses pengenalan pola yang dfokuskan pada deteksi wajah ini citra inputan yang dipakai adalah citra berwarna dengan format .jpg. Pembacaan citra inputan diperlukan untuk mendapatkan data masukan yang selanjutnya akan diproses hingga didapatkan keluaran berupa citra wajah yang berhasil terdeteksi berdasarkan template wajah yang ada diatasnya. Berikut diagram alir dalam proses deteksi wajah

3.4.1 Konversi RGB ke Grayscale

Citra kemungkinan kulit disini citra inputan akan diubah kedalam citra aras keabuan (grayscale), dengan nilai keabuan akan menampilkan kemungkinan suatu piksel yang merupakan bagian dari wilayah kulit.

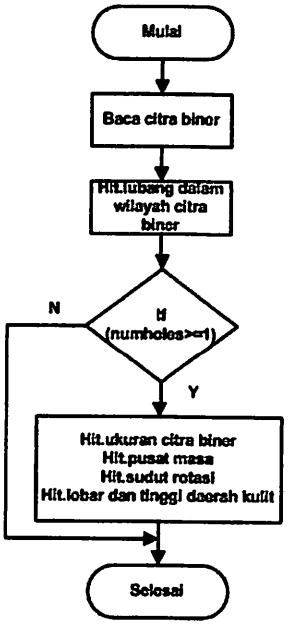
3.4.2 Penentuan Wilayah Kulit

Dalam proses deteksi wajah tahap awal yang dilakukan dalam penentuan wilayah kulit adalah dengan melabelkan wilayah-wilayah di dalam citra biner (citra segmentasi kulit). Selanjutnya akan dilakukan proses iterasi terhadap tiap-tiap wilayah yang ditemukan untuk menentukan apakah suatu wilayah kemungkinan suatu wajah atau bukan. Untuk dapat diproses lebih lanjut, suatu wilayah kulit setidaknya harus mempunyai sebuah lubang atau lebih di dalamnya. Jika jumlah lubang lebih besar atau sama dengan satu, maka dilakukan proses pencarian pusat massa, penentuan arah, perhitungan lebar dan tinggi wilayah kulit, perhitungan tinggi dan lebar wilayah kulit baru, serta pembacaan citra wajah model (template wajah).

Untuk proses penentuan wilayah kulit dalam proses deteksi wajah langkah-langkahnya adalah sebagai berikut :

1. Mulai
2. Baca citra biner dari hasil proses segmentasi citra gambar RGB ke biner
3. Tentukan jumlah lubang dalam daerah kulit yang dimiliki dalam citra biner
4. Jika daerah kulit yang terdeteksi memiliki lubang lebih atau sama dengan satu dalam wilayahnya maka daerah itu terdeteksi sebagai wilayah kulit.
5. Daerah yang terdeteksi sebagai wilayah kulit selanjutnya akan dilakukan proses perhitungan seperti menghitung ukuran citra biner, menghitung pusat masa dan sudut rotasi serta lebar dan tinggi daerah kulit. Proses ini dilakukan untuk menentukan posisi template dalam proses template matching nantinya.
6. Selesai
7.

Berikut adalah flowchart proses penentuan wilayah kulit dalam proses deteksi wajah :



Gambar 3.8 Flowchart Proses Penentuan Wilayah Kulit Dalam Proses Deteksi Wajah

3.4.3 Template Matching

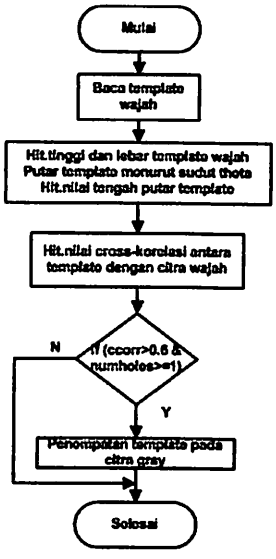
Dalam proses template matching akan didapatkan nilai korelasi-silang (cross-correlation value) antara bagian dari citra yang menyatakan wilayah wajah dengan citra wajah model yang telah diproses. Dalam proses template matching ini wajah berhasil terdeteksi jika dalam proses pencarian didapat nilai korelasi > 0.6 maka dalam citra yang diproses terdapat wajah yang barhasil terdeteksi.

Untuk proses template matching dalam proses deteksi wajah langkah-langkahnya adalah sebagai berikut :

1. Mulai

2. Baca citra template wajah yang akan digunakan sebagai template
3. Hitung tinggi dan lebar daerah template, kemudian putar template menurut sudut theta agar template yang dihasilkan sejajar dalam arah yang sama dengan daerah kulit, setelah itu menghitung nilai tengah putar template .
4. Hitung nilai cross-korelasi antara template dengan citra untuk menentukan posisi menempatkan citra template dengan citra gray
5. Jika nilai korelasi > 0.6 dan jumlah lubang lebih atau sama dengan satu maka template wajah sesuai dengan posisi wajah citra pada citra gray
6. Selanjutnya dilakukan proses penenpatan template wajah pada citra gray yang terdeteksi sebagai wilayah kulit wajah
7. Selesai

Berikut adalah flowchart proses template matching dalam proses deteksi wajah :



Gambar 3.9 Flowchart Proses Template Matching Dalam Proses Deteksi Wajah

Template wajah yang digunakan sebagai template adalah citra gray wajah yang diambil secara frontal, dimana wajah yang digunakan sebagai template wajah adalah wajah yang tidak terhalang obyek apapun.

3.3 Desain Perancangan Graphical User Interface (GUI)

GUI didesain untuk memfasilitasi sistem operasi program yang interaktif. Pada pembuatan skripsi ini desain tampilan yang di gunakan adalah sebagai berikut :

Gambar 3.10 Desain Perancangan GUI Di Matlab

Keterangan :

Angka 1 : Static Text untuk judul Skripsi.

Angka 2 : Push Button1 untuk menampilkan citra yang akan di proses

Angka 3 : Push Button2 untuk melakukan proses segmentasi

Angka 4 : Push Button3 untuk melakukan proses deteksi wajah

Angka 5 : Axes 1,2,3 untuk tampilan citra

BAB IV

IMPLEMENTASI DAN ANALISA PROGRAM

4.1. Implementasi Sistem

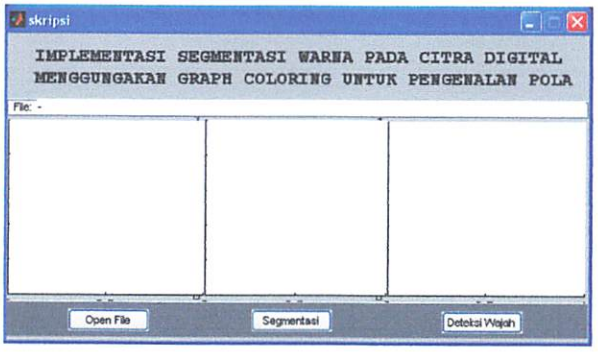
Implementasi dilakukan dengan menerapkan hasil desain yang telah dibuat kedalam bahasa pemograman (*Coding*) Matlab 7.0.4, sehingga prosedur-prosedur yang telah dibuat dapat dimengerti oleh mesin dan menghasilkan keluaran seperti apa yang diharapkan. Berikut ini adalah perlengkapan yang digunakan dalam implementasi system, yang dtunjukkan pada tabel 4.1

Tabel 4.1. Spesifikasi Perlengkapan Implementasi

No.	Perlengkapan	Spesifikasi	Keterangan
1	Sofware	Sistem Operasi	**Windows xp**
		Bahasa Pemograman	**Matlab 7.04**
		Program Tambahan	**Adobe Photoshop CS2**
2	Notebook	Processor	**Dual Core 2.0 GHz**
		Memori	**.100 GB DDR2**
		Hard disk	**250 GB**

4.1.1. Tampilan Program

Pada program ini, akan terdapat beberapa komponen yang digunakan untuk mempermudah pengguna menggunakan program ini seperti yang ditampilkan pada gambar 4.1



Gambar 4.1 Tampilan Program

Pada program ini terdapat beberapa komponen dan beberapa file .m. File .m tersebut yaitu Segmentasi dan DeteksiWajah yang berisi callback dari GUI dan sebagai interasi komponen-komponen yang digunakan pada GUI program.

4.1.2. Komponen-Komponen Pada Program

Dalam program ini terdapat beberapa komponen yang telah disediakan oleh Matlab 7.0.4. Berikut ini adalah komponen-komponen yang ada pada program,ditunjukkan pleh tabel 4.2 :

Tabel 4.2. Komponen Pada Program

No.	Komponen	Gambar
1	Push button Open File	Open File
2	Push button segmentasi	Segmentasi
3	Push button Deteksi Wajah	Deteksi Wajah

4.1.3. Cara Kerja Komponen-Komponen Program

Cara kerja program ini adalah, pertama-tama user melakukan proses inputan gambar dimana gambar bisa diambil lansung dari database gambar yang sudah disimpan sebelumnya. Selanjutnya dalam prosesnya user akan dihadapkan pada pilihan-pilihan komponen yang ada pada program yaitu :

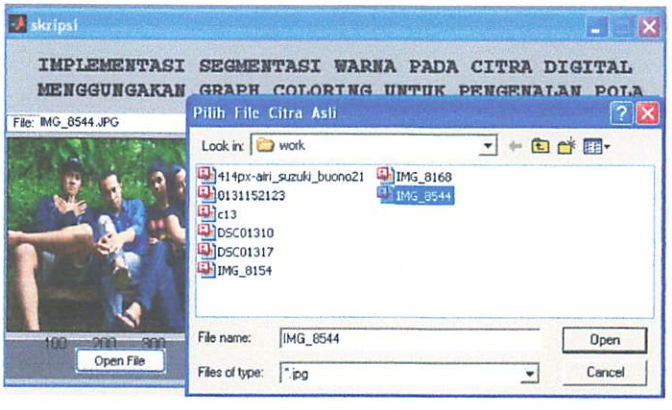
1. Push button Open File untuk mengambil gambar dari database gambar yang akan ditampilkan di jendela pertama. Berikut ini adalah script dari komponen push button Open File :

```
% --- Executes on button press in open.
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
%  Memilih Citra
[FileName,PathName] = uigetfile({'*.jpg'}, 'Pilih File Citra
Asli');
if isequal(FileName, 0) errordlg('Error...!!!','No File Selected');
return;
else
```

```
handles.data1=imread(fullfile(PathName,FileName));
guidata(hObject,handles);
handles.current_data1=handles.data1;
% Menampilkan Citra Pada Citra Masukan
axes(handles.axes1); image(handles.current_data1);
end

% Membaca Spesifikasi Citra
set(handles.tlog,'String',FileName);
set(handles.kosong1,'String',FileName);
set(handles.kosong2,'String',size(handles.data1,1));
set(handles.kosong3,'String',size(handles.data1,2));
```

Gambar 4.2 adalah tampilan hasil dari proses pengambilan gambar dari window open dialog dengan nama window "Open File",.



Gambar 4.2 Tampilan Push Button Open File

2. Push button Segmentasi berisi program proses segmentasi citra dengan metode graph coloring. Dalam proses segmentasi citra dengan metode graph coloring, dapat dilihat pada citra masukan terdapat citra dengan batas-batas yang belum jelas hanya dengan melihat menggunakan penglihatan mata saja. Setelah melakukan pengaturan menggunakan metode graph coloring, maka terlihatlah hasil proses tersebut pada citra keluaran yakni diperoleh batas-batas atau daerah-daerah terpisah dimana setiap daerah adalah homogen dan mengacu pada sebuah kriteria keseragaman yang jelas, hasil dari segmentasi tersebut akan dtampilkan pada jendela kedua.

Berikut ini adalah script dari komponen push button Segmentasi :

```
% --- Executes on button press in segment.
function segment_Callback(hObject, eventdata, handles)
% hObject    handle to segment (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


 image=handles.data1;
 delete 'temp.ppm';
delete 'segmentasi.ppm';
 imwrite(image,'temp.ppm','ppm');
```

```
% Proses Segmentasi Citra
[status, result]=system(['segment.exe 0.5 1000 100 temp.ppm
segmentasi.ppm']);

result=imread('segmentasi.ppm');
 if status~=0
errordlg('segmentasi : Sistem','Eksekusi sistem error - check fungsi
"Segment"!')

end
handles.result=result;
guidata(hObject,handles);
handles.current_result=handles.result;

% Menampilkan Citra pada Citra Keluaran
axes(handles.axes2);
imshow(handles.current_result);
```

Gambar 4.3 adalah tampilan hasil dari proses segmentasi :

Gambar 4.3 Tampilan Hasil Push Button Segmentasi

3. Push button Deteksi Wajah berisi program untuk melakukan proses deteksi wajah dimana dalam proses deteksi wajah ini akan melalui beberapa tahapan pemrosesan deteksi wajah mulai dari tahapan pencarian wilayah wajah (processregion.m) sampai pada tahapan template matching (faceInfo.m) hingga didapat hasil deteksi yang diharapkan. Hasil keluaran dari proses deteksi ini adalah figure-figure yang berisi hasil dari proses deteksi wajah dengan citra grayscale yang mengandung template wajah diatasnya dan wajah citra asli yang berhasil dideteksi dengan tanda segi empat yang mengelilingi citra tersebut, hasilnya deteksi tersebut akan di tampilkan pada jendela ke tiga. Berikut ini adalah script dari komponen push button deteksi wajah :

```
% --- Executes on button press in detwajah.
function detwajah_Callback(hObject, eventdata, handles)
% % hObject    handle to detwajah (see GCBO)
```

```
% % eventdata  reserved - to be defined in a future version of  MATLAB
% % handles    structure with handles and user data (see GUIDATA)
%
% %Membaca koordinat wajah yang mungkin adalah kosong:
% FaceCoord=[];
%
% %Menbaca template gambar wajah:
% frontalmodel=imread('frontal.tif');
%
% imsource=handles.Img;
%
% %Segmentasi warna kulit menjadi hitam putih:
% skincolor=handles.Img;
% BW=im2bw(skincolor);
% %figure,imshow(BW),title('Segmentasi Hitam Putih');
%
% %Mengubah gambar asli menjadi sekala abu2:
% imsourcegray=rgb2gray(imsource);
% %figure,imshow(imsourcegray),title('grayscale');
%
% %Mengambil hasil gambar dari warna kulit yg tersegmentasi:
% [L,numobj]=bwlabel(BW,8);
% map=[0 0 0;jet(numobj)];
```

BAB V

PENUTUP

Dari beberapa uraian yang telah dikemukakan pada bab-bab sebelumnya, dapat diambil kesimpulan dan saran sebagai berikut :

5.1. Kesimpulan

Berdasarkan hasil pengujian yang dilakukan pada proyek akhir ini, maka disimpulkan bahwa :

1. Tingkat pencahayaan gambar sangat mempengaruhi dalam proses segmentasi warna.
2. Kebanyakan kesamaan warna dari hsasil proses segmentasi warna karena terdapatnya wilayah-wilayah yang memiliki keserupaan warna antar objek dan tingkat pencahayaan citra.
3. Deteksi wajah dapat diimplementasikan dengan menggunakan metode template matching berdasarkan segmentasi warna kulit dengan tingkat keberhasilan deteksi sebesar 68,57 pada citra yang diuji sebanyak 12 citra gambar.
4. Besar kecil ukuran file citra dapat mempengaruhi hasil segmentasi warna
5. Pencahayaan, warna kulit dan posisi letak citra gambar sangat mempengaruhi dalam proses deteksi.
6. Deteksi Wajah hanya bisa mentedeksi wajah manusia, untuk selain wajah manusia tidak dapat terdeteksi karena tamplate yang digunakan untuk

mendeteksi wajah adalah template wajah manusia. Sehingga untuk gambar hewan atau pemandangan alam akan tidak terdeteksi wajah manusia.

7. Kebanyakan kesalahan dalam proses deteksi karena terdapatnya wilayah-wilayah yang memiliki keserupaan warna dengan wilayah kemungkinan warna kulit seperti warna pakaian yang dikenakan ataupun warna latar belakang.
8. Dalam beberapa kasus, kesalahan dalam proses deteksi wajah disebabkan oleh penetapan batasan untuk jumlah hole (lubang) yang lebih besar atau sama dengan satu yang akan diproses lebih lanjut dalam proses penentuan wilayah wajah melalui proses penentuan wilayah kulit, sehingga setiap wilayah kulit yang mempunyai jumlah lubang lebih dari satu atau sama dengan satu, akan mempunyai kemungkinan besar terdeteksi sebagai wilayah wajah, meskipun wilayah tersebut sebenarnya bukanlah wilayah wajah.
9. Ukuran besar kecilnya resolusi dan besar kecilnya file citra gambar mempengaruhi kecepatan dalam proses segmentasi dan deteksi. Karena semakin besar ukuran citra gambar yang dinputkan semakin lama proses segmentasi dan deteksi yang dijalankan.

5.2. Saran

Setelah melakukan evaluasi terhadap sistem secara keseluruan, penulis berharap skripsi ini dapat dikembangkan lebih lanjut dengan saran-saran pengembangan sebagai berikut:

1. Mencoba banyak preprocessing lainnya, sehingga dapat mengetahui apakah ada preprocessing yang lebih baik lagi
2. Menambah metode lain pada proses deteksi wajah agar kesalahan dalam deteksi dapat diminimalkan.

L A M P I R A N

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No.2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI

FAKULTAS TEKNOLOGI INDUSTRI

Nama : HENDRY KUSNOYO

NIM : 05. 12. 723

Jurusan : Teknik Elektro S-1

Konsentrasi : Teknik Komputer dan Informatika S-1

Judul Skripsi : **IMPLEMENTASI SEGMENTASI WARNA PADA CITRA DIGITAL MENGGUNAKAN METODE GRAPH COLORING UNTUK PENGENALAN POLA**

Dipertahankan di hadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari : Kamis

Tanggal : 18 Agustus 2011

Dengan Nilai : 82,50 (A)

Panitia Ujian Skripsi:

Ketua Majelis Penguji	Sekretaris Majelis Penguji
Ir. Yusuf Ismail Nakhoda, MT	**Dr. Eng. Aryuanto S. ST, MT**
NIP.Y.1018800189	**NIP.Y.1030800417**

Anggota Penguji:

Penguji I	Penguji II
Sandy Nataly Mantja, S.Kom	**M.Ibrahim Ashari, ST, MT**
NIP.P.1030800418	**NIP.P.1030100358**

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

FORMULIR PERBAIKAN SKRIPSI

Nama : HENDRY KUSNOYO
NIM : 0512723
Jurusan : Teknik Elektro S-1
Konsentrasi : Komputer dan Informatika
Masa Bimbingan : 17 Desember 2010 s/d 17 Juni 2011
Judul Skripsi : IMPLEMENTASI *SEGMENTASI WARNA PADA CITRA DIGITAL MENGGUNAKAN METODE GRAPH COLORING UNTUK PENGENALAN POLA*

Tanggal	Penguji	Uraian	Paraf
18 Februari 2011	Penguji 1	1. Tambahkan daftar pustaka sampai dengan 20 2. Lesimpulan harus di buktikan di BAB IV 3. Di tulis jika gambar wajah sebagian, hewan dan tumbuhan si BAB IV dan kesimpulan	
	Penguji 2	1. Tmabahkan nomer gambar dan tabel pada keterangan kalimat	

Disetujui,

Dosen Penguji I

Sandy Nataly M, S.Kom.
NIP.P. 1030800418

Dosen Penguji II

M.Ibrahim Ashari, ST, MT.
NIP.P. 1030100358

Mengetahui,

Dosen Pembimbing

Dr. Eng. Aryuanto Soetedjo, ST, MT
NIP.P.1030800417

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo, Km. 2
MALANG

PERMOHONAN PERSETUJUAN SKRIPSI

Yang betanda tangan dibawah ini :

N a m a	: HENDRY KUSNOYO
N I M	: 05.12.723
Semester	: 10 (sepuluh)
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: **TEKNIK ELEKTRONIKA**
	TEKNIK ENERGI LISTRIK
	TEKNIK KOMPUTER DAN INFORMATIKA
Alamat	: Jl. Kebalen Wetan no 2 malang

Dengan ini kami mengajukan permohonan untuk mendapatkan persetujuan untuk membuat ***SKRIPSI Tingkat Sarjana.*** Untuk melengkapi permohonan tersebut, bersama kami lampirkan persyaratan-persyaratan yang harus dipenuhi.
Adapun persyaratan-persyaratan pengambilan ***SKRIPSI*** adalah sebagai berikut :

1. Telah melaksanakan semua praktikum sesuai dengan konsentrasinya (.....)
2. Telah lulus dan menyerahkan Laporan Praktek Kerja (.....)
3. Telah lulus seluruh mata kuliah keahlian (MKB) sesuai konsentrasinya (.....)
4. Telah menempuh mata kuliah ≥ 134 sks dengan IPK ≥ 2 dan tidak ada nilai E (.....)
5. Telah mengikuti secara aktif kegiatan seminar skripsi yang diadakan Jurusan (.....)
6. Memenuhi persyaratan administrasi (.....)

Demikian permohonan ini untuk mendapatkan penyelesaian lebih lanjut dan atas perhatiannya kami ucapkan terima kasih.

Telah diteliti kebenaran data tersebut diatas
Recording Teknik Elektro

(.....)

Malang, ...200
Pemohon

(HENDRY KUSNOYO)

Disetujui
Ketua Jurusan Teknik Elektro

Ir. F. Yudi Limpraptono, MT
NIP. P. 1039500274

Mengetahui
Dosen Wali

(.....)

Bagi mahasiswa yang telah memenuhi persyaratan mengambil SKRIPSI agar membuat proposal dan mendapat persetujuan dari Ketua Jurusan/Sekretaris Jurusan T. Elektro S-1

1. IPK 3,42 - 252
2. 136
3. 4 praktikum

Form. S-1a

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

LEMBAR PENGAJUAN JUDUL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik/Teknik Elektronika/Teknik Komputer & Informatika*)

1.	Nama Mahasiswa: HENDRY KUSNOYO			Nim: 05.12.723
2.	Waktu Pengajuan	Tanggal:	Bulan:	Tahun:
3.	Spesifikasi Judul (berilah tanda silang)**) a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri e. Elektronika & Komponen f. Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainnya			
4.	Konsultasikan judul sesuai materi bidang ilmu kepada Dosen*) Ir. Aryuanto S, ST MT	Ketua Jurusan Ir. F. Yudi Limpraptono,MT NIP. P. 1039500274		
5.	Judul yang diajukan mahasiswa:	Implementasi Segmentasi Citra menggunakan Metode Graph		
6.	Perubahan judul yang disetujui Dosen sesuai materi bidang ilmu	Implementasi Segmentasi warna pada Citra Digital menggunakan Metode Graph Coloring		
7.	Catatan: ..... Persetujuan Judul skripsi yang dikonsultasikan kepada Dosen materi bidang ilmu	Disetujui 19/7/ 2010 Dosen		

Perhatian:

1. Formulir pengajuan ini harap dikembalikan kepada jurusan paling lambat satu minggu setelah disetujui kelompok dosen keahlian dengan dilampirkan proposal skripsi beserta persyaratan skripsi sesuai form S-1
2. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ...atau g sesuai bidang keahlian

Form S-2

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-gura No.2
M A L A N G

Lampiran : 1 (satu) berkas
Pembimbing Skripsi

Kepada : Yth. Bapak **Dr.Eng. Aryuanto Soetedjo, ST, MT**
Dosen Institut Teknologi Nasional
M AL A N G

Yang bertanda tangan di bawah ini :

Nama	: HENDRY KUSNOYO
Nim	: 0512723
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Komputer dan Informatika

Dengan ini mengajukan permohonan kiranya Bapak bersedia menjadi Dosen Pembimbing Utama / ~~Pendamping~~ *), untuk penyusunan Skripsi dengan judul (proposal terlampir) :

IMPLEMENTASI SEGMENTASI WARNA PADA CITRA DIGITAL

MENGGUNAKAN

METODE GRAPH COLORING

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh Ujian Akhir Sarjana Teknik.

Demikian permohonan kami dan atas kesediaan Bapak kami ucapkan terima kasih.

Malang, 20 juli 2010

Mengetahui
Ketua Jurusan Teknik Elektro

Ir. F. Yudi Limpraptono, MT
NIP.Y. 1039 5900274

Hormat kami

Hendry Kusnoyo

***)coret yang tidak perlu**

Form S-3 a

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-gura No.2
MALANG

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari mahasiswa :

Nama	:	HENDRY KUSNOYO
Nim	:	0512723
Semester	:	10 (sepuluh)
Jurusan	:	Teknik Elektro S-1
Konsentrasi	:	Teknik Komputer dan Informatika

Dengan ini Menyatakan bersedia / tidak bersedia*) Membimbing Skripsi dari mahasiswa tersebut, dengan judul:

IMPLEMENTASI SEGMENTASI WARNA PADA CITRA DIGITAL MENGGUNAKAN METODE GRAPH COLORING

Demikian surat Pernyataan ini kami buat agar dapat dipergunakan seperlunya.

Malang, 20 Juli 2010

Kami yang membuat pernyataan,

Catatan :
Setelah disetujui agar formulir ini
Diserahkan mahasiswa/I yang bersangkutan
Kepada Jurusan untuk diproses lebih lanjut.

***)coret yang tidak perlu**

Dr. Eng. Aryuanto S, ST, MT
NIP.Y. 1030800417

Form S-3 b

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo 10 KM
MALANG

FORMULIR BIMBINGAN SKRIPSI

ama : HENDRY KUSNOYO

IM : 05.12.723

lasa Bimbingan : Agustus 2010 s/d September 2011

idul Skripsi : IMPLEMENTASI SEGMENTASI WARNA PADA CITRA DIGITAL MENGGUNAKAN METODE GRAPH COLORING UNTUK PENGENALAN POLA

No	Tanggal	Uraian	Paraf Pembimbing
1	4 Mei 2011	Revisi BAB I : batasan masalah	
2	4 Mei 2011	Laporan BAB II	
3	19 Juni 2011	Demo Program	
4	16 Agustus 2011	Laporan BAB III	
5	16 Agustus 2011	Laporan BAB IV	
6	16 Agustus 2011	Laporan BAB V	
7	16 Agustus 2011	Laporan BAB VI	
8			
9			
10			

Malang,
Dosen Pembimbing

(Dr. Eng. Aryuanto Soetedjo, ST, MT)
NIP.P 1030800417

FORM S-4b

pencahayaan yang mirip pada tiap titik pixel citra. Sehingga hasil segmentasi tidak dapat sempurna. Sedangkan untuk hasil deteksi wajah pada gmbar selain manusia tidak dapat terdeteksi untuk wajah manusia, hal ini disebabkan pada gambar tersebut tidak terdapat gambar wajah manusia.

Dari hasil pengujian yang dilakukan pada 12 citra wajah yang dilakukan berdasarkan hasil analisa diketahui bahwa prosentase keberhasilan dalam proses deteksi wajah dengan metode template metching ini sebesar 73,68%, dimana prosentase keberhasilan deteksi ini didapat dengan persamaan sebagai berikut :

$$\text{Prosentase keberhasilan} : \frac{\sum \text{wajah yang berhasil dikenali}}{\sum \text{wajah pada gambar}} \times 100\%$$

$$: \frac{24}{35} \times 100\%$$

$$: 68{,}57\ \%$$

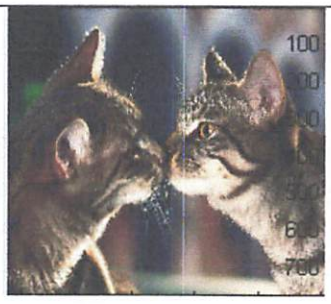
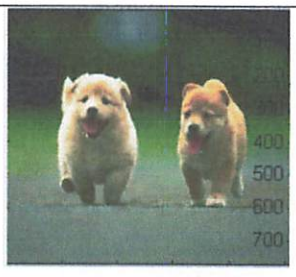
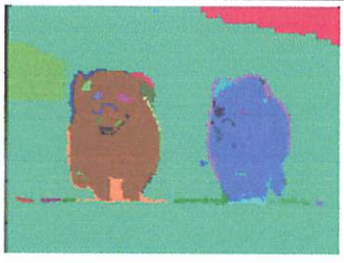
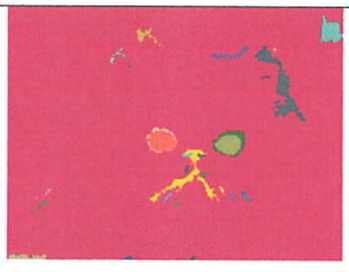
dapat dilihat pada tabel 4.3. dimana semakin besar ukuran file citra yang di inputkan proses segmentasi dan deteksi akan semakin lama

Pada pengujian segmentasi tabel 4.4. terdapat banyak hasil segmentasi yang berwarna seragam, hal ini disebabkan tingkat kecerahan pencahayaan yang mirip pada tiap titik pixel citra. Sehingga hasil segmentasi tidak dapat sempurna.

Sementara dalam pengujian deteksi wajah tabel 4.5. proses kesalahan deteksi seperti warna kulit yang agak gelap, posisi citra yang berubah, kondisi citra dan pencahayaan, serta sudut kemiringn yang terlalu besar mempengaruhi dalam proses deteksi ini dimana untuk kesalahan deteksi ada wilayah lain yang memiliki keserupaan dengan wilayah kemungkinan kulit yang terdeteksi sabagai warna kulit, kesalahan deteksi ini salah satunya disebabkan karena posisi dan pencahayaan yang mengenai citra tersebut tidak merata sehingga pada wilayah kulit yang tersegmentasi diwajah ada yang menyatu dengan wilayah bukan kulit wajah hal ini tidak sesuai dengan aturan program yang telah ditetapkan pada bab sebelumnya dimana citra yang berhasil terdeteksi adalah jika citra tersebut memiliki lubang lebih dari satu lubang atau sama dengan satu dalam daerah kulit seperti lubang mata atau mulut. Pada program deteksi wajah ini faktor utama yang sangat berpengaruh dalam proses deteksi wajah adalah pancahayaan, latar belakang, warna kulit, posisi dan keadaan citra.

Pada pengujian segmentasi untuk berbagai ukuran pada tabel 4.6 menunjukkan adanya perbedaan hasil segmentasi yang sangat mencolok antara file berukuran 320 dengan 640. Perbedaan tersebut disebabkan oleh jumlah dan ukuran pixsel pada kedua ukuran tersebut berbeda jauh. Sehingga menimbulkan perbedaan warna pada hasil segmentasi yang sangat mencolok.

Pada pengujian segmentasi dan deteksi wajah pada gambar selain manusia yang di tunjukkan pada tabel 4.7 diatas menunjukkan bahwa hasil segmentasi pada gambar selain manusia masih terdapat keseragaman warna segmentasi, hal ini disebabkan tingkat kecerahan

3			Tidak ada hasil
4			Tidak ada hasil
5			Tidak ada hasil
6			Tidak ada hasil

4.2.6 Analisa Data

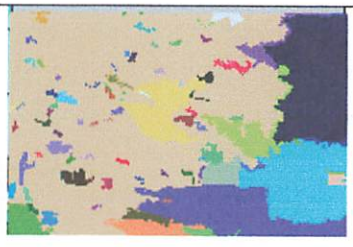
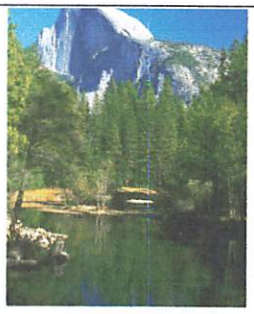
Dari proses pengujian yang telah dilakukan dalam program segmentasi dan deteksi wajah dapat diambil suatu analisa bahwa ukuran file citra yang di inputkan dapat mempengarui kecepatan dalam proses segmentasi dan deteksi wajah, hasil dari pengujian ini

5			

4.2.5 Pengujian Segmentasi Dan Deteksi Wajah pada Gambar Selain Manusia

Pada tabel 4.7 adalah hasil dari pengujian yang dilakukan pada gambar selain wajah manusia,pengujian ini dilakukan untuk mengetahui hasil segmentasi dan keakuratan deteksi wajah pada gambar selain wajah manusia yang akan digunakan untuk pengambilan keputusan.

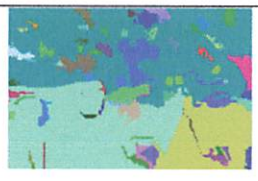
Tabel 4.7 Pengujian Segmentasi Dan Deteksi Wajah pada Gambar Selain Manusia

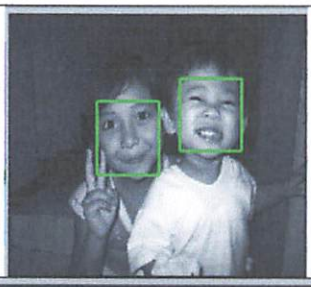
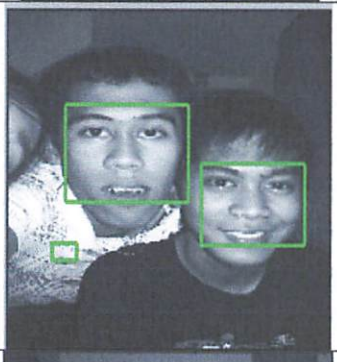
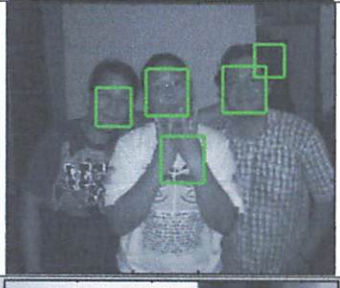
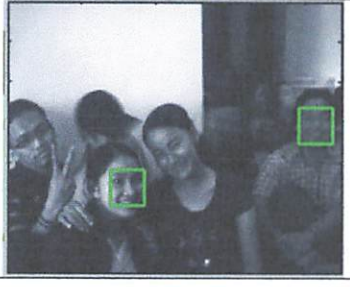
No	Citra Inputan	Segmentasi	Deteksi Wajah
1			Tidak ada hasil
2			Tidak ada hasil

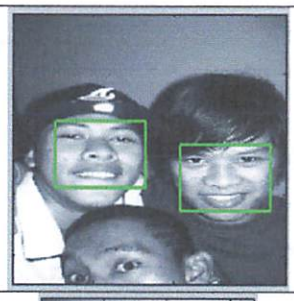
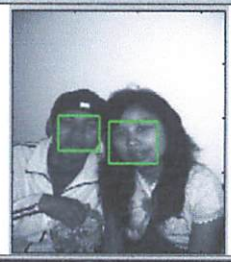
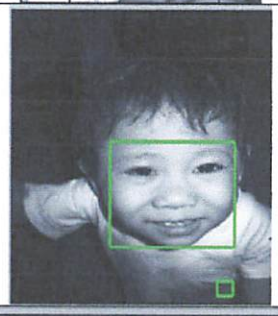
4.2.4 Pengujian Segmentasi Berbagai Ukuran Gambar

Pengujian ini dilakukan untuk mengetahui perbedaan hasil proses segmentasi yang akan digunakan untuk pengambilan kesimpulan, yang di tampilkan pada tabel 4.6

Tabel 4.6 Pengujian Program Segmentasi Berbagai Ukuran Gambar

No	Citra Inputan	Segmentasi	
		320	640
1			
2			
3			
4			

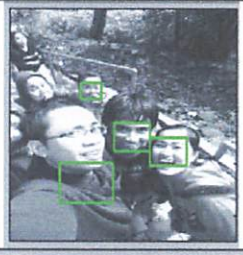
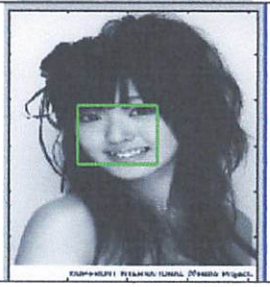
9		
10		
11		
12		

4		
5		
6		
7		
8		

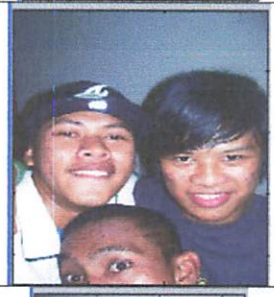
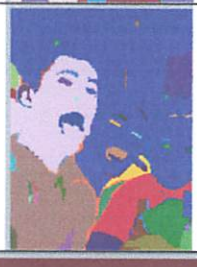
4.2.3 Pengujian Program Deteksi Wajah

Pengujian ini dilakukan untuk mengetahui keakuratan proses deteksi wajah dengan template dalam berbagai kondisi yang dapat mempengaruhi keberhasilan dalam proses deteksi wajah sehingga nantinya dapat digunakan dalam mengambil kesimpulan, yang di tampilkan pada tabel 4.5.

Table 4.5 Pengujian program Deteksi Wajah

No	Citra Inputan	Hasil Deteksi Wajah
1		
2		
3		

8		 100 200 300 400
9		 100 200 300 400
10		
11		 100 200 300 400 500 600
12		 50 100 150 200 250 300

3		
4		
5		
6		
7		

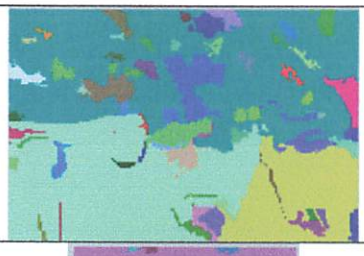
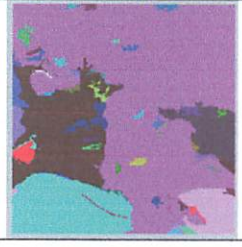
Tabel 4.3 Pengujian progam dengan ukuran File yang Berbeda

Ukuran File Citra	Waktu Proses Segmentasi	Waktu Poses Pengenalan Pola
448 x 299	± 3 detik	± 30 detik
640 x 427	± 5 detik	± 70 detik
1024 x 683	± 10 detik	± 100 detik

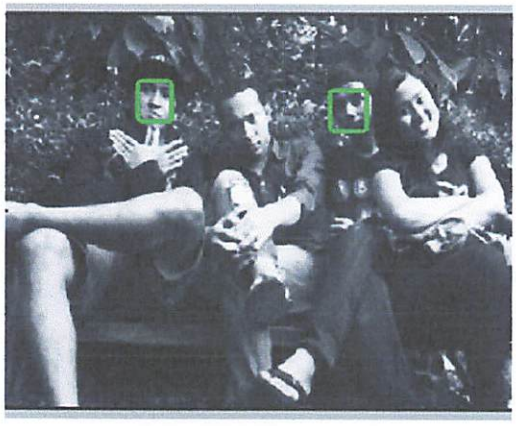
4.2.2 Pengujian Program Segmentasi

Dalam pengujian ini dilakukan suatu proses segmentasi citra dengan metode graph coloring nantinya dapat digunakan dalam mengambil kesimpulan, yang di tampilkan pada tabel 4.4.

Table 4.4 Pengujian Prgogram Segmentasi

No	Citra Inputan	Hasil Segmentasi
1		
2		

Hasil akhir dari keluaran proses deteksi wajah adalah dengan diberikannya kotak inspeksi (segi empat warna merah) pada citra inputan asli, yang menunjukkan wilayah wajah berhasil terdeteksi,seperti yang di tampilakan pada gambar 4.7.



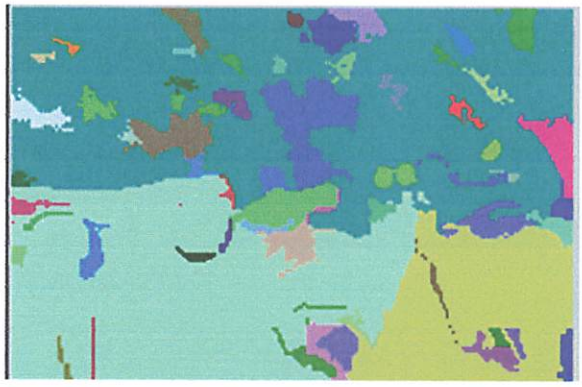
Gambar 4.7 Tampilan hasil deteksi wajah

4.2 Pengujian Sistem

Pengujian ini dilakukan untuk mengetahui kinerja program segmentasi warna pada citra digital metode graph coloring untuk pengenalan pola.

4.2.1 Pengujian Program Dengan Ukuran File Yang Berbeda

Pada pengujian kali ini akan dilakukan dengan perbandingan ukuran file citra dengan waktu proses yang diperlukan dalam proses segmentasi dan pengenalan pola. Dari Pengujian yang telah dilakukan didapatkan hasil yang dtampilkan pada tabel 4.3:



Gambar 4.6 Tampilan hasil segmentasi

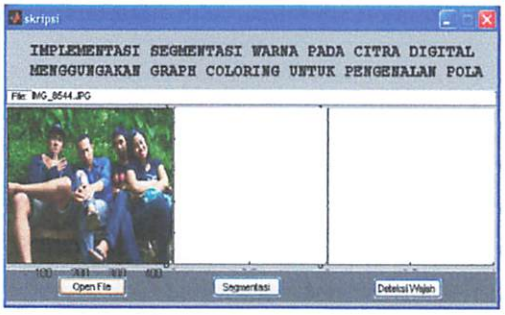
Proses selanjutnya setelah penginputan citra dan segmentasi, jika menekan tombol deteksi wajah program akan secara otomatis memproses citra inputan tersebut. Dalam prosesnya, tahapan yang akan dilakukan oleh program dalam proses deteksi wajah adalah mencari daerah kamungkinan kulit, dimana hasil citra kemungkinan kulit akan menunjukkan bahwa wilayah kulit mempunyai bagian yang lebih terang dibandingkan dengan wilayah lain yang ada dalam suatu citra.

Proses selanjutnya adalah melakukan pelabelan wilayah dalam citra biner dengan wilayah putih menunjukkan wilayah kulit dan wilayah hitam sebagai wilayah bukan kulit.

Kemudian dari proses pelabelan wilayah dilakukan proses template matching yang digunakan untuk mendapatkan nilai korelasi-silang (cross-corelation) antara wilayah kulit yang diindikasikan sebagai wilayah wajah template. Setelah itu sistem akan menentukan wilayah wajah baru disini adalah citra grayscale, dengan wilayah yang dinyatakan sebagai kulit wajah diganti dengan wajah template.

4.1.4. Hasil Implementasi

Gambar 4.5 adalah hasil implementasi program dan berikut adalah tampilan pada saat proses input citra dari database citra dimasukkan.



Gambar 4.5 Tampilan Inputan File

Setelah melakukan penginputan citra, jika tombol segmentasi ditekan program akan secara otomatis memproses citra inputan tersebut. Dalam prosesnya, tahapan yang akan dilakukan oleh program dalam proses segmentasi adalah proses segmentasi citra dengan metode graph coloring, dapat dilihat pada citra masukan terdapat citra dengan batas-batas yang belum jelas hanya dengan melihat menggunakan penglihatan mata saja. Setelah melakukan pengaturanm enggunakan metode graph coloring, maka terlihatlah hasil proses tersebut pada citra keluaran yakni diperoleh batas-batas atau daerah-daerah terpisah dimana setiap daerah adalah homogen dan mengacu pada sebuah kriteria keseragaman yang jelas,seperti yang dtampilkan pada gambar 4.6.

```
    ind=find(isinf(vec));
    a=200;
    vec(ind)=sign(vec(ind))*a;

    h1=line([vec(1) vec(2)],[vec(3) vec(3)]);
    h2=line([vec(2) vec(2)],[vec(3) vec(4)]);
    h3=line([vec(1) vec(2)],[vec(4) vec(4)]);
    h4=line([vec(1) vec(1)],[vec(3) vec(4)]);

    h=[h1 h2 h3 h4];
    set(h,'Color',col);
    set(h,'LineWidth',t)
  end
end
```

Gambar 4.4 adalah tampilan hasil dari proses deteksi wajah :

Gambar 4.4 Tampilan Hasil Push Button Deteksi Wajah

```
x=imread(get(handles.tlog,'String'));


axes(handles.axes3);
imshow(x)


if size(x,3)>1
   x=rgb2gray(x);
end
x=double(x);
[output,count,m,svec]=facefind(x);
imagesc(x)
colormap(gray)


col=[1 0 0];
col=[0 1 0];
t=2;
N=size(output,2);
if (N>0)
   for i=1:N
      x1=output(1,i);
      x2=output(2,i);
      y1=output(3,i);
      y2=output(4,i);
      vec=[x1 x2 y1 y2];
```

```
%
[RectCoord,imsourcegray]=processregion(imsourcegray,bwsegment,num
holes,frontalmodel);
%
% %Untuk menambah facem vektor wajah koordinat
% if (RectCoord~=-1)
%     FaceCoord=[FaceCoord;RectCoord];
% end;
%     end;
% end
%
% %Final Output:
% figure,imshow(imsource),title('Hasil Deteksi Wajah');
%
% %Untuk menampilkan persegi panjang jika menemukan wajah dalam
gambar
% [numfaces x]=size(FaceCoord);
% for i=1 :numfaces,
%     hd=rectangle('Position',FaceCoord(i,:));%tampilin kotak
%     set(hd,'edgecolor','R');%tampilin box warna putih
% end;
%
```

```
% %figure,imshow(L+1,map,'notruesize'),title('Segmentasi Daerah Warna
Kulit');
%
% for i=1:numobj
%
%   %Untuk menghitung koordinat untuk wilayah ini
%    [x,y]=find(bwlabel(BW)==i);
%
%   %Untuk mendapatkan gambar yang hanya memiliki daerah ini,
sisanya hitam
%    bwsegment=bwselect(BW,y,x,8);
%
%   %Untuk menghitung segmen dalam daerah ini
%    [L,numobjs]=bwlabel(bwsegment,4);
%
%   %Untuk mendapatkan jumlah lubang
%    numfeatures=bweuler(bwsegment,4);
%    numholes=1-numfeatures;
%
%   %jika wajah perlu memiliki lebih dari satu lubang, jika tidak
%   %akan dibuang
%    if (numholes>=1)
%
% %Koordinat persegi panjang (sambung ke function processregion):
```

LAMPIRAN

Tampilan Utama

```
function varargout = skripsi(varargin)
% SKRIPSI M-file for skripsi.fig
%      SKRIPSI, by itself, creates a new SKRIPSI or raises the
existing
%      singleton*.
%
%      H = SKRIPSI returns the handle to a new SKRIPSI or the
handle to
%      the existing singleton*.
%
%      SKRIPSI('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in SKRIPSI.M with the given
input arguments.
%
%      SKRIPSI('Property','Value',...) creates a new SKRIPSI
or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before skripsi_OpeningFunction gets
called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to skripsi_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help skripsi

% Last Modified by GUIDE v2.5 09-Aug-2011 00:39:03

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @skripsi_OpeningFcn, ...
                   'gui_OutputFcn',  @skripsi_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before skripsi is made visible.
function skripsi_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to skripsi (see VARARGIN)

% Choose default command line output for skripsi
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes skripsi wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command
line.
function varargout = skripsi_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

 % --- Executes on button press in open.
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

%   Memilih  Citra
[FileName,PathName]  =  uigetfile({'*.jpg'},  'Pilih  File
Citra  Asli');
if isequal(FileName,  0) errordlg('Error...!!!','No  File
Selected'); return;
else
handles.data1=imread(fullfile(PathName,FileName));
```

```
guidata(hObject,handles);
handles.current_data1=handles.data1;
%   Menampilkan  Citra  Pada Citra  Masukan
axes(handles.axes1);
image(handles.current_data1);
end
%   Membaca  Spesifikasi  Citra
set(handles.tlog,'String',FileName);
set(handles.segment,'Enable','on');
set(handles.detwajah,'Enable','off');

 %set(handles.kosong1,'String',FileName);
%set(handles.kosong2,'String',size(handles.data1,1));
%set(handles.kosong3,'String',size(handles.data1,2));

% --- Executes on button press in segment.
function segment_Callback(hObject, eventdata, handles)
% hObject    handle to segment (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

image=handles.data1;

delete 'temp.ppm';
delete 'segmentasi.ppm';

imwrite(image,'temp.ppm','ppm');

%   Proses  Segmentasi  Citra
[status,  result]=system(['segment.exe  0.5  1000  100
temp.ppm segmentasi.ppm']);
result=imread('segmentasi.ppm');
set(handles.detwajah,'Enable','on');
if status~=0
errordlg('segmentasi  : Sistem','Eksekusi  sistem  error    -
check fungsi  ''Segment''!')
set(handles.detwajah,'Enable','off');
end

handles.result=result;
guidata(hObject,handles);
handles.current_result=handles.result;

%   Menampilkan  Citra  pada Citra  Keluaran
axes(handles.axes2);
imshow(handles.current_result);

% --- Executes on button press in detwajah.
function detwajah_Callback(hObject, eventdata, handles)
% % hObject    handle to detwajah (see GCBO)
% % eventdata  reserved - to be defined in a future version of
MATLAB
% % handles    structure with handles and user data (see
GUIDATA)
```

```
%
% %Membaca koordinat wajah yang mungkin adalah kosong:
% FaceCoord=[];
%
% %Menbaca template gambar wajah:
% frontalmodel=imread('frontal.tif');
%
% imsource=handles.Img;
%
% %Segmentasi warna kulit menjadi hitam putih:
% skincolor=handles.Img;
% BW=im2bw(skincolor);
% %figure,imshow(BW),title('Segmentasi Hitam Putih');
%
% %Mengubah gambar asli menjadi sekala abu2:
% imsourcegray=rgb2gray(imsource);
% %figure,imshow(imsourcegray),title('grayscale');
%
% %Mengambil hasil gambar dari warna kulit yg tersegmentasi:
% [L,numobj]=bwlabel(BW,8);
% map=[0 0 0;jet(numobj)];
% %figure,imshow(L+1,map,'notruesize'),title('Segmentasi
Daerah Warna Kulit');
%
% for i=1:numobj
%
%    %Untuk menghitung koordinat untuk wilayah ini
%     [x,y]=find(bwlabel(BW)==i);
%
%    %Untuk mendapatkan gambar yang hanya memiliki daerah ini,
sisanya hitam
%     bwsegment=bwselect(BW,y,x,8);
%
%    %Untuk menghitung segmen dalam daerah ini
%     [L,numobjs]=bwlabel(bwsegment,4);
%
%    %Untuk mendapatkan jumlah lubang
%     numfeatures=bweuler(bwsegment,4);
%     numholes=1-numfeatures;
%
%    %jika wajah perlu memiliki lebih dari satu lubang, jika
tidak
%    %akan dibuang
%     if (numholes>=1)
%
% %Koordinat persegi panjang (sambung ke function
processregion):
%
[RectCoord,imsourcegray]=processregion(imsourcegray,bwsegment,
numholes,frontalmodel);
%
% %Untuk menambah facem vektor wajah koordinat
% if (RectCoord~=-1)
%     FaceCoord=[FaceCoord;RectCoord];
% end;
%     end;
```

```
% end
%
% %Final Output:
% figure,imshow(imsource),title('Hasil Deteksi Wajah');
%
% %Untuk menampilkan persegi panjang jika menemukan wajah
dalam gambar
% [numfaces x]=size(FaceCoord);
% for i=1:numfaces,
%     hd=rectangle('Position',FaceCoord(i,:));%tampilin kotak
%     set(hd,'edgecolor','R');%tampilin box warna putih
% end;
%

x=imread(get(handles.tlog,'String'));
temp=x;

if size(x,3)>1
    x=rgb2gray(x);
end
x=double(x);
[output,count,m,svec]=facefind(x);
imagesc(x)
colormap(gray)

col=[1 0 0];
col=[0 1 0];
t=2;
N=size(output,2);
if (N>0)
    axes(handles.axes3);
    imshow(temp);
    for i=1:N
        x1=output(1,i);
        x2=output(2,i);
        y1=output(3,i);
        y2=output(4,i);
        vec=[x1 x2 y1 y2];
        ind=find(isinf(vec));
        a=200;
        vec(ind)=sign(vec(ind))*a;

        h1=line([vec(1) vec(2)],[vec(3) vec(3)]);
        h2=line([vec(2) vec(2)],[vec(3) vec(4)]);
        h3=line([vec(1) vec(2)],[vec(4) vec(4)]);
        h4=line([vec(1) vec(1)],[vec(3) vec(4)]);

        h=[h1 h2 h3 h4];
        set(h,'Color',col);
        set(h,'LineWidth',t)
    end


end
```

```
 % --- If Enable == 'on', executes on mouse press in 5 pixel
border.
% --- Otherwise, executes on mouse press in 5 pixel border or
over open.
function open_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)


% --- Executes during object creation, after setting all
properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
```

Deteksi Wajah

Recsize

```
function [left, right, up, down] = recsize(A)
[m n] = size(A);

left = -1;
right = -1;
up = -1;
down = -1;

for j=1:n,
  for i=1:m,
     if (A(i,j) ~= 0)
       left = j;
       break;
     end;
    end;
  if (left ~= -1) break; end;
end;

for j=n:-1:1,
  for i=1:m,
     if (A(i,j) ~= 0)
       right = j;
       break;
     end;
  end;
  if (right ~= -1) break; end;
end;

for i=1:m,
```

```
    for j=1:n,
     if (A(i,j) ~= 0)
       up = i;
       break;
     end;
    end;
 if (up ~= -1) break; end;
end;

for i= m:-1:1,
  for j=1:n,
     if (A(i,j) ~= 0)
       down = i;
       break;
     end;
  end;
  if (down ~= -1) break; end;
end;

if (left == -1) left =1; end;
if (right == -1) right = n; end;
if (up == -1) up=1; end;
if (down == -1) down = m; end;
```

Processregion

```
%Mengingat citra dengan hanya satu wilayah dia atasnya,
menentukan apakah
%daerah ini menunjukkan wajah atau tidak. Jika demikian,
kembali
%kekoordinat persegi panjang yang akan diambil untuk
menunjukkan wajah yang
%terdeteksi

%Penjelasan :
%Imsourcegray : gambar asli di grayscale
%Bwsegment : gambar hitam putih dengan wilayah yang diinginkan
diatasnya
%Mumholes : jumlah daerah hitam di dalam wilayah
%Frontal model : citra grayscale model wajah frontal saya
function [RectCoord, imsourcegray] =
processregion(imsourcegray, bwsegment, numholes, ...
                      frontalmodel)

  RectCoord = -1;

 %Untuk mendapatkan ukuran gambar
  [m n] = size(bwsegment);

 %Untuk mendapatkan pusat massa dari gambar
  [cx,cy]=center(bwsegment);

 %Untuk mengisi lubang pada gambar biner
  bwnohole=bwfill(bwsegment,'holes');
```

```
 %Untuk mendapatkan kemunkinan hanya wajah dari foto atau
citra masukan
  justface = uint8(double(bwnohole) .* double(imsourcegray));

 %Untuk mendapatkan sudut rotasi
  angle = orient(bwsegment,cx,cy);

  %Untuk menghitung lebar dan tinggi daerah
  bw = imrotate(bwsegment, angle, 'bilinear');
  bw = bwfill(bw,'holes');
  [l,r,u,d] = recsize(bw);
  wx = (r - l +1);  % lebar
  ly = (d - u + 1); % tinggi

  %Untuk mendapatkan rasio antara tiggi (ly) dan lebar (wx)
daerah
   wratio = ly/wx

  %Untuk digunakan jika menemukan daerah yang sangat tinggi
  %Sesuaikan rasio dengan mengurangi ketinggian wilayah
    if (wratio > 1.6)

      ly = floor(1.5 * wx);  %Perhitungan tinggi baru

  %Untuk menghilangkan bagian-bagian dari gambar yang dipotong
setelah
  %menghitung tinggi baru daerah
      [l,r,u,d] = recsize(bwnohole);
      start = floor((u+ly)*cos(-angle*pi/180));
      for i=start:m,
         for j=1:n,
           bwsegment(i,j) = 0;
         end
      end

  %Untuk menghitung koordinat pusat wilayah baru
      [cx,cy]=center(bwsegment);

  %Untuk mendapatkan rasio baru
      wratio = ly/wx;
    end;

  %Untuk menentukan nilai cross-korlasi antara daerah gambar
yang mungkin
  %menunjukkan wajah dan model wajah jika jumlah lubang lebih
besar dari
  %satu. Ratio > 0,8 kondisi menghindari masalah berhubungan
dengan
  %daerah yang terlalau besar
  if (numholes >=1  &  wratio >= 0.8)
    [ccorr,mfit, RectCoord] =
faceInfo(justface,frontalmodel,ly,wx, cx,cy, angle);
   else ccorr = 0;
  end;
```

```
 %Jika memiliki lubang, dan hasil dari korelasi-silang di atas
1,6 ini
 %adalah wajah
 if (ccorr > 0.6 & numholes >= 1)

 %Untuk mendapatkan sebuah gambar dengan seluruh hitam di
daerah lokasi
 %model wajah
    mfitbw = (mfit >=1);

 %Flip nilai gambar, sehingga dapat memiliki daerah putih,
daerah ini akan
 %dikalikan dengan gambar asli. Hitam "lubang" akan
ditambahkan oleh model
 %wajah
   invbw = xor(mfitbw,ones(size(mfitbw)));

 %Untuk mengalikan gambar diatas dengan yang asli
   source_with_hole = uint8(double(invbw) .*
double(imsourcegray));

%Untuk  menambah putar wajah model.
%Gambar ini mengandung bagian gambar asli yang tidak menghadap
dan wajah
%model ditambahkan
    final_image = uint8(double(source_with_hole) +
double(mfit));

   figure,imshow(final_image),title('Hasil Template
Matching');

   imsourcegray = final_image;
 end;
```

Orient

```
%Menghitung sudut kemiringan wilayah tersegmentasi
%menurut koordinat (xmean,ymean)dan tersegmentasi daerah itu
sendiri
function [theta] = orient(bw,xmean,ymean)
[m n] =size(bw);
bw=double(bw);

a = 0; b = 0; c = 0;
for i=1:m,
  for j=1:n,
     a = a + (j - xmean)^2 * bw(i,j);
     b = b + (j - xmean) * (i - ymean) * bw(i,j);
     c = c + (i - ymean)^2 * bw(i,j);
  end;
end;
b = 2 * b;

theta = atan(b/(a-c))/2;
```

```
%Perubahan dari radian ke derajat
theta = theta*(180/pi);
```

FaceInfo

```
%Bagian ini menjelaskan beberapa informasi wilayah yang
mungkin menunjukkan wajah.
%Fungsi ini mengambil :
%1.Nilai korelasi silang antara daerah wajah dan frontal
%2.Model diposisikan pada daerah wajah saja
%3.Rectangel koordinat wajah yang mungkin menunjukkan wajah

function [ccorr, mfit, RectCoord] = faceInfo(mult,
frontalmodel,ly,wx,cx, cy, angle)

%Resize model frontal sesuai dengan tinggi dan lebar daerah
  model_rot = imresize(frontalmodel,[ly wx],'bilinear');

%Model diputar
  model_rot = imrotate(model_rot,-angle,'bilinear');

%Untuk mendapatkkan koordinat putar gambar
  [l,r,u,d] = recsize(model_rot);

%Untuk menghasilkan gambar baru yang hanya memiliki wilayah
model
  model_rot=imcrop(model_rot,[l u (r-l) (d-u)]);

%Menyingkirkan batas kabisingan
  model_rot = clean_model(model_rot);
  bwmodel_rot = (model_rot >=1);

%Menghitung nilai tengah putar model wajah
  [modx,mody] =center(bwmodel_rot);

%Untuk mendapatkan ukuran rotasi model wajah
  [morig, norig] = size(bwmodel_rot);

%Membuat gambar grayscale yang akan memiliki model wajah
diatasnya
  mfit = zeros(size(mult));
  mfitbw = zeros(size(mult));
  [limy, limx] = size(mfit);

%Untuk menghitung koordinat mana model wajah akan berada di
citra utama
  startx = cx-modx;
  starty = cy-mody;
  endx = startx + norig-1;
  endy = starty + morig-1;

%Untuk periksa batas gambar
  startx = checklimit(startx,limx);
  starty = checklimit(starty,limy);
```

```
  endx = checklimit(endx,limx);
  endy = checklimit(endy,limy);

%Perintah untuk menghasilkan gambar baru yang sama memilliki
ukuran seperti
%yang asli, tetapi dengan wajah model diatasnya sesuai rotasi
  for i=starty:endy,
   for j=startx:endx,
     mfit(i,j) = model_rot(i-starty+1,j-startx+1);
    end;
  end;

%Untuk menampilan masing2 posisi template di gambar:
 % figure,imshow(mfit,[0 255]),title('Tampilan Posisi
Template');

%Untuk mendapatkan nilai cross-korelasi antara model wajah dan
daerah yang
%mungkin menunjukkan wajah dalam gambar
ccorr = corr2(mfit,mult)

%Untuk mendapatkan koordinat persegi panjang yang akan
menunjukkan posisi
%wajah dalam gambar
[l,r,u,d] = recsize(bwmodel_rot);

%Menghitung titik awal dalam x dan y, serta lebar persegi
panjang
sx = startx+l;
sy = starty+u;

%Untuk membuat koordinat persegi panjang dan kembali
RectCoord = [sx  sy (r-1) (d-u)];

%Memverifikasi koordinat antara daerah gambar
function newcoord = checklimit(coord,maxval)

  newcoord = coord;
  if (newcoord<1) newcoord=1; end;
  if (newcoord>maxval) newcoord=maxval; end;
```

Clean Model

```
%Membersihkan tepi gambar setelah diputar
function model = clean_model(model)

[m n] = size(model);

%Membersikan dari kiri ke kanan
 for i=1:m,
    nfound = 0;
    for j=1:n,
      if (model(i,j) < 95 & nfound == 0) model(i,j) = 0; end;
      if (model(i,j) >=95 & nfound == 0) nfound=1; end;
```

```
    end;
 end;

 %Membersikan dari kanan ke kiri
 for i=1:m,
   nfound = 0;
   for j=n:-1:1,
      if (model(i,j) < 80 & nfound == 0) model(i,j) = 0; end;
      if (model(i,j) >=80 & nfound == 0) nfound=1; end;
   end;
 end;
```

Center

```
%Menghitung pusat massa dari daerah kulit berdasarkan
wilayahnya
function [xmean, ymean] = center(bw)
bw=bwfill(bw,'holes');

area = bwarea(bw);
[m n] =size(bw);
bw=double(bw);

xmean =0; ymean = 0;
for i=1:m,
  for j=1:n,
     xmean = xmean + j*bw(i,j);
     ymean = ymean + i*bw(i,j);
  end;
end;

xmean = xmean/area;
ymean = ymean/area;

xmean = round(xmean);
ymean = round(ymean);
```

Segmentasi

Convolve

```
/* konvolusi */

#ifndef CONVOLVE_H
#define CONVOLVE_H

#include <vector>
#include <algorithm>
#include <cmath>
#include <math.h>
#include "image.h"

/* konvolusi src dengan mask dan dst mengembalikan */
static void convolve_even(image<float> *src, image<float>
*dst,
std::vector<float> &mask) {
int width = src->width();
int height = src->height();
int len = mask.size();
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
float sum = mask[0] * imRef(src, x, y);
for (int i = 1; i < len; i++) {
sum += mask[i] *
(imRef(src, max(x-i,0), y) +
imRef(src, min(x+i, width-1), y));
```

```
}
imRef(dst, y, x) = sum;
}
}
}
/* konvolusi src dengan mask dan dst mengembalikan */
static void convolve_odd(image<float> *src, image<float>
*dst,
std::vector<float> &mask) {
int width = src->width();
int height = src->height();
int len = mask.size();
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
float sum = mask[0] * imRef(src, x, y);
for (int i = 1; i < len; i++) {
sum += mask[i] *
(imRef(src, max(x-i,0), y) -
imRef(src, min(x+i, width-1), y));
}
imRef(dst, y, x) = sum;
}
}
}
#endif
```

Disjoint_set

```
#ifndef DISJOINT_SET
```

```
#define  DISJOINT_SET

typedef  struct  {
int  rank;
int  p;
int  size;
}  uni_elt;
class  universe  {
public:
universe(int  elements);
~universe();
int  find(int  x);
void  join(int  x,  int  y);
int  size(int  x)  const  {  return  elts[x].size;  }
int  num_sets()  const  {  return  num;  }
private:
uni_elt  *elts;
int  num;
};
universe::universe(int  elements)  {
elts  =  new  uni_elt[elements];
num  =  elements;
for  (int  i  =  0;  i  <  elements;  i++)  {
elts[i].rank  =  0;
elts[i].size  =  1;
elts[i].p  =  i;
}
```

```
}
universe::~universe()  {
delete  [] elts;
}
int  universe::find(int  x)  {
int  y  =  x;
while  (y  !=  elts[y].p)
y  =  elts[y].p;
elts[x].p  =  y;
return  y;
}
void  universe::join(int  x,  int  y)  {
if (elts[x].rank  >  elts[y].rank)  {
elts[y].p  =  x;
elts[x].size  +=  elts[y].size;
}  else  {
elts[x].p  =  y;
elts[y].size  +=  elts[x].size;
if (elts[x].rank  ==  elts[y].rank)
elts[y].rank++;
}
num--;
}
#endif
```

Filter

```
/*  filter  sederhana  */
```

```
#ifndef FILTER_H
#define FILTER_H

#include <vector>
#include <cmath>
#include "image.h"
#include "misc.h"
#include "convolve.h"
#include "imconv.h"

#define WIDTH 4.0

/* normalkan mask sehingga akan menggabungkannya jadi
satu */
static void normalize(std::vector<float> &mask) {
int len = mask.size();
float sum = 0;
for (int i = 1; i < len; i++) {
sum += fabs(mask[i]);
}
sum = 2*sum + fabs(mask[0]);
for (i = 0; i < len; i++) {
mask[i] /= sum;
}
}
/* buat filter */
#define MAKE_FILTER(name, fun)
```

```
static std::vector<float> make_ ## name (float sigma)
{
sigma = max(sigma, 0.01F);
int len = (int)ceil(sigma * WIDTH) + 1;
std::vector<float> mask(len);
for (int i = 0; i < len; i++) {
mask[i] = fun;
}
return mask;
}
MAKE_FILTER(fgauss, exp(-0.5*square(i/sigma)));
/* convolve image with gaussian filter */
static image<float> *smooth(image<float> *src, float
sigma) {
std::vector<float> mask = make_fgauss(sigma);
normalize(mask);
image<float> *tmp = new image<float>(src->height(), src-
>width(), false);
image<float> *dst = new image<float>(src->width(), src-
>height(),
false);
convolve_even(src, tmp, mask);
convolve_even(tmp, dst, mask);
delete tmp;
return dst;
}
/* konvolusi citra dengan filter gaussian */
image<float> *smooth(image<uchar> *src, float sigma) {
image<float> *tmp = imageUCHARtoFLOAT(src);
```

```
image<float> *dst = smooth(tmp, sigma);
delete tmp;
return dst;
}
/* hitung laplacian */
static image<float> *laplacian(image<float> *src) {
int width = src->width();
int height = src->height();
image<float> *dst = new image<float>(width, height);
for (int y = 1; y < height-1; y++) {
for (int x = 1; x < width-1; x++) {
float d2x = imRef(src, x-1, y) + imRef(src, x+1, y)
-
2*imRef(src, x, y);
float d2y = imRef(src, x, y-1) + imRef(src, x, y+1)
-
2*imRef(src, x, y);
imRef(dst, x, y) = d2x + d2y;
}
}
return dst;
}
#endif
```

Image

```
/* pengkelasan citra */

#ifndef IMAGE_H
```

```
#define IMAGE_H

#include <cstring>

template <class T>
class image {
public:
/* membangun citra */
image(const int width, const int height, const bool
init = true);
/* menghapus citra */
~image();
/* init citra */
void init(const T &val);
/* mengcopy citra */
image<T> *copy() const;
/* memperoleh lebar citra */
int width() const { return w; }
/* memperoleh tinggi citra */
int height() const { return h; }

/* data citra */ T *data;

/* pointer */ T **access;
private:
int w, h;
};
/* gunakan imRef untuk mengakses data citra */
```

```
#define imRef(im, x, y) (im->access[y][x])

/* gunakan imPtr untuk memperoleh pointer ke data
citra. */

#define imPtr(im, x, y) &(im->access[y][x])

template <class T>

image<T>::image(const int width, const int height, const
bool init) {

w = width;

h = height;

data = new T[w * h]; // pengalokasian tempat untuk
data citra access = new T*[h]; // pengalokasian tempat
untuk pointer// penginisialan pointer

for (int i = 0; i < h; i++)

access[i] = data + (i * w);

if (init)

memset(data, 0, w * h * sizeof(T));

}

template <class T>

image<T>::~image() {

delete [] data;

delete [] access;

}

template <class T>

void image<T>::init(const T &val) {

T *ptr = imPtr(this, 0, 0);

T *end = imPtr(this, w-1, h-1);

while (ptr <= end)

*ptr++ = val;

}
```

```
template <class T>
image<T> *image<T>::copy() const {
image<T> *im = new image<T>(w, h, false); memcpy(im->data, data, w * h * sizeof(T)); return im;
}
#endif
```

Imcov

```
/* konversi citra */

#ifndef CONV_H
#define CONV_H

#include <climits>
#include "image.h"
#include "imutil.h"
#include "misc.h"

#define RED_WEIGHT 0.299
#define GREEN_WEIGHT 0.587
#define BLUE_WEIGHT 0.114

static image<uchar> *imageRGBtoGRAY(image<rgb> *input) {
int width = input->width();
int height = input->height();
image<uchar> *output = new image<uchar>(width, height, false);
```

```
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
imRef(output, x, y) = (uchar)
(imRef(input, x, y).r * RED_WEIGHT +
imRef(input, x, y).g * GREEN_WEIGHT +
imRef(input, x, y).b * BLUE_WEIGHT);
}
}
return output;
}
static image<rgb> *imageGRAYtoRGB(image<uchar> *input) {
int width = input->width();
int height = input->height();
image<rgb> *output = new image<rgb>(width, height,
false);

for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) { imRef(output, x,
y).r = imRef(input, x, y); imRef(output, x, y).g =
imRef(input, x, y); imRef(output, x, y).b = imRef(input,
x, y);
}
}
return output;
}
static image<float> *imageUCHARtoFLOAT(image<uchar> *input)
{
int width = input->width();
int height = input->height();
```

```
image<float> *output = new image<float>(width, height,
false);
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
imRef(output, x, y) = imRef(input, x, y);
}
}
return output;
}
static image<float> *imageINTtoFLOAT(image<int> *input) {
int width = input->width();
int height = input->height();
image<float> *output = new image<float>(width, height,
false);
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
imRef(output, x, y) = imRef(input, x, y);
}
}
return output;
}
static image<uchar> *imageFLOATtoUCHAR(image<float> *input,
float min, float max) {
int width = input->width();
int height = input->height();
image<uchar> *output = new image<uchar>(width, height,
false);
if (max == min)
return output;
```

```
float scale = UCHAR_MAX / (max - min);
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
uchar val = (uchar)((imRef(input, x, y) - min) *
scale);
imRef(output, x, y) = bound(val, (uchar)0,
(uchar)UCHAR_MAX);
}
}
return output;
}
static image<uchar> *imageFLOATtoUCHAR(image<float> *input)
{
float min, max;
min_max(input, &min, &max);
return imageFLOATtoUCHAR(input, min, max);
}
static image<long> *imageUCHARtoLONG(image<uchar> *input)
{
int width = input->width();
int height = input->height();
image<long> *output = new image<long>(width, height,
false);
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
imRef(output, x, y) = imRef(input, x, y);
}
}
return output;
}
```

```
static image<uchar> *imageLONGtoUCHAR(image<long> *input,
long min, long max) {

int width = input->width();

int height = input->height();

image<uchar> *output = new image<uchar>(width, height,
false);

if (max == min)

return output;

float scale = UCHAR_MAX / (float)(max - min);

for (int y = 0; y < height; y++) {

for (int x = 0; x < width; x++) {

uchar val = (uchar)((imRef(input, x, y) - min) *
scale);

imRef(output, x, y) = bound(val, (uchar)0,
(uchar)UCHAR_MAX);

}

}

return output;

}

static image<uchar> *imageLONGtoUCHAR(image<long> *input)
{

long min, max;

min_max(input, &min, &max);

return imageLONGtoUCHAR(input, min, max);

}

static image<uchar> *imageSHORTtoUCHAR(image<short> *input,
short min, short max) {

int width = input->width();

int height = input->height();

image<uchar> *output = new image<uchar>(width, height,
false);
```

```
if (max == min)
return output;
float scale = UCHAR_MAX / (float)(max - min);
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
uchar val = (uchar)((imRef(input, x, y) - min) *
scale);
imRef(output, x, y) = bound(val, (uchar)0,
(uchar)UCHAR_MAX);
}
}
return output;
}
static image<uchar> *imageSHORTtoUCHAR(image<short> *input)
{
short min, max;
min_max(input, &min, &max);
return imageSHORTtoUCHAR(input, min, max);
}
#endif
```

Imutil

```
/* perlengkapan citra */
#ifndef IMUTIL_H
#define IMUTIL_H
#include "image.h"
#include "misc.h"
/* compute Hitung Nilai Minimum dan Maksimum Citra */
template <class T>
```

```
void min_max(image<T> *im, T *ret_min, T *ret_max) {
int width = im->width();
int height = im->height();
T min = imRef(im, 0, 0); T max = imRef(im, 0, 0);
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) { T val =
imRef(im, x, y);
if (min > val)
min = val;
if (max < val)
max = val;
}
}
*ret_min = min;
*ret_max = max;
}
/* threshold citra */
template <class T>
image<uchar> *threshold(image<T> *src, int t) {
int width = src->width();
int height = src->height();
image<uchar> *dst = new image<uchar>(width, height);
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
imRef(dst, x, y) = (imRef(src, x, y) >= t);
}
}
return dst;
```

```
}
#endif
```

Imsc

```
/*  perangkat  acak */

#ifndef  MISC_H
#define  MISC_H

#include  <cmath>

#ifndef  M_PI
#define  M_PI  3.141592653589793
#endif
typedef  unsigned  char  uchar;
typedef  struct  {  uchar  r, g,  b;  }  rgb;
inline  bool  operator==(const  rgb  &a,  const  rgb  &b)  {
return  ((a.r  ==  b.r)  &&  (a.g  ==  b.g)  &&  (a.b  ==
b.b));
}
template  <class  T>
inline  T  abs(const  T  &x)  {  return  (x  >  0  ? x  : -x);
};
template  <class  T>
inline  int  sign(const  T  &x)  {  return  (x  >=  0  ? 1  :
-1);  };
template  <class  T>
inline  T  square(const  T  &x)  {  return  x*x;  };
template  <class  T>
```

```
inline T bound(const T &x, const T &min, const T
&max) {
return (x < min ? min : (x > max ? max : x));
}
template <class T>
inline bool check_bound(const T &x, const T&min, const
T &max) {
return ((x < min) || (x > max));
}
inline int vlib_round(float x) { return (int)(x +
0.5F); }
inline int vlib_round(double x) { return (int)(x +
0.5); }
inline double gaussian(double val, double sigma) {
return exp(-square(val/sigma)/2)/(sqrt(2*M_PI)*sigma);
}
#endif
```

Pnmfile

```
/* dasar citra Input Output */

#ifndef PNM_FILE_H
#define PNM_FILE_H

#include <cstdlib>
#include <climits>
#include <cstring>
#include <fstream>
#include "image.h"
```

```
#include "misc.h"

#define BUF_SIZE 256 class pnm_error { };

static void read_packed(unsigned char *data, int size,
std::ifstream

&f) {

unsigned char c = 0;

int bitshift = -1;

for (int pos = 0; pos < size; pos++) {

if (bitshift == -1) {

c = f.get();

bitshift = 7;

}

data[pos] = (c >> bitshift) & 1;

bitshift--;

}

}

static void write_packed(unsigned char *data, int size,
std::ofstream

&f) {

unsigned char c = 0;

int bitshift = 7;

for (int pos = 0; pos < size; pos++) { c = c |
(data[pos] << bitshift); bitshift--;

if ((bitshift == -1) || (pos == size-1)) {

f.put(c);

bitshift = 7;

c = 0;

}

}
```

```
}
/* membaca file pnm, mengabaikan komentar */
static void pnm_read(std::ifstream &file, char *buf) {
char doc[BUF_SIZE];
char c;
file >> c;
while (c == '#') {
file.getline(doc, BUF_SIZE);
file >> c;
}
file.putback(c);
file.width(BUF_SIZE); file >> buf; file.ignore();
}
static image<uchar> *loadPBM(const char *name) {
char buf[BUF_SIZE];
/* membaca header */
std::ifstream file(name, std::ios::in | std::ios::binary);
pnm_read(file, buf);
if (strncmp(buf, "P4", 2))
throw pnm_error();
pnm_read(file, buf); int width = atoi(buf); pnm_read(file,
buf);
int height = atoi(buf);
/* membaca data */
image<uchar> *im = new image<uchar>(width, height);
for (int i = 0; i < height; i++)
read_packed(imPtr(im, 0, i), width, file);
return im;
```

```
}
static void savePBM(image<uchar> *im, const char *name)
{
int width = im->width();
int height = im->height();
std::ofstream file(name, std::ios::out | std::ios::binary);
file << "P4\n" << width << " " << height << "\n";
for (int i = 0; i < height; i++)
write_packed(imPtr(im, 0, i), width, file);
}
static image<uchar> *loadPGM(const char *name) {
char buf[BUF_SIZE];
/* membaca header */
std::ifstream file(name, std::ios::in | std::ios::binary);
pnm_read(file, buf);
if (strncmp(buf, "P5", 2))
throw pnm_error();
pnm_read(file, buf); int width = atoi(buf); pnm_read(file,
buf);
int height = atoi(buf);
pnm_read(file, buf);
if (atoi(buf) > UCHAR_MAX)
throw pnm_error();
/* membaca data */
image<uchar> *im = new image<uchar>(width, height);
file.read((char *)imPtr(im, 0, 0), width * height *
sizeof(uchar));
return im;
}
```

```
static void savePGM(image<uchar> *im, const char *name)
{
int width = im->width();
int height = im->height();
std::ofstream file(name, std::ios::out | std::ios::binary);
file << "P5\n" << width << " " << height << "\n" <<
UCHAR_MAX << "\n";
file.write((char *)imPtr(im, 0, 0), width * height *
sizeof(uchar));
}
static image<rgb> *loadPPM(const char *name) {
char buf[BUF_SIZE], doc[BUF_SIZE];
/* read header */
std::ifstream file(name, std::ios::in | std::ios::binary);
pnm_read(file, buf);
if (strncmp(buf, "P6", 2))
throw pnm_error();
pnm_read(file, buf); int width = atoi(buf); pnm_read(file,
buf);
int height = atoi(buf);
pnm_read(file, buf);
if (atoi(buf) > UCHAR_MAX)
throw pnm_error();
/* membaca data */
image<rgb> *im = new image<rgb>(width, height);
file.read((char *)imPtr(im, 0, 0), width * height *
sizeof(rgb));
return im;
}
```

```
static void savePPM(image<rgb> *im, const char *name) {
int width = im->width();
int height = im->height();
std::ofstream file(name, std::ios::out | std::ios::binary);
file << "P6\n" << width << " " << height << "\n" << UCHAR_MAX << "\n";
file.write((char *)imPtr(im, 0, 0), width * height * sizeof(rgb));
}
template <class T>
void load_image(image<T> **im, const char *name) {
char buf[BUF_SIZE];
/* membaca header */
std::ifstream file(name, std::ios::in | std::ios::binary);
pnm_read(file, buf);
if (strncmp(buf, "VLIB", 9))
throw pnm_error();
pnm_read(file, buf); int width = atoi(buf); pnm_read(file, buf);
int height = atoi(buf);
/* membaca data */
*im = new image<T>(width, height);
file.read((char *)imPtr((*im), 0, 0), width * height * sizeof(T));
}
template <class T>
void save_image(image<T> *im, const char *name) {
int width = im->width();
int height = im->height();
```

```
std::ofstream file(name, std::ios::out | std::ios::binary);
file << "VLIB\n" << width << " " << height << "\n";
file.write((char *)imPtr(im, 0, 0), width * height *
sizeof(T));
}
#endif
```

Segment_graph

```
#ifndef SEGMENT_GRAPH
#define SEGMENT_GRAPH

#include <algorithm>
#include <cmath>
#include "disjoint-set.h"

// fungsi threshold
#define THRESHOLD(size, c) (c/size)
typedef struct {
float w;
int a, b;
} edge;
bool operator<(const edge &a, const edge &b) {
return a.w < b.w;
}
/*
* menggolongkan graph
*
```

```
* mengembalikan gambaran segmentasi dari himpunan
disjoint dari

* forest

*

* num_vertices: banyaknya simpul di graph.

* num_edges: banyaknya sisi di graph

* edges: array dari sisi.

* c: konstanta untuk fungsi threshold.

*/

universe *segment_graph(int num_vertices, int num_edges,
edge *edges, float c) {

// susun sisi berdasarkan bobot std::sort(edges, edges +
num_edges);

// buat himpunan disjoint dari forest universe *u =
new universe(num_vertices);

// init threshold

float *threshold = new float[num_vertices];

for (int i = 0; i < num_vertices; i++)

threshold[i] = THRESHOLD(1,c);

// Untuk setiap sisi dengan bobot tidak menurun, lakukan
ini... for (i = 0; i < num_edges; i++) {

 edge *pedge = &edges[i];

// Komponen yang terhubung dari sisi int a = u-
>find(pedge->a);

int b = u->find(pedge->b);

if (a != b) {

if ((pedge->w <= threshold[a]) && (pedge->w <=
threshold[b])) {

u->join(a, b);

a = u->find(a);

threshold[a] = pedge->w + THRESHOLD(u->size(a), c);
```

```
}
}
}
delete  threshold;
return  u;
}
#endif
```

Segment_image

```
#ifndef  SEGMENT_IMAGE
#define  SEGMENT_IMAGE

#include  <cstdlib>
#include  "image.h"
#include  "misc.h"
#include  "filter.h"
#include  "segment-graph.h"
// pewarnaan  acak rgb  random_rgb(){
rgb  c;
double  r;
c.r = (uchar)rand(); c.g = (uchar)rand(); c.b  =
(uchar)rand();
return  c;
}
// Ukuran  ketidaksamaan  antara  piksel
static  inline  float  diff(image<float>  *r,  image<float>
*g,
image<float>  *b,
```

```
int x1, int y1, int x2, int y2) {
return sqrt(square(imRef(r, x1, y1)-imRef(r, x2, y2)) +
square(imRef(g, x1, y1)-imRef(g, x2, y2)) +
square(imRef(b, x1, y1)-imRef(b, x2, y2)));
}
/*
* menggolongkan citra
*
* mengembalikan gambaran segmentasi dari warna citra.
*
* im: citra yang akan digolongkan.
* sigma: untuk memperhalus citra.
* c: konstanta untuk fungsi threshold.
* min_size: ukuran minimum komponen.
* num_ccs: banyaknya komponen terhubung dalam
segmentasi.
*/
image<rgb> *segment_image(image<rgb> *im, float sigma,
float c, int min_size,
int *num_ccs) {
int width = im->width();
int height = im->height();
image<float> *r = new image<float>(width, height);
image<float> *g = new image<float>(width, height);
image<float> *b = new image<float>(width, height);
// smooth each color channel
for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) { imRef(r, x, y) =
imRef(im, x, y).r; imRef(g, x, y) = imRef(im, x, y).g;
imRef(b, x, y) = imRef(im, x, y).b;
```

```
}
}
image<float> *smooth_r = smooth(r, sigma);
image<float> *smooth_g = smooth(g, sigma);
image<float> *smooth_b = smooth(b, sigma);
delete r; delete g; delete b;
// bangun graph
edge *edges = new edge[width*height*4];
int num = 0;
for (y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
if (x < width-1) { edges[num].a = y * width + x;
edges[num].b = y * width + (x+1);
edges[num].w = diff(smooth_r, smooth_g, smooth_b, x, y,
x+1, y);
num++;
}
if (y < height-1) { edges[num].a = y * width + x;
edges[num].b = (y+1) * width + x;
edges[num].w = diff(smooth_r, smooth_g, smooth_b, x, y,
x, y+1);
num++;
}
if ((x < width-1) && (y < height-1)) { edges[num].a =
y * width + x; edges[num].b = (y+1) * width + (x+1);
edges[num].w = diff(smooth_r, smooth_g, smooth_b, x, y,
x+1,
y+1);
num++;
}
```

```
1);

if ((x < width-1) && (y > 0)) { edges[num].a = y *
width + x; edges[num].b = (y-1) * width + (x+1);

edges[num].w = diff(smooth_r, smooth_g, smooth_b, x, y,
x+1, y-

num++;

}

}

}

delete smooth_r;

delete smooth_g;

delete smooth_b;

// segment

universe *u = segment_graph(width*height, num, edges,
c);

// pemrosesan komponen yang kecil for (int i = 0; i <
num; i++) {

int a = u->find(edges[i].a);

int b = u->find(edges[i].b);

if ((a != b) && ((u->size(a) < min_size) || (u->size(b)
<

min_size)))

u->join(a, b);

}

delete [] edges;

*num_ccs = u->num_sets();

image<rgb> *output = new image<rgb>(width, height);

// mengambil warna acak untk setiap komponen rgb *colors
= new rgb[width*height];

for (i = 0; i < width*height; i++)
```

```
colors[i] = random_rgb();
for (y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
int comp = u->find(y * width + x);
imRef(output, x, y) = colors[comp];
}
}
delete [] colors;
delete u;
return output;
}

#endif
```