PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENGENALAN PENCAHAYAAN INVARIAN DARI OBYEK TIGA DIMENSI MENGGUNAKAN WARNA INVARIAN LOKAL

TUGAS AKHIR

Disusun Oleh :

ANDRI DARMAWAN

NRP. 2693 100 046

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2000

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENGENALAN PENCAHAYAAN INVARIAN DARI OBYEK TIGA DIMENSI MENGGUNAKAN WARNA INVARIAN LOKAL

TUGAS AKHIR

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

Mengetahui / Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Esther Hanaya, M.Sc.	Rully Soelaiman, S.Kom.
NIP. 130 816 212	NIP. 132 085 802

SURABAYA
Agustus, 2000

Ini adalah anugerah Tuhanku untuk mencoba aku, apakah aku bersyukur atau kufur (Ali Imron 37)

Maka nikmat Tuhan kamu yang manakah yang kamu dustakan?
(Ar Rahman 13)

ABSTRAK

ABSTRAK

Warna sebuah obyek yang ditangkap oleh mata manusia merupakan warna pantul permukaan obyek terhadap pencahayaan yang mengenainya bukan warna permukaan asli obyek tersebut. Sehingga sebuah obyek yang telah dikenai oleh pencahayaan yang berubah-ubah, akan tetap dikenali sebagai obyek yang sama.

Untuk mengenali sebuah obyek berdasarkan pencahayaan yang mengenainya, dalam tugas akhir ini digunakan metode *color constant color indexing*. Metode ini terbagi atas beberapa tahap, yakni memisahkan warna obyek dengan pencahayaan yang mengenainya, memisahkan warna obyek dengan warna latar belakangnya dan membuat histogram rasio warnanya. Sehingga dalam proses pengenalan sebuah obyek terhadap obyek lainnya, dilakukan proses irisan histogram rasio warna keduanya untuk memperoleh tingkat akurasi kesamaan kedua obyek tersebut.

Dari beberapa hasil uji coba perangkat lunak yang telah dilakukan, terlihat bahwa kemampuan pengenalan sebuah obyek tidak dipengaruhi oleh pencahayaan yang mengenainya, sepanjang pencahayaan tersebut tidak terlalu terang ataupun terlalu gelap.

KATA PENGANTAR

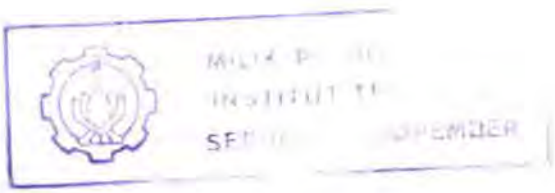
KATA PENGANTAR

Alhamdulillahirrobbil'alamiin, hanya kepada-Mu ya Allah yang berhak di puji dan disembah di seluruh alam semesta ini. Karena atas hidayah-Mu akhirnya penulis dapat menyelesaikan pembuatan tugas akhir ini. Buku ini disusun untuk melengkapi pembuatan tugas akhir yang berjudul :

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENGENALAN PENCAHAYAAN INVARIAN DARI OBYEK TIGA DIMENSI MENGGUNAKAN WARNA INVARIAN LOKAL.

Dalam penyusunan buku ini, penulis berusaha untuk membahas permasalahan sedetail dan seurut mungkin, mulai dari dasar teori sampai implementasi dalam bentuk perangkat lunak. Oleh karena itu untuk memperoleh pembahasan yang tepat dari tugas akhir ini, sebaiknya buku ini dibaca secara berurutan mulai dari awal sampai akhir.

Tugas akhir ini dibuat selama beberapa bulan dan dalam jangka waktu itu penulis mengalami masa-masa jenuh dan beberapa kesulitan. Tetapi akhirnya masa-masa tersebut dapat terlewati berkat adanya dorongan, dukungan, bantuan dan do'a langsung maupun tidak langsung dari berbagai pihak. Oleh karena itu perkenankan penulis mengucapkan terima kasih dengan sepenuh hati dan mendo'akan semoga Allah SWT mencatat sebagai amal sholeh, kepada :



1. Abah Hadi Parno dan Umi Winarni, kedua adik-ku tersayang, Irma Indriani dan Choirul Andriyanto .
2. Ir. Esther Hanaya, MSc dan Rully Soelaeman, S.Kom yang memberi bantuan dan bimbingan selama pembuatan tugas akhir ini.
3. Seluruh Dosen Teknik Informatika-ITS yang telah memberi banyak ilmu, kesempatan, kemudahan (dan kesulitan:☺) selama penulis kuliah disini.
4. Karyawan Teknik Informatika-ITS (Mas Yudi, Pak 'In, cak Pri, cak Sholeh, cak Qodir) termasuk pak Satpam.
5. Farida dan Isna Faiza yang sempat menemani penulis selama ini.
6. Spesial buat yang bantu secara langsung selama pembuatan tugas akhir kepadaku : M. Aziz (yang ngasih password), Ilham Riyanto (Master of Programming), Agung 'Doyi' Heriawan (yang mau diajak 'ribut'), Hamsyi (atas ulasan-nya), M. Ndaru Purnomo Sidi (atas wawasan-nya), Achmad 'Macho' Husin, Roy Setya Eka, Akhmad Syaiful dan Rachmad Tri Wibowo.
7. Spesial buat yang bantu dan menemani penulis selama kuliah disini : C-09 seluruhnya tanpa terkecuali, Rifa'i, Jamroni, Hudan, Asfik, Dini, Na'im, dan Susan.
8. Spesial buat yang bantu penulis di luar sana : Irfani, Anthony Hamzah, Nur Rozzi, Budi, Mila, Indri, Sisca dan Luca.
9. Beberapa komunitas yang ikut mendampingi penulis selama ini : Asrama ITS Blok-D, GR-06, GL-29, Halaqoh Madani dan Mas Djunaedy, Jama'ah Tabligh, LAB-man old 'n new version (atas 'kebersamaan'-nya), teman-teman seperjuangan para TA-wan Teknik Informatika September 2000,

10. Beberapa Entertainment yang membantu 'menyegarkan' kepenatan penulis: Panitia Euro 2000 dan Ligina VI (meskipun Italia dan Arema kalah), buku-buku karangan Jalaluddin Rakhmad, Emha 'Ainun Najib, Alwi Shihab dan Hermawan Kertajaya, berita-berita terbaru dari detik.com.
11. Beberapa pihak yang pasti ikut membantu penulis selama ini, yang mungkin secara tidak sengaja dan lupa tidak disebutkan penulis di atas.

Sebagai penutup dari kata pengantar ini, penulis hanya bisa berdo'a : "Ya Allah tuhanku, berikanlah kepada kami kebaikan di dunia dan di akhirat, dan jadikanlah kami termasuk golongan orang-orang yang selalu bersyukur atas segala nikmat yang telah Engkau berikan. *Amin Ya Robbal'alamiin*"

DAFTAR ISI

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR GAMBAR

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Dewasa ini teknologi komputer berkembang dengan pesat, baik dari segi perangkat keras (*hardware*) maupun perangkat lunak (*software*). Komputer juga telah merambah berbagai segi kehidupan manusia, mulai dari keperluan pribadi sampai keperluan bisnis di perusahaan-perusahaan besar. Salah satu bentuk perkembangan yang ada dalam bidang perangkat lunak berupa teknik pengenalan sebuah obyek, meskipun bentuk pengenalan lebih khusus banyak juga dikembangkan. Misalnya pengenalan wajah, sidik jari, tanda tangan dan sebagainya.

Pada pengenalan obyek, metode pendekatannya dapat dikelompokkan berdasarkan informasi yang digunakan untuk proses pengenalannya, misalnya warna, tekstur, bentuk obyek dan lain-lain. Tugas akhir ini, hanya difokuskan untuk membahas salah satu dari pendekatan pengenalan yang ada. Pendekatan yang dimaksudkan adalah lewat pengolahan informasi warna yang dipunyai oleh sebuah obyek.

Warna permukaan sebuah obyek yang diterima oleh mata manusia dipengaruhi oleh beberapa hal. Salah satunya dipengaruhi oleh pencahayaan yang mengenai permukaan obyek. Pengaruh ini hanya berakibat pada warna

yang diterima oleh mata manusia, sedangkan warna asli permukaan obyek masih tetap. Sehingga sebuah obyek akan tetap terkenali sebagai obyek yang sama meskipun telah terjadi perubahan pencahayaan yang mengenainya.

Salah satu metode yang bertujuan untuk mengenali sebuah obyek menggunakan warna yakni metode *color constant color indexing*. Metode ini didasarkan pada pengindeksan rasio warna sebuah obyek ke dalam sebuah histogram. Sehingga dalam proses pengenalan sebuah obyek terhadap obyek lainnya, dilakukan proses irisan histogram rasio warna keduanya untuk memperoleh tingkat akurasi kesamaan sebuah obyek terhadap obyek lainnya.

1.2 TUJUAN

Tujuan tugas akhir ini adalah merancang dan membuat sebuah perangkat lunak untuk pengenalan sebuah obyek yang berdasarkan pada pencahayaan yang mengenainya dengan menggunakan informasi warna yang dimiliki oleh obyek tersebut.

1.3 PERMASALAHAN

Untuk dapat menghasilkan perangkat lunak dengan tujuan yang disebutkan di atas maka secara garis besar permasalahan yang harus dipecahkan sebagai berikut :

- Pengenalan obyek berdasarkan pencahayaan yang mengenainya.
- Pembuatan database model obyek citra

1.4 BATASAN MASALAH

Permasalahan yang berhubungan dengan perangkat lunak untuk pengenalan obyek ini sangatlah rumit dan kompleks serta membutuhkan waktu proses yang tidak sedikit, sehingga dalam tugas akhir ini diberikan beberapa batasan masalah, yaitu :

- Citra gambar yang digunakan berformat bitmap (*bmp*) dengan ukuran 150x140 piksel (*true color*). Baik pada citra yang digunakan sebagai model maupun sebagai citra yang dicari obyek di dalamnya
- Obyek yang terdapat pada setiap citra hanyalah satu obyek dengan latar belakangnya berwarna hitam
- Obyek tidak mengalami perubahan skala
- Data histogram dari citra-citra yang dijadikan model disimpan dalam bentuk *file*. Hal ini berhubungan dengan kemudahan dalam pengaksesan dan besarnya *resource* komputer yang dipakai selama proses.

1.5 METODOLOGI

Langkah-langkah yang dilakukan untuk pembuatan tugas akhir ini dibagi menjadi beberapa tahapan, sebagai berikut :

- Studi literatur

  Pada tahap ini dilakukan studi literatur terhadap teori pengolahan citra digital, khususnya pada materi pengenalan obyek

- Menetapkan definisi kebutuhan

  Pada tahap ini dilakukan pendefinisian terhadap kemampuan perangkat lunak yang akan dirancang dan batasan-batasannya.

- Perancangan perangkat lunak

  Pada tahap ini dilakukan perancangan struktur data, algoritma dan diagram alir yang akan digunakan dalam program

- Pembuatan perangkat lunak

  Pada tahap ini dilakukan pengimplementasian struktur data dan algoritma yang telah dirancang ke dalam bahasa pemrograman.

- Evaluasi dan revisi

  Pada tahap ini dilakukan pengevaluasian dan perbaikan program yang telah dibuat, sampai didapatkan hasil yang sesuai atau tidak menyimpang terlalu jauh dari definisi kebutuhan

- Penyusunan naskah tugas akhir

  Pada tahap ini dilakukan penulisan naskah, dan di dalamnya menjelaskan dasar teori yang dipergunakan serta penyusunan laporan

1.6 SISTEMATIKA PEMBAHASAN

Pembahasan mengenai perangkat lunak yang disusun dalam tugas akhir ini dibagi dalam beberapa bab, yang dijelaskan di bawah ini sebagai berikut :

Bab I, **Pendahuluan,** yang menjelaskan latar belakang pembuatan perangkat lunak, tujuan, permasalahan, batasan masalah, langkah-langkah pembuatan tugas akhir ini, dan sistematika penulisan buku tugas akhir

Bab II, **Pengolahan Citra Digital**, yang menjelaskan mengenai dasar-dasar pengolahan sebuah citra yang tentunya berhubungan dengan tujuan dari pembuatan tugas akhir ini

Bab III, **Color Constancy,** yang menjelaskan konsep dasar yang menunjang pada metode yang dipakai, dengan disertai permasalahan yang ada pada konsep ini.

Bab IV, **Color Constant Color Indexing**, yang menjelaskan mengenai metode yang dipakai dalam proses pengenalan obyek.

Bab V, **Perancangan dan Pembuatan Perangkat Lunak,** yang menjelaskan perancangan perangkat lunak dalam pembuatan tugas akhir ini yang meliputi deskripsi perangkat lunak, perancangan data, perancangan proses, serta implementasi dari perangkat lunak.

Bab VI, **Hasil dan Evaluasi**, yang menjelaskan hasil uji coba perangkat lunak untuk pengenalan obyek serta mengevaluasi hasil dari metode yang telah digunakan.

Bab VII, **Kesimpulan dan Saran**, merupakan kesimpulan dari pembahasan bab-bab sebelumnya, hasil evaluasi serta beberapa saran pengembangan untuk penelitian selanjutnya

BAB II

PENGOLAHAN CITRA DIGITAL

BAB II

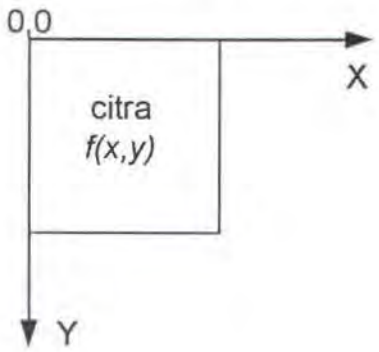
PENGOLAHAN CITRA DIGITAL

Pembahasan pada bab ini menguraikan beberapa konsep dasar yang digunakan untuk menunjang tugas akhir ini. Pada subbab-subbab berikut dijelaskan tentang teori-teori pengolahan citra yang berhubungan dengan metode yang dipakai penulis dalam pembuatan perangkat lunak.

2.1 Model Citra Sederhana

Istilah citra monokrom atau citra sederhana menunjuk kepada fungsi dua dimensi *f(x,y)* dimana *x* dan *y* menyatakan indeks koordinat spasial (atau koordinat kartesian) dan nilai dari *f* pada setiap titik *(x,y)* menyatakan tingkat kecerahan (atau *gray level*) dari citra pada titik tersebut[1]. Untuk dapat diproses oleh komputer, fungsi citra *f(x,y)* harus melalui proses digitalisasi atau didiskritkan, baik koordinat spasial ataupun tingkat kecerahannya. Digitalisasi terhadap koordinat spasial *(x,y)* disebut dengan citra *sampling*, sedangkan digitalisasi terhadap tingkat kecerahan atau amplitudo disebut sebagai *gray level quantization*. Apabila variabel *x* dan *y* kontinyu, citra tersebut adalah citra analog dan merupakan fungsi kontinyu, apabila *x* dan *y* dilakukan proses *sampling* maka fungsi akan menjadi fungsi diskrit. Istilahnya

kontinyu di sini menjelaskan bahwa indeks *x, y* dapat bernilai real, sedangkan bersifat diskrit menjelaskan bahwa indeks *x, y* hanya bernilai integer.



Gambar 1. Citra yang didiskritkan dengan koordinat kartesius

Citra yang kita lihat atau kita amati dalam keseharian penglihatan kita merupakan pemantulan cahaya dari obyek. Pada dasamya *f(x,y)* dapat dikarakteristikkan menjadi dua komponen. Satu komponen merupakan jumlah atau intensitas cahaya yang dikenakan pada obyek (*illumination*), sementara komponen yang lain merupakan jumlah cahaya yang dipantulkan oleh obyek tersebut (*reflectance*). Keduanya dituliskan dengan fungsi yang berturut-turut *i(x,y)* dan *r(x,y)*. Fungsi *i(x,y)* dan *r(x,y)* merupakan kombinasi perkalian untuk membentuk fungsi *f(x,y)* yang dapat dituliskan dengan persamaan :

$$f(x,y) = i(x,y)\, r(x,y) \quad dengan$$
$$0 < i(x,y) < \infty \quad dan \qquad (2.1)$$
$$0 < r(x,y) < 1$$

[1] Rafael C. Gonzales, Richard E. Woods, *Digital Image Processing,* Second Edition, Addison Wesley Publishing Co., Tennesee, 1987, halaman 6

Persamaan di atas menandakan bahwa nilai refleksi dibatasi oleh nilai 0 (absorbsi total) dan nilai 1 (refleksi total). Fungsi *i(x,y)* ditentukan oleh sumber atau asal sinar sedangkan fungsi *r(x,y)* ditentukan oleh karakteristik dari obyek.

Kita dapat menganggap citra kontinyu *f(x,y)* sebagai matriks dengan ukuran *NxM* seperti ditunjukkan pada persamaan di bawah ini, dimana setiap elemen dari matriks merupakan nilai diskrit.

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & \cdots & f(1,M-1) \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \qquad (2.2)$$

Citra yang tidak berwarna (hitam putih) dikenal juga sebagai citra dengan derajat keabuan tertentu (*grey-level*). Derajat keabuan yang dimiliki ini bisa beragam mulai dari 2 derajat keabuan (yaitu 0 dan 1) yang dikenal juga sebagai citra monokrom, 16 derajat keabuan dan 256 derajat keabuan, semakin besar jumlah derajat keabuan yang dimiliki semakin halus citra tersebut.

Dalam sebuah citra monokrom, sebuah piksel diwakili oleh 1 bit data yang berisikan data tentang derajat keabuan yang dimiliki piksel tersebut. Data akan berisi 1 bila piksel tersebut berwarna putih dan data akan berisi nilai 0 bila piksel tersebut berwarna hitam.

Citra yang memiliki 16 derajat keabuan (mulai dari 0 yang mewakili warna hitam sampai dengan 15 yang mewakili warna putih) direpresentasikan oleh 4 bit data. Sedangkan citra dengan 256 derajat keabuan (mulai dari 0 yang mewakili

warna hitam sampai dengan 255 yang mewakili warna putih) direpresentasikan oleh 8 bit data.

Dalam citra berwarna banyaknya jumlah warna bisa beragam mulai dari 16, 256, 65536 atau 16 juta warna, yang masing-masing direpresentasikan oleh 4, 8, 16, atau 24 bit data untuk setiap pikselnya. Warna yang ada terdiri dari 3 komponen utama yaitu merah (*red*), hijau (*green*) dan nilai biru (*blue*). Paduan dari ketiga komponen utama ini dikenal sebagai *RGB Color.*

2.2 Notasi antar Piksel[2]

Hubungan antar piksel pada citra digital merupakan hal yang sangat penting. Seperti sebelumnya, sebuah gambar akan dinotasikan dengan *f(x,y),* sedangkan piksel tertentu dinotasikan dengan huruf kecil seperti *p* dan *q,* dan subset dari himpunan piksel akan dinotasikan dengan huruf besar yang tebal, seperti **S** dan **V**.

2.2.1 Tetangga suatu piksel

Suatu piksel *p* pada koordinat *(x,y)* mempunyai 4 tetangga pada arah horisontal dan vertikal, dengan koordinat

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$

Himpunan dari piksel-piksel ini disebut 4-tetangga dari *p,* yang dinotasikan dengan $N_4(p)$.

---

[2] ibid, halaman 40 -42

Piksel-piksel yang berada pada diagonal dari *p* juga ada 4, dan kita sebut dengan diagonal tetangga, yang mempunyai koordinat

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$

Dan dinotasikan dengan $N_D(P)$. Apabila piksel 4 tetangga dan diagonal tetangga digabungkan, maka kita sebut dengan 8 tetangga dan *p*, dinotasikan dengan $N_8(p)$.

Perlu dicatat, bahwa beberapa piksel dari $N_4(p)$, $N_D(p)$, dan $N_8(p)$ akan berada di luar gambar jika $p(x,y)$ berada pada batas dari gambar.

2.2.2 Keterhubungan (*connectivity*)

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

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

Ada tiga jenis keterhubungan, yaitu :

(a). 4-keterhubungan

Dua piksel *p* dan *q* dengan nilai intensitas ada di **S** termasuk jenis keterhubungan ini jika *q* berada di set $N_4(p)$

(b). 8-keterhubungan

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

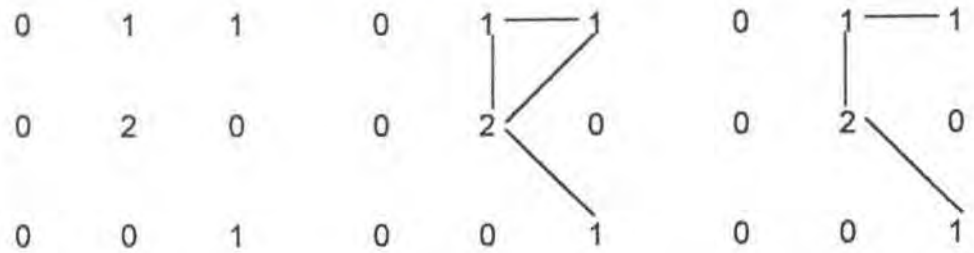
(c). c-keterhubungan (campuran).

Dua piksel p dan q dengan nilai intensitas ada di **S** termasuk jenis keterhubungan ini, jika :

(i). q berada di $N_4(p)$, atau

(ii) q berada di $N_D(p)$ dan set $N_4(p) \cap N_4(q)$ adalah kosong.

Keterhubungan campuran merupakan hasil modifikasi dari 8-keterhubungan, dan ini diperkenalkan untuk menghilangkan jalur keterhubungan ganda (*multiple path connection*) yang sering terjadi jika menggunakan 8-keterhubungan.



Gambar 2. Keterhubungan antar piksel

a. *Susunan piksel*

b. *8-keterhubungan dari intensitas 2*

c. *Keterhubungan-campuran untuk piksel yang sama*

Sebagai contoh, susunan piksel tersusun dengan bentuk seperti Gambar 2(a). Diasumsikan S={1,2}, dan jalur 8-tetangga dari piksel dengan nilai intensitas 2

adalah Gambar 2(b). Ambiguitas dalam jalur hubungan yang dihasilkan 8-tetangga dihilangkan dengan m-keterhubungan, seperti ditunjukkan pada Gambar 2(c).

Piksel *p* disebut berdekatan/bersebelahan (*adjacent*) dengan piksel *q* jika mereka saling terhubung. Kita mungkin mendefiniskan 4-,8-, atau c-, kedekatan dari piksel, tergantung pada jenis keterhubungan yang ditentukan. Dua sub himpunan S_1 dan S_2 dari suatu gambar adalah berdekatan, jika ada beberapa piksel pada S_1 yang berdekatan dengan beberapa piksel dari S_2.

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

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

dimana $(x_0, y_0) = (x, y)$ dan $(x_m, y_n) = (s, f)$, (x_1, y_1) bersebelahan dengan (x_{i-1}, y_{i-1}), $1 \leq i \leq n$, dan *n* adalah panjang dari jalur. Kita bisa mendefiniskan 4-,8-, atau c-jalur, tergantung dari jenis kedekatan yang dibutuhkan.

Jika *p* dan *q* adalah piksel pada gambar dengan sub himpunan **S**, maka *p* terhubung ke dalam **S** jika jalur dari *p* dan *q* terdiri dari semua piksel yang ada di **S**. Untuk setiap piksel *p* di **S**, himpunan piksel di **S** yang terhubung ke *p* disebut komponen terhubung (*connected component)* dari **S**. Karena itu tiap dua piksel dari sebuah komponen terhubung adalah terhubung satu sama lainnya, dan komponen terhubung yang berlainan adalah saling lepas (*disjoint*).

2.3 Segmentasi Citra

Segmentasi adalah proses yang membagi suatu citra ke dalam bagian-bagian atau kelompok-kelompok yang mempunyai persamaan sifat atau ciri-ciri. Ada dua pendekatan dalam proses segmentasi yaitu : *edges based* dan *region based.* Pada pendekatan pertama, dilakukan terlebih dahulu deteksi tepi lokal (lokal diskontinuitas). Selanjutnya dari tepi-tepi lokal tersebut digabung untuk membentuk satu segmen penuh. Sedangkan pendekatan kedua didasarkan pada kesamaan antara region, contohnya *thresholding*, *region growing*, *region splitting* dan *merging*. Pada subbab-subbab berikut ini dibahas tentang beberapa model untuk mendeteksi adanya diskontinuitas.

2.3.1 Deteksi diskontinuitas

Teknik-teknik diskontinuitas digunakan untuk mendeteksi adanya titik terisolasi, garis dan tepi suatu citra. Metode yang umum digunakan untuk mendeteksi ciri-ciri tersebut adalah dengan menggunakan *mask* yang akan dikonvolusikan secara spasial dengan citra yang akan diproses.

w1	w2	w3
w4	w5	w6
w7	w8	w9

Gambar 3. Mask 3x3

w1, w2, ..., w9 menyatakan koefisien atau bobot dari *mask 3x3* seperti yang ditunjukkan pada *Gambar 3*. Jika *x1, x2,..., x9* menyatakan *gray-level* dari piksel di bawah *mask*, maka masing-masing nilai bobot dan nilai *gray-level* dapat dinyatakan dengan vektor kolom berikut :

$$w = \begin{bmatrix} w1 \\ w2 \\ . \\ . \\ .w9 \end{bmatrix}$$

dan

$$x = \begin{bmatrix} x1 \\ x2 \\ . \\ . \\ . \\ x9 \end{bmatrix}$$

Kemudian, kita dapat menyatakan nilai *gray-level* baru sebagai perkalian dalam antara *w* dan *x*.

$$T[x] = w^T x = w_1.x_1 + w_2.x_2 + ... + w_9.x_9 \qquad (2.3)$$

dimana, *T[x]* = nilai *gray-level* yang baru

$w^T x$ = perkalian antara vektor *w* dengan vektor *x*

$w_i x_i$ = perkalian antara nilai bobot w_i dengan nilai *gray-level* x_i yang terletak tepat di bawah nilai bobot w_i

2.3.2 Deteksi titik

Tujuan deteksi titik dalam suatu citra adalah untuk menghilangkan titik yang terisolasi atau *noise* dalam analisa praktis. *Mask* yang digunakan sebagai dasar deteksi titik ditunjukkan pada gambar di bawah ini :

-1	-1	-1
-1	8	-1
-1	-1	-1

Gambar 4. Mask yang digunakan untuk mendeteksi titik terisolasi

Gambar di atas dapat didefiniskan sebagai sebuh persamaan :

$$w'.x = -x_1 - x_2 - x_3 - x_4 + 8.x_5 - x_6 - x_7 - x_8 - x_9 \qquad (2.4)$$

Daerah dengan *gray-level* yang konstan hasil dari operasi di atas adalah 0. Sebaliknya, jika pada titik pusat *mask* ada suatu titik yang terisolasi, yang intensitasnya lebih besar daripada latar belakangnya, maka akan dihasilkan nilai yang lebih besar dari 0.

2.3.3 Deteksi garis

Hampir sama dengan deteksi titik, perbedaannya terletak pada obyek yang dideteksi, yaitu berupa segmen-segmen garis. Terdapat beberapa proses untuk mendeteksi segmen-segmen garis. Pertama dilakukan proses sepanjang arah horisontal yang mendeteksi garis pada arah–arah horisontal. Proses yang kedua

dilakukan sepanjang arah vertikal yang mendeteksi adanya garis-garis pada arah vertikal. Sedangkan proses deteksi garis yang lain adalah pada arah –45° dengan arah vertikal dan +45° dengan arah vertikal.

Beberapa contoh *mask* yang sering digunakan untuk mendeteksi segmen-segmen garis dapat dilihat seperti yang ada pada gambar dibawah ini :

-1	-1	-1
2	2	2
-1	-1	-1

-1	2	-1
-1	2	-1
-1	2	-1

-1	-1	2
-1	2	-1
2	-1	-1

2	-1	-1
-1	2	-1
-1	-1	2

Gambar 5. Contoh mask untuk mendeteksi segmen-segmen garis

(a) Mask untuk mendeteksi segmen-segmen garis pada arah Horisontal

(b) Mask untuk mendeteksi segmen-segmen garis pada arah Vertikal

(c) Mask untuk mendeteksi segmen-segmen garis pada arah -45^0

(d) Mask untuk mendeteksi segmen-segmen garis pada arah +45^0

2.3.4 Deteksi tepi

Deteksi tepi merupakan salah satu proses yang terjadi sebelum proses pengolahan yang sering dibutuhkan pada analisis citra. Proses tersebut bertujuan untuk meningkatkan penampakan garis pada citra. Jadi prosesnya bersifat memperkuat komponen frekuensi tinggi.

Tepi suatu citra dapat kita definisikan sebagai batas (*boundary*) di antara dua daerah (*region*) yang mempunyai ciri-ciri *gray-level* yang berbeda. Tepi suatu citra

menyatakan diskontinuitas dari intensitas piksel dalam suatu citra. Informasi yang terdapat dalam citra dua dimensi (*2-D*) dapat direduksi dengan proses deteksi tepi.

2.3.4.1 Filter Laplacian

Filter Laplacian adalah suatu pengukuran 2D isotropik dari turunan parsial kedua dari sebuah citra. Nilai Laplacian *L(x, y)* sebuah citra dengan nilai intensitas piksel *I(x, y)* didefiniskan seperti di bawah ini :

$$L(x,y)=\frac{\partial^2 I}{\partial x^2}+\frac{\partial^2 I}{\partial y^2} \qquad (2.5)$$

Mask untuk operator Laplacian dapat dilihat pada gambar di bawah ini :

0	-1	0
-1	4	-1
0	-1	0

Gambar 6. Mask Laplacian yang sering digunakan

dengan menggunakan persamaan :

$$T[x] = wT\,x = w1.x1 + w2.x2 + \ldots + w9.x9 \qquad (2.6)$$

Nilai gray level baru dapat diperoleh dengan persamaan berikut ini:

$$L[f(x,y)] = x2 + x4 + x6 + x8 \qquad (2.7)$$

2.3.4.2 Filter Gaussian

Filter Gaussian merupkan salah satu *filter lowpass*, yang berfungsi sebagai media penghalusan citra (*image smoothing*). Kondisi yang dibentuk oleh filter ini

$$G(x, y) = \frac{1}{2\pi\sigma^2} \frac{(x^2+y^2)}{2\sigma^2} \quad (2.8)$$

dikenal dengan istilah *Gaussian Kernel.* Notasi x dan y merupakan koordinat kartesius filter, σ merupakan nilai sigma yang menentukan tingkat kehalusan citra yang diproses dan nilainya dapat ditentukan oleh pengguna, sedangkan $g(x, y)$ merupakan kernel untuk filter Gaussian.

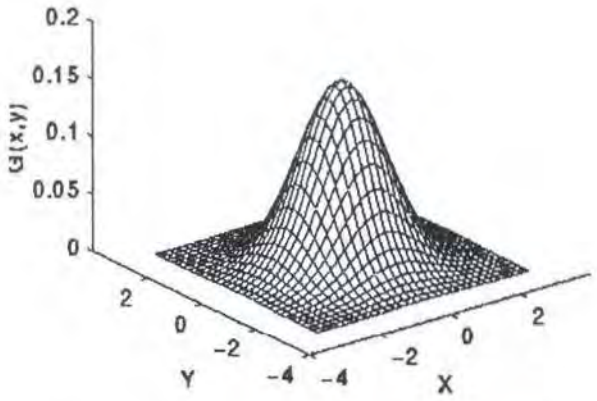
Fungsi persamaan untuk Gaussian Filter ditunjukkan pada gambar di bawah ini :

$\frac{1}{115}$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Gambar 7. Mask Gaussian yang sering digunakan

Hasil yang diperoleh pada kernel ini dikonvolusikan pada citra asal yang akan menjadi lebih halus. Tujuan dari penghalusan Gaussian yakni untuk menggunakan distibusi 2D sebagai sebuah fungsi *point-spread* dan ini dicapai melalui konvolusi.



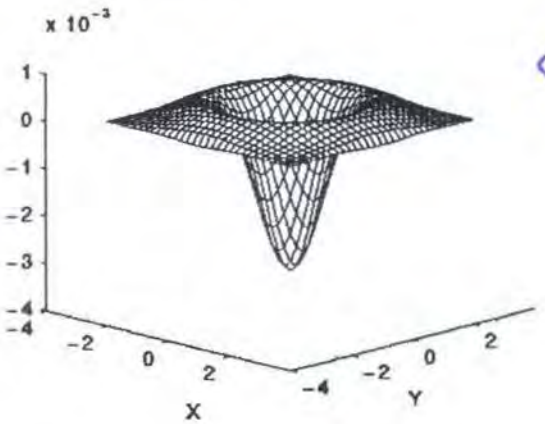
Gambar 8. Fungsi dari filter Gaussian

2.3.4.3 Filter Laplacian of Gaussian

Untuk fungsi Laplacian of Gaussian (LoG) difokuskan pada nilai nol dan dengan standar deviasi Gaussian σ yang mempunyai bentuk :

$$LoG(x,y) = -\frac{1}{\pi\sigma^2}\left[1-\frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (2.9)$$

dan ditunjukkan pada gambar di bawah ini :



Gambar 9. Fungsi dari filter Laplacian of Gaussian

Apabila digunakan sebuah deviasi Gaussian $\sigma = 1.4$, *LoG(x,y)* menjadi

0	0	3	2	2	2	3	0	0
0	2	3	5	5	5	3	2	0
3	3	5	3	0	3	5	3	3
2	5	3	-12	-23	-12	3	5	2
2	5	0	-23	-40	-23	0	5	2
2	5	3	-12	-23	-12	3	5	2
3	3	5	3	0	3	5	3	3
0	2	3	5	5	5	3	2	0
0	0	3	2	2	2	3	0	0

Gambar 10. Mask LoG

Operator LoG menghitung turunan spasial kedua dari sebuah citra.Artinya bahwa dalam daerah di mana citra mempunyai sebuah nilai intensitas konstan, misal gradien intensitas bernilai nol, respon LoG akan bernilai nol. Pada perubahan intensitas, bagaimanapun respon LoG akan bernilai positif pada kondisi gelap dan akan bernilai negatif pada kondisi terang. Artinya bahwa sebuah tepi citra antara dua region bersifat sama tetapi intensitasnya berbeda, respon LoG akan bernilai :

1. nol pada sebuah jarak yang panjang antar tepi
2. positif hanya untuk satu sisi dari tepi
3. negatif hanya pada sisi lainnya dari tepi
4. nol pada beberapa titik diantaranya, pada tepi itu sendiri

2.4 Warna Sebuah Citra

Persepsi warna tergantung pada sifat-sifat fisik cahaya yang merupakan energi elektromagnet dan interaksinya dengan benda-benda fisik, serta pada interpretasi oleh sistem saraf penglihatan manusia.

Sistem penglihatan manusia dapat menangkap energi elektromagnet dengan panjang gelombang kira-kira 400 hingga 700 *nanometer* (1 *nanometer* = 10^{-9} meter) sebagai cahaya nampak. Cahaya dapat diterima baik secara langsung dari sumber cahaya seperti bola lampu atau secara tidak langsung dengan pemantulan dari permukaan suatu benda ataupun pembiasan melalui suatu benda.

Jika cahaya yang diterima oleh mata terdiri dari semua panjang gelombang dengan jumlah yang sama, maka sumber cahaya atau benda yang menghasilkan cahaya tersebut disebut dengan sumber cahaya atau benda yang akromatik. Sumber cahaya atau benda seperti ini akan terlihat putih abu-abu atau hitam. Sebuah benda yang memantulkan 80% dari seluruh sinar secara akromatik akan terlihat putih. Sedangkan bila jumlah cahaya tersebut kurang dari 3%, benda tersebut akan terlihat hitam.

2.4.1 Jenis model warna

Ada dua sistem pencampuran warna utama dalam komputer grafik: sistem *red, green dan blue* (RGB) dan sistem *cyan, magenta dan yellow* (CMY). Sistem RGB dikatakan sebagai sistem warna yang aditif sedangkan sistem CMY merupakan sistem warna yang subtraktif. Kedua sistem warna ini merupakan komplemen satu sama lain. Yang dimaksud dengan warna komplemen adalah

warna yang dihasilkan apabila warna putih minus suatu warna. Karena itu *cyan* adalah putih minus merah, magenta adalah putih minus hijau dan kuning adalah putih minus biru.

Sistem CMY banyak digunakan pada tinta printer dan film. Sedangkan sistem RGB banyak digunakan untuk CRT display. Pada umumnya nilai warna *C* didapat dari :

$$C = rR + gG + bB \qquad (2.10)$$

RGB merupakan sinar merah, hijau dan biru sedangkan *r, g, b* berhubungan dengan bobot relatif dari *R, G* dan *B* dengan nilai antara 0 dan 1. Sedangkan transformasi antara sistem RGB dan CMY adalah:

$$[R \quad G \quad B] = [1 \quad 1 \quad 1] - [C \quad M \quad Y] \qquad (2.11)$$

Model warna yang lain adalah sistem warna HSL, yang menyampaikan informasi *Hue, Saturation* dan *Luminance*. *Hue* adalah nilai kewarnaan seperti kemerahan, kehijauan, kekuningan dan sebagainya. Nilai ini dinyatakan dengan sudut 0 sampai 360 derajat. *Saturation* menyatakan tingkat kemurnian warna dalam proses. Warna murni mempunyai harga S sebesar 100%. Penambahan warna putih akan mengurangi harga S. Harga *luminance* menunjukkan intensitas warna, semakin tinggi harga L maka semakin cerah warnanya.

2.4.2 Faktor-faktor penentu warna benda

Seperti telah disebutkan pada bagian sebelumnya bahwa warna dari suatu benda yang kita lihat, bukan hanya merupakan warna obyek itu sendiri melainkan telah dipengaruhi oleh bermacam-macam faktor.

Ada dua jenis sumber cahaya yang menerangi benda yaitu sumber cahaya titik dan sinar *ambient*. Yang dimaksud dengan sumber cahaya titik adalah sumber cahaya yang memancarkan sinar ke segala arah. Sedangkan sinar *ambient* adalah sinar yang tampak dari segala arah atau cahaya dari lingkungan sekitar. Biasanya sinar ini selalu terdapat di sekitar benda dan sulit dihitung tepatnya.

Cahaya yang mengenai permukaan benda, berinteraksi dalam tiga cara yang berbeda yaitu :

1. Sebagian diserap oleh permukaan benda dan diubah menjadi panas
2. Sebagian dipantulkan oleh permukaan
3. Sebagian disebarkan ke dalam benda seperti dalam sebuah gelas.

Jumlah energi cahaya yang diserap, dipantulakan atau diteruskan tergantung dari panjang gelombang cahaya. Jika perbandingan panjang gelombang dari seluruh cahaya yang dipantulkan sama maka permukaan benda akan bervariasi antara putih, abu-abu dan hitam. Jika beberapa cahaya mempunyai panjang gelombang berbeda, maka permukaan tersebut akan kelihatan "berwarna". Sehingga sebuah benda akan "berwarna" tergantung dari panjang gelombang yang diserap permukaan.

Sifat-sifat cahaya yang dipantulkan oleh permukaan benda dipengaruhi oleh komposisi, arah dan letak sumber cahaya, orientasi permukaan dan sifat-sifat permukaan benda. Cahaya yang dipantulkan oleh permukaan benda juga dapat dibedakan dari cara pemantulan cahaya, yakni secara *diffuse* dan secara spekular. Sebuah cahaya dikatakan dipantulkan secara *diffuse*, jika cahaya tersebut sudah menembus permukaan benda, kemudian diserap dan dipantulkan kembali. Cahaya

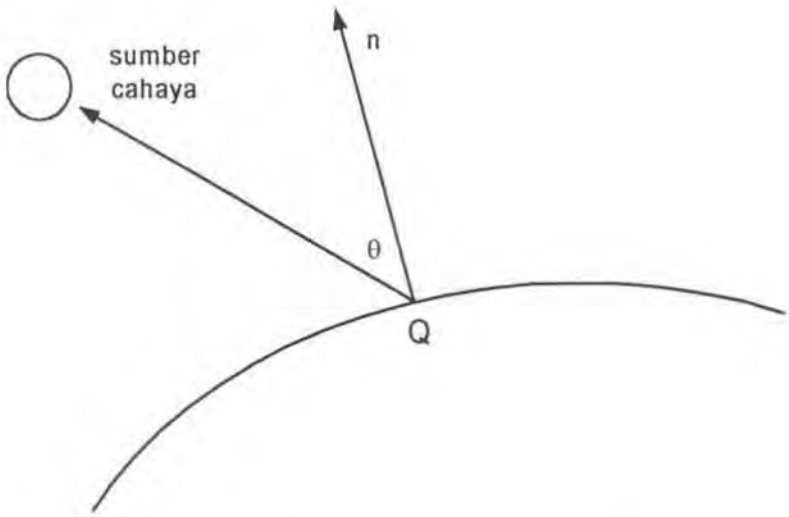
hasil pemantulan secara *diffuse* ini tersebar secara merata pada keseluruhan arah, jadi letak pengamat tidaklah penting. Pemantulan secara spekular terjadi di luar permukaan benda dan pemantulan ini tergantung pada posisi pengamat.

2.4.2.1 Karakteristik diffuse

Untuk permukaan sempurna akan memantulkan cahaya secara *diffuse* dan tingkat intensitas cahaya yang dipantulkan sesuai hukum kosinus Lambert. Hukum ini menyatakan bahwa intensitas cahaya pantulan berbanding lurus dengan kosinus sudut antara arah cahaya dengan sumbu normal permukaan

$$I = I_l k_d \cos\theta \qquad 0 \le \theta \le \pi/2 \tag{2.12}$$

I adalah intensitas cahaya yang dipantulkan, I_l adalah intensitas cahaya yang memantul dari permukaan benda, k_d adalah konstanta *diffuse* yang mempunyai nilai antara 0 dan 1, dan θ adalah sudut antara arah sinar dengan vektor normal dari permukaan benda seperti terlihat pada gambar di bawah ini



Gambar 11. Pemantulan diffuse

Konstanta *diffuse* mempunyai nilai yang tergantung pada materi pembentuk permukaan, konstanta ini juga merupakan fungsi dari panjang gelombang. Artinya besarnya konstanta ini dipengaruhi oleh panjang gelombang dari cahaya yang datang. Jadi konstanta *diffuse* berbeda untuk jenis warna yang berbeda pula.

2.4.2.2 Pengaruh cahaya dari lingkungan

Pewarnaan Lambertian mengasumsikan bahwa cahaya hanya datang dari sebuah sumber cahaya. Karena asumsi ini, permukaan yang tidak secara langsung menerima cahaya dari sumber cahaya tidak akan menerima cahaya sama sekali sehingga akan terlihat hitam. Benda yang diwarnai secara Lambertian terlihat kurang alamiah, karena dalam kenyataan permukaan benda juga menerima sinar dari lingkungan sekitarnya.

Sinar semacam ini dikenal dengan sebutan sinar *ambient*. Sinar *ambient* berfungsi sebagai sumber cahaya yang sama, mempunyai sifat banyak jumlahnya dan terpencar. Sumber-sumber sinar dari alam ini didekati dengan menambahkan sebuah suku secara linier pada model Lambertian. Dengan adanya sinar *ambient* ini, persamaan 2.12 akan menjadi :

$$I = I_a k_a + I_l k_d \cos\theta \qquad 0 \le \theta \le \pi/2 \qquad (2.13)$$

I_a adalah intensitas sinar *ambient*, sedangkan k_a adalah konstanta *diffuse ambient* yang bernilai antara 0 dan 1. Pewarnaan dengan acuan persamaan di atas masih kurang alamiah. Karena persamaan ini tidak memiliki faktor yang menunjukkan jarak benda dengan sumber cahaya dan jarak benda atau sumber cahaya ke mata

pengamat. Karena jarak tersebut mempunyai arti yang penting maka intensitas cahaya akan berkurang sejauh sinar melintas.

Pelemahan intensitas sinar karena pengaruh jarak akan berbanding terbalik dengan kuadrat jarak yang ditempuh sinar. Dengan hukum pelemahan secara linier (*linier attenuation law*), maka pelemahan akan berbanding terbalik dengan jarak ditambah suatu konstanta. Persamaan 2.13 setelah dimodifikasi dengan perubahan unsur jarak, menjadi :

$$I = I_a k_a + \frac{I_l k_d \cos\theta}{d + k} \qquad 0 \le \theta \le \pi/2 \qquad (2.14)$$

k adalah sebuah konstanta sembarang dan *d* adalah jarak dari titik pemantulan sinar atau sumber cahaya ke pengamat.

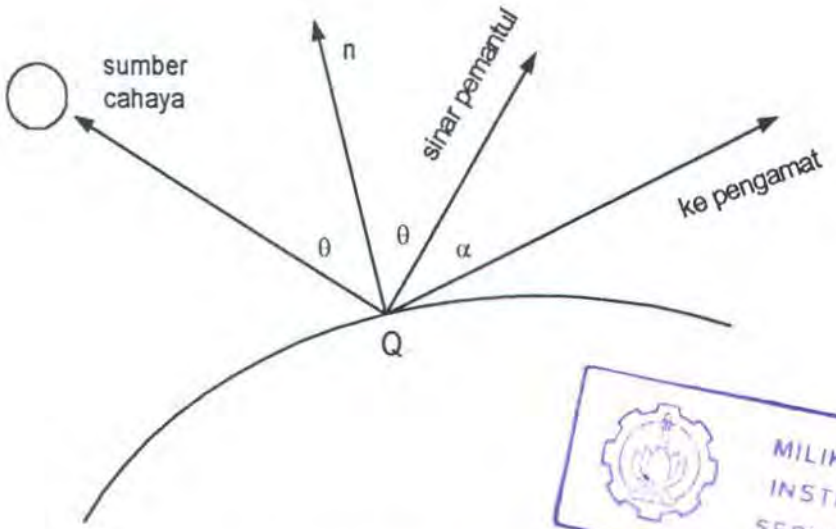
Dalam beberapa kasus, sumber cahaya mempunyai jarak tak hingga, misalnya sumber cahaya dari matahari. Faktor pelemah yang mempunyai kemungkinan nilainya besar sekali akan mengakibatkan suku terakhir persamaan 2.14 bernilai nol. Masalah ini dipecahakan dengan mengabaikan faktor jarak untuk sumber-sumber cahaya yang mempunyai jarak tak terhingga. Nilai *d* dihitung dari jarak benda terdekat dengan pengamat. Dengan cara ini, benda yang terletak paling dekat dengan pengamat akan memiliki intensitas penuh dan benda-benda yang lebih jauh mempunyai intensitas yang lebih rendah.

2.4.2.3 Karakteristik spekular

Selain pemantulan *diffuse* seperti dijelaskan pada bagian sebelumnya, terdapat pemantulan secara spekular. Intensitas cahaya hasil pemantulan spekular ini tergantung pada sudut datang sinar, panjang gelombang sinar dan sifat-sifat fisik

permukaan pemantul. Seperti dijelaskan sebelumnya, pemantulan spekular ini tergantung pada arah sinar dan letak pengamat. Sebuah cermin yang merupakan permukaan yang dapat mencerminkan secara sempurna, sudut pantul akan sama dengan sudut datang sesuai dengan hukum Snellius. Hal ini menyebabkan hanya pengamat yang terletak tepat pada arah sinar yang dapat melihat sinar.

Peristiwa seperti ini digambarkan pada gambar di bawah ini denga arah penglihatan *S*, berhimpit dengan vektor pantul *R*, atau besar sudut α adalah nol.



Gambar 12. Pemantulan spekular

Pada pemantulan tidak sempurna, jumlah sinar yang mencapai pengamat tergantung pada distribusi spasialnya yang terfokus. Sedangkan permukaan yang kasar mempunyai distribusi spasial yang tersebar.

Kelemahan cahaya dari permukaan benda yang mengkilap adalah efek dari pemantulan secara spekular. Karena cahaya hasil pantulan terfokus pada

pengkilatan, maka kilauan tersebut akan berpindah-pindah jika pengamat berpindah juga. Lebih jauh lagi, cahaya hasil pantulan secara spekular mempunyai karakteristik yang sama dengan cahaya sumber. Contohnya yaitu kilauan pada sebuah permukaan mengkilap yang berwarna biru, tidak akan berwarna biru melainkan berwarna putih.

Implementasi pemantulan secara spekular ini dilakukan dengan model yang diterangkan secara empiris oleh Bui-Tuong Phong. Persamaan yang dimaksud adalah :

$$I_s = I_1 \omega(i,\lambda)\cos^n \alpha \qquad (2.15)$$

$\omega(i,\lambda)$ adalah kurva pemantulan, yakni perbandingan antara cahaya hasil pemantulan dengan cahaya sumber sebagai sebuah fungsi sudut datang *I* dan panjang gelombang λ. Sedangkan *n* adalah sebuah pangkat yang mewakili distribusi spasial dari cahaya yang dipantulkan. Harga *n* yang besar menghasilkan distribusi spasial yang terfokus, seperti dihasilkan oleh permukaan logam atau benda mengkilap lainnya.

BAB III

COLOR CONSTANCY

BAB III

COLOR CONSTANCY

Pada bab ini akan dibahas mengenai konsep *color constancy* yang merupakan dasar metode *color constant color indexing*, metode yang digunakan dalam pembuatan tugas akhir ini. Membahas mengenai definisi dan konsep awal *color constancy*, dan dilanjutkan dengan beberapa uraian permasalahan pada konsep ini beserta pembatasan yang dipakai pada konsep ini.

3.1 Definisi dan Konsep

Warna sebuah obyek dapat dikatakan tidak bergantung terhadap pencahayaan yang mengenainya. Warna merupakan suatu besaran permukaan obyek, bukan besaran cahaya pantul yang mengenainya. Kemampuan untuk mengenali sebuah obyek lewat warnanya, dan warna di sini merupakan hasil pemantulan dari permukaan obyek, disebut dengan *color constancy*.

Pemantulan cahaya dari sebuah permukaan tergantung tidak hanya pada spektrum pencahayaan dan pemantulan permukaan, akan tetapi masih terdapat beberapa faktor lainnya, yakni spekularitas dan seluruh pencahayaan yang ada. Untuk maksud tersebut dalam perhitungan *color constancy* sering dibuat dengan kondisi sebuah permukaan yang datar.

3.1.1 Deskripsi respon sensor

Cahaya pantul dari sebuah permukaan yang mengenai sebuah permukaan yang datar, dianalogikan seperti pada sebuah retina mata manusia. Pada masing-masing lokasi *x*, terdapat jenis sensor yang berbeda. Nilainya dinyatakan dengan sensor ke-*k* (sebuah skalar), ρ_k^x merupakan integral dari fungsi respon dikalikan dengan sinyal warna yang datang. Masing-masing ρ_k^x dicocokkan dengan sebuah pemantulan permukaan. Persamaan matematikanya dapat ditulis menjadi :

$$\rho_k^x = \int_{\omega} C^x(\lambda) R_x(\lambda) d\lambda \qquad (3.1)$$

dimana λ merupakan panjang gelombang, $R_k(\lambda)$ adalah fungsi respon dari sensor ke-*k*, $C^x(\lambda)$ adalah sinyal warna pada lokasi *x* dan fungsi integral didapat pada spektrum yang tampak ω. Sinyal warna merupakan hasil dari sebuah pemantulan tunggal permukaan, $S(\lambda)$ dikalikan dengan pencahayaan *ambient* $E(\lambda)$, $C(\lambda) = E(\lambda)S(\lambda)$.

3.1.2 Deskripsi permukaan

Tujuan dari *color constancy* adalah mengubah vektor respon sensor $\underline{\rho}^x$ menjadi sebuah deskripsi permukaannya, $\underline{d}^x$, dimana $\underline{d}^x$ merupakan representasi dari 3 sifat pemantulan yang terdapat pada permukaan benda dan berubah-ubah terhadap nilai $E(\lambda)$. Sehingga permukaan dengan perbedaan fungsi pemantulan permukaan harus juga mempunyai perbedaan deskripsi permukaan yang berubah-ubah. Secara formal dapat ditulis :

$$\underline{d}^x = \mathrm{T}\left(E(\lambda) : \underline{\rho}^x\right) \qquad (3.2)$$

dimana $\underline{d}^x$ merupakan sebuah pencahayaan yang tergantung pada transformasi dari $\underline{\rho}^x$. Pada kondisi pencahayaan yang sama, sebuah transformasi tunggal akan menggunakan keseluruhan citra. Transformasi *T* sering dianggap bersifat linier. Pada kasus ini $T\left(E(\lambda):\underline{\rho}^x\right)=T(E(\lambda))\underline{\rho}^x$. Sehingga jika terdapat 3 sensor, *s* = 3, maka $T(E(\lambda))$ berupa matriks *3x3*.

3.1.3 Fungsi kontinuitas sebagai vektor

Sebuah fungsi dimensi satu $F(\lambda)$ yang berubah terhadap λ dapat di anggap sebuah interval tertutup dari λ, dan bisa dianggap sebagai sebuah vektor. Sehingga fungsi λ dapat dideskripsikan nilainya sebagai sebuah nilai diskrit dari panjang gelombang yang ada pada spektrum tampak. Dengan menggunakan spektrum, λ dimisalkan sebesar 10*nm* dengan interval antara 400 sampai 650*nm* (vektor mempunyai 26 komponen). *R(λ), C(λ), E(λ)* dan *S(λ)* diubah menjadi vektor : $\underline{R}$, $\underline{C}$, $\underline{E}$, dan $\underline{S}$. Dan *s* sensor menjadi sebuah matriks *R*, *26 x s* . Kolom ke-*k* dari *R* merupakan vektor *reseptor* ke-*k*. Sehingga dapat ditulis :

$$\rho_k^x = \sum_{i=1}^{26} R_{ik} \qquad (3.3)$$

Rumus di atas merupakan perkalian skalar vektor dari sensor ke-*k* dengan sinyal warna. Disamping itu, respon sensor ke-*k* dapat dihitung lewat rumus di bawah ini, dimana indeks *k* bersifat menurun :

$$\underline{\rho}^x = R^t C \qquad (3.4)$$

dimana *t* merupakan matriks *transpose*. Dengan demikian representasi dari vektor sangat berguna untuk analisa kemungkinan pengukuran secara tepat fungsi spektrum secara kompleks. Teknik aljabar vektor sering digunakan pada waktu perhitungan *color constancy*.

3.1.4 Model dimensi hingga

Warna dimodelkan menggunakan sebuah model linier dimensi hingga untuk pemantulan permukaan dan pencahayaannya. Misal *S* merupakan sebuah matriks d^S (dimensi dari *S*) vektor-vektor basis pemantulan. Dengan demikian sebuah vektor pemantulan permukaan $\underline{S}$ didefinisikan :

$$\underline{S} \approx S\underline{\sigma} \qquad (3.5)$$

dimana $\underline{\sigma}$ adalah sebuah komponen vektor kolom berat d^S. Maloney membuktikan pemantulan permukaan dapat secara baik dimodelkan dengan sebuah kumpulan vektor basis antara 3 dan 6. Secara sederhana pencahayaan sering dimodelkan dengan sebuah kumpulan kecil dari vektor basis. Misal ξ merupakan sebuah matriks vektor basis d^E, maka :

$$\underline{E} = \xi\underline{\varepsilon} \qquad (3.6)$$

dimana $\underline{\varepsilon}$ merupakan sebuah vektor berat dimensi d^E .

3.2 Penyelarasan Von Kries

Von Kries menerangkan mengenai respon penyelarasan pada sebuah permukaan $S(\lambda)$ pada sebuah chanel *k* dirumuskan sebagai berikut :

$$d_k^x = \frac{\int S_x(\lambda)E(\lambda)R_x(\lambda)d\lambda}{\int E(\lambda)R_x(\lambda)d\lambda} \qquad (3.7)$$

Von Kries berpendapat bahwa untuk sebuah pencahayaan E, d^x_k akan tetap konstan. Untuk membatasi $E(\lambda)$, beberapa penulis berasumsi bahwa terdapat sebuah *receptor* warna putih dalam setiap bidang. Asumsi ini bersifat umum untuk beberapa algoritma untuk *color constancy*, termasuk teori *retinex* dari Land yang akan dijelaskan pada bagian selanjutnya..

Konsep Von Kries ini sebenarnya merupakan sebuah transformasi matriks diagonal dengan koefisien pada tiap-tiap *channel* didefinisikan :

$$c_k = \frac{1}{\int E(\lambda)R_k(\lambda)d\lambda} \qquad (3.8)$$

3.3 Algoritma Land

Algoritma *retinex* Land menyatakan bahwa koefisien dari transformasi matriks diagonal dengan asumsi bahwa pada tiap bidang berisi sebuah *reflector* yang seragam. Bagaimanapun kecuali pada konsep Von Kries, mata manusia secara eksplisit tidak dapat mengukur pencahayaan. Disamping itu algoritma *retinex* menerangkan bagaimana menghubungkan respon sensor dengan *receptor* warna putih.

Dengan rasio respon sensor yang berada pada lokasi x_1 dan x_2, misal ρ_k^{x1}/ρ_k^{x2}. Maka jika x_1 berupa bilangan yang nyata maka rasio akan lebih kecil ketika x_2 dicocokkan ke *receptor* warna putih (nilai pemantulan antara 0 dan 1). Lebih lanjut dengan sebuah lintasan acak secara kontinu yang terdiri dari x_1, x_2,

$x_3, \ldots x_N$. Rasio ρ_k^{x1}/ρ_k^{x2} dapat dihitung sebagai lintasan dengan mengalikan rasio lokal. Misal :

$$\frac{\rho_k^{x4}}{\rho_k^{x1}} = \frac{\rho_k^{x4}}{\rho_k^{x3}} \frac{\rho_k^{x3}}{\rho_k^{x2}} \frac{\rho_k^{x2}}{\rho_k^{x1}} \qquad (3.9)$$

Land menyebutkan bahwa sebuah rasio yang dihitung pada kondisi yang menanjak (*incremental*) sebagai sebuah *designator*. Algoritma *retinex* berasumsi banyak lintasan acak yang dapat dihitung. Pada tiap lokasi nilai *designator* terkecil disimpan. Jika lintasan acak mencukupi untuk dihitung maka designator pada keseluruhan lokasi akan relatif terhadap *receptor* warna putih. Dengan demikian pencahayaan akan terkurangi dan designator dari *color constancy* akan diperoleh.

Untuk mengubah intensitas pencahayaan secara perlahan, rasio lokal dibatasi. Dengan demikian jika ρ_k^{x1}/ρ_k^{x2} diperkirakan mendekati satu kemudian respon sensor dipertimbangkan berhubungan ke pemantulan permukaan yang sama. Secara beraturan rasio ini di set menjadi satu.

Untuk memperoleh intensitas dari pencahayaan, rasio lokal harus dibatasi. Kemudian jika ρ_k^{x1}/ρ_k^{x2} diperkirakan bernilai hampir 1 maka respon sensor menunjukkan pemantulan permukaan yang sama. Dengan demikian rasio bisa dibatasi menjadi 1.

Asumsi white patch sangatlah kuat batasannya sehingga Land memodifikai menjadi algoritma *retinex* dengan menggunakan asumsi bahwa rata-rata dari keseluruhan designator pada tiap-tiap lokasi citra bersifat konstan. Brainard menyatakan bahwa jika terdapat banyak lintasan acak dihitung kemudian rata-rata designator pada x_α bernilai :

$$\frac{(n-1)\rho_k^{x\alpha}}{\sum_{i=1}^{N-1}\rho_k^{xi}} \qquad (3.10)$$

dimana *i* merupakan indeks dari keseluruhan piksel pada citra.

3.4 Algoritma Horn

Horn mengambil asumsi *retinex* dari Land pada algoritma ini, terutama pada notasi acak lintasannya. Citra dinormalisasi dengan sebuah pencahayaan yang sama . Dengan demikian respon dari dua buah lokasi citra dapat dengan secara langsung dibandingkan. Proses perhitungannya dapat dijelaskan sebagai berikut :

1. Fungsi logaritma warna citra dihitung, tujuannya untuk memisahkan secara efektif antara pemantulan dan komponen pencahayaan. Dengan menggunakan fungsi logaritma pada kedua sisi dari persamaan :

   $$d_x^k = c_k p_x^k \qquad (3.11)$$

   didapat : $\log(d_x^k) = \log(c_x) + \log(p_x^k)$ (3.11)

2. Perubahan pemantulan dibedakan dari jenis pencahayaan dengan menggunakan rumus fungsi Laplacian untuk fungsi logaritma dari warna citra. Untuk nilai fungsi Laplacian yang kecil merupakan nilai gradien pencahayaan, sedangkan untuk nilai yang besar merupakan besarnya sudut pemantulan. Batas dari fungsi Laplacian secara efektif menghilangkan jenis pencahayaan
3. Kemampuan fungsi inverse Laplacian memberi sebuah nilai log warna citra yang baru. Dan antilog dari hasil ini pada sebuah citra diambil pada sebuah pencahayaan (pencahayaan yang tidak diketahui)

Adapun beberapa masalah pada algoritma Horn, yakni :

1. Untuk memecahkan fungsi inverse Laplacian dibutuhkan pembatasan. Dengan kata lain respon sensor pada batas keseluruhan citra harus berhubungan dengan pemantulan permukaan yang sama
2. Konsep *color constacy* masih membutuhkan sebuah *patch* pemantulan. Meskipun teori Land mengenai white patch atau rata-rata patch bisa digunakan, akan tetapi algoritma ini masih membutuhkan warna-warna lain yang ada pada bidang citra

BAB IV

COLOR CONSTANT COLOR INDEXING

BAB IV

COLOR CONSTANT COLOR INDEXING

Pada bab ini akan dijelaskan tentang dasar teori metode yang digunakan pada tugas akhir ini, yakni metode *color constant color indexing*. Sebagai perbandingan akan dibahas terlebih dahulu metode *color indexing* yang menjadi dasar pengembangan metode yang digunakan pada tugas akhir ini. Metode *color constant color indexing* mempunyai kemampuan lebih baik dalam mengenali perubahan intensitas pencahayaan yang mengenai obyek dibandingkan dengan metode *color indexing*.

4.1 Color Indexing

Swain mengemukakan sebuah algoritma untuk mengidentifikasi sebuah obyek, dengan cara melakukan pengindeksan warnanya Masing-masing warna obyek disimpan dalam sebuah histogram 3 dimensi. Obyek dipisahkan dengan citranya terlebih dahulu sebelum dilakukan pengindeksan. Hal ini untuk mencegah warna latar belakang obyek berpengaruh pada saat pengidentifikasian obyek.

Untuk melakukan perbandingan antara dua buah histogram obyek yang berbeda, dilakukan proses irisan histogram (*histogram intersection*).Irisan histogram dari H_1 dan H_2 dari dua buah obyek didefinisikan dengan :

$$H_1 \cap H_2 = \sum_i \sum_j \sum_k \min(H_1(i,j,k), H_2(i,j,k)) \qquad (4.1)$$

i, j, k adalah komponen histogram. Histogram yang didapat bernilai antara 0 dan 1. Untuk menguji performa dari metode Swain ini dilakukan 4 perubahan, antara lain :

1. Mengubah warna latar belakang pada sebuah obyek citra hanya akan menambah nilai perbandingan dari irisan histogram jika :
   a. Piksel mempunyai warna yang sama sebagai salah satu warna dengan obyek yang dibandingkan.
   b. Jumlah piksel dari warna itu dalam model lebih sedikit dibandingkan dengan jumlah warna piksel dari warna citra

   Dengan demikian perbandingan yang benar akan selalu ditemukan kecuali dua obyek sangat sederhana atau warna latar belakang didesain secara khusus untuk merusak proses perbandingan
2. Secara eksperimen program dari Swain bisa digunakan ketika sebuah obyek diubah satu demi satu.
3. Tampilan merubah posisi (atau obyek berotasi), beberapa warna akan tampak dan yang lainnya hilang sehingga mempengaruhi histogram warna. Swain memberi solusi untuk masalah ini dengan menyimpan histogram obyek sama berdasarkan titik pandang yang berbeda.
4. Mengubah intensitas dari cahaya secara efektif mengubah panjang dari komponen warna. Setiap piksel dalam citra akan dikalikan dengan sebuah konstant faktor *k*. Swain menunjukkan secara eksperimen hasil dari beberapa intensitas mengubah dan bahkan menjadikan nilai *k* mendekati 1 sehingga

pengidentifikasian obyek akan terganggu. Mengubah karakteristik spektrum dari pencahayaan akan menghambat algoritma ini.

4.2 Color Constant Color Indexing

Pada metode ini yang diindeks berupa rasio warna, dimana rasio warna beberapa sifat, yakni:

1. Rasio dapat dihitung secara lokal
2. Pencahayaan hanya dibatasi menjadi nilai lokal yang konstan
3. Permukaan pada sebuah tetangga lokal akan cenderung menjadi titik pandang yang independen

Keuntungan pembuatan histogram menggunakan rasio warna :

1. Permukaan Lambertian dengan titik sumber pencahayaan, rasio vektor respon, hubungan antara dua buah permukaan pada orientasi yang sama mempunyai titik pandang yang indenpenden. Misal *v* menyatakan arah pencahayaan dan *n* adalah permukaan normal maka magnitudonya merupakan hasil perkalian skalar *v.n*
2. Histogram rasio tidak dipengaruhi oleh perubahan jarak pandang antara obyek dengan pengamat.

Rasio warna sangatlah efisien dalam penghitungan *log space* lewat sebuah operator beda konvolusi sederhana. Pengaruh beda tersebut merupakan turunan dari *log*-warna pada citra. Kerugiannya, pada turunan pertama rasio warna bersifat

non-isotropik yang mempengaruhi orientasi pengenalan obyek. Pilihan alamiah dari operator isotropik terdiri dari magnitudo gradien atau Laplacian.

Perubahan pemantulan dibedakan dari jenis pencahayaan dengan menggunakan rumus fungsi Laplacian untuk fungsi logaritma dari warna citra. Untuk nilai fungsi Laplacian yang kecil merupakan nilai gradien pencahayaan, sedangkan untuk nilai yang besar merupakan besarnya sudut pemantulan. Batas dari fungsi Laplacian secara efektif menghilangkan jenis pencahayaan. Kemampuan fungsi inverse Laplacian memberi sebuah nilai log warna citra yang baru.

Secara sederhana filter Laplacian dapat ditulis ($-4_{0,0}$; $1_{-1,0}$; $1_{0,-1}$;$1_{0,1}$; $1_{1,0}$) dimana $-4_{0,0}$ merupakan berat dari –4 pada lokasi i_k^z (0,0). Jika merupakan logaritam dari ρ_k^z, kemudian Laplacian pada lokasi citra (x,y) dihitung sebagai $-4i_k^{x,y} + i_k^{x-1,y} + i_k^{x,y-1} + i_k^{x,y+1} + i_k^{x+1,y}$. Pada non-log space mempunyai nilai :

$$\frac{\rho_k^{x-1,y}}{\rho_k^{x,y}} \frac{\rho_k^{x,y-1}}{\rho_k^{x,y}} \frac{\rho_k^{x,y+1}}{\rho_k^{x,y}} \frac{\rho_k^{x+1,y}}{\rho_k^{x,y}} \qquad (4.2)$$

Proses color constant color indexing terdiri dari tiga tahap, yakni :

1. Tahap logaritma

$$i_k(x,y) \Leftarrow \ln\left(\rho_k(x,y)\right) \quad k = 1...3 \qquad (4.3)$$

Pada tahap ini bertujuan untuk menormalisasi nilai warna pada citra, dengan kata lain untuk menghilangkan pengaruh pencahayaan yang mengenai bidang citra.

2. Tahap pencarian perbedaan konvolusi (rumus Laplacian)

$$d_k(x,y) = \nabla^2 i_k(x,y) \quad k = 1...3 \tag{4.4}$$

Pada tahap ini bisa juga disebut dengan proses segmentasi, yang bertujuan untuk memisahkan obyek dengan latar belakangnya.

3. Tahap histogram

$$H(i,j,k) = \sum_{x,y} z = \begin{cases} & d_1(x,y) = i \\ 1 \quad jika & d_2(x,y) = j \\ & d_3(x,y) = k \\ 0 \quad sebaliknya & \end{cases} \tag{4.5}$$

Perhitungan pada sebuah histogram rasio berisi informasi mengenai batas warna, bukan mengenai area warna. Hasil perhitungan pada proses ini untuk obyek yang dijadikan model, dimasukkan ke dalam sebuah database. Sehingga untuk mengenali sebuah obyek, maka nilai histogram dari obyek tersebut akan dibandingkan dengan nilai histogram yang terdapat pada database model dengan menggunakan rumus irisan histogram (metode Swain)

$$H_1 \cap H_2 = \sum_i \sum_j \sum_k \min(H_1(i,j,k), H_2(i,j,k)) \tag{4.6}$$

BAB V

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

BAB V
PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Bab ini menjelaskan mengenai perancangan dan pembuatan perangkat lunak yang meliputi kebutuhan sistem, perancangan input sistem dan proses dan implementasi dari perancangan yang telah dibuat.

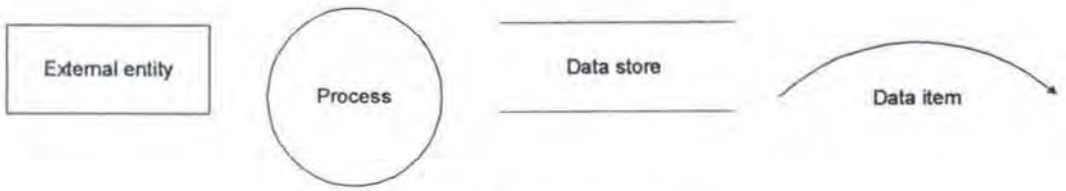
5.1 Kebutuhan Sistem

Sistem perangkat keras yang dibutuhkan untuk menjalankan perangkat lunak ini dengan baik adalah sebuah IBM PC atau kompatibelnya, dengan spesifikasi mikroprosesor minimal Pentium 100. Spesifikasi yang lebih rendah sebenarnya bisa dipakai, akan tetapi akan mengalami kelambatan yang akan terjadi.

Untuk menampilkan gambar diperlukan monitor setara *display card* yang mampu menangani warna sampai 16 juta warna (256 x 256 x 256). Sedangkan untuk kecepatan proses selain mikroprosesor yang diajukan di atas, diperlukan kapasitas memori yang cukup. Manajemen memori yang dilakukan oleh Windows sangat bagus sehingga masalah ini dapat sedikit ditangani. Karena selain menggunakan memori dalam RAM yang ada, Windows dapat memanfaatkan ruang kosong *harddisk* sebagai *virtual memory*. Untuk lebih baiknya memori yang digunakan minimal 32 MByte.

5.2 Deskripsi Perangkat Lunak

Pada subbab ini akan menjelaskan deskripsi dari perangkat lunak yang dirancang dengan menggunakan DFD (Data Flow Diagram). DFD merupakan teknik grafis yang menggambarkan aliran informasi dan informasi data dari input sampai output. Disebut juga *data flow graph* atau *bubble chart*[3]. Komponen dari DFD adalah:



Gambar 13. Komponen DFD

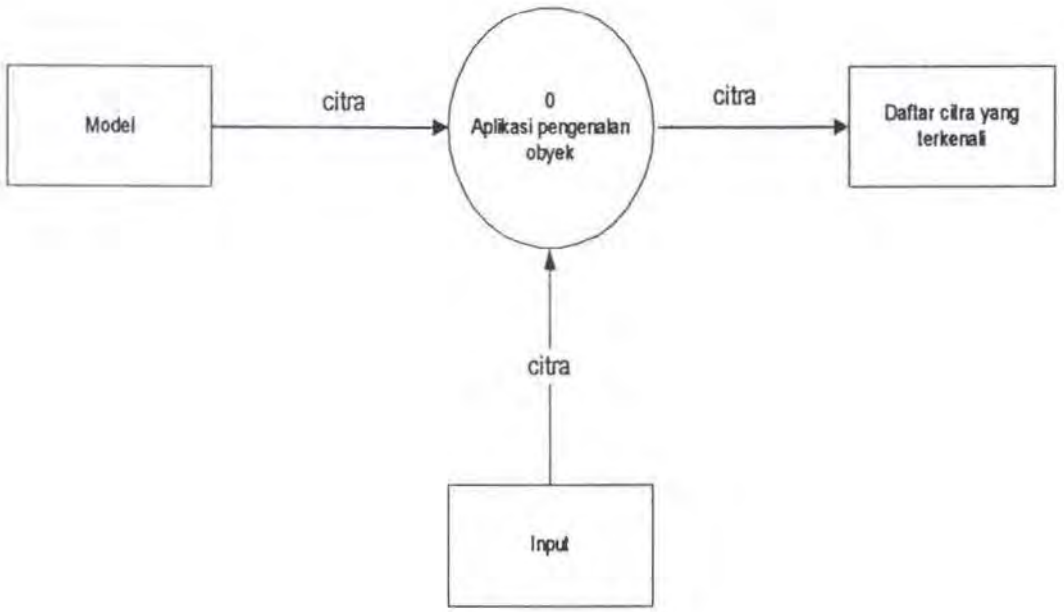
Penjelasan :

- *External entity*, yaitu entitas yang berperan sebagai penghasil atau pemakai informasi yang berada di luar sistem yang dimodelkan
- *Process*, yaitu proses transformasi informasi yang berada dalam sistem, satu atau beberapa proses. Biasanya berupa *file* data dalam media penyimpan.
- *Data store*, yaitu satu atau sekumpulan data atau informasi untuk penyimpanan data pada waktu proses

[3] Roger S. Pressman, *Software Engineering a Practitioner's Approach*, 3rd edition, McGraw-Hill International, Singapore, 1992, hal. 209

- *Data item*, yaitu satu atau sekumpulan data atau informasi yang mengalir dari satu proses ke proses lainnya. Tanda panah menunjukkan arah aliran data.

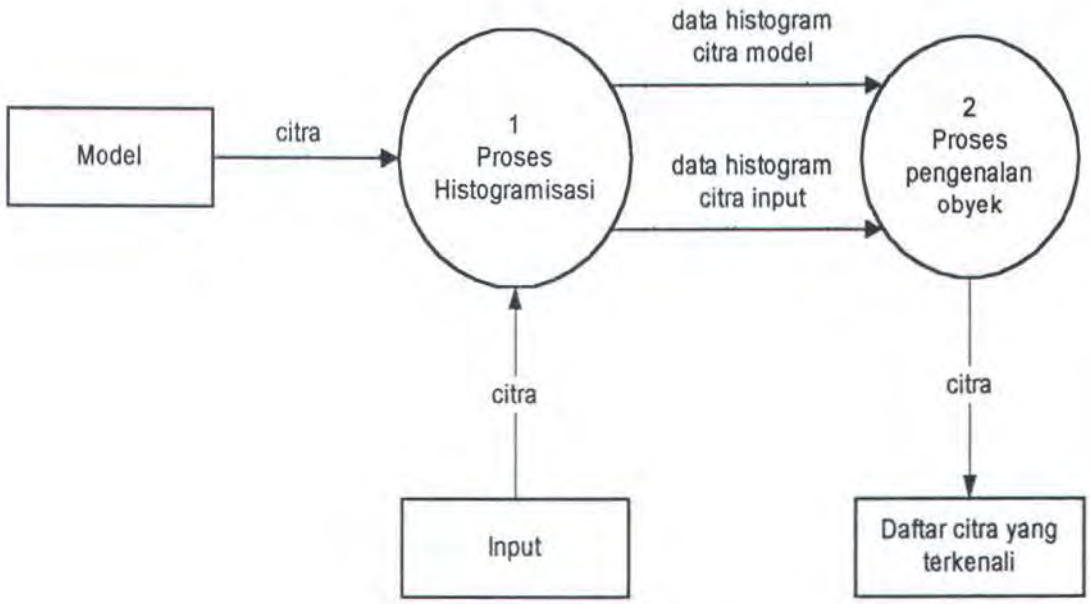
Susunan *data flow diagram* untuk perangkat lunak yang dibuat sebagaimana digambarkan berikut ini :



Gambar 14. DFD level 0

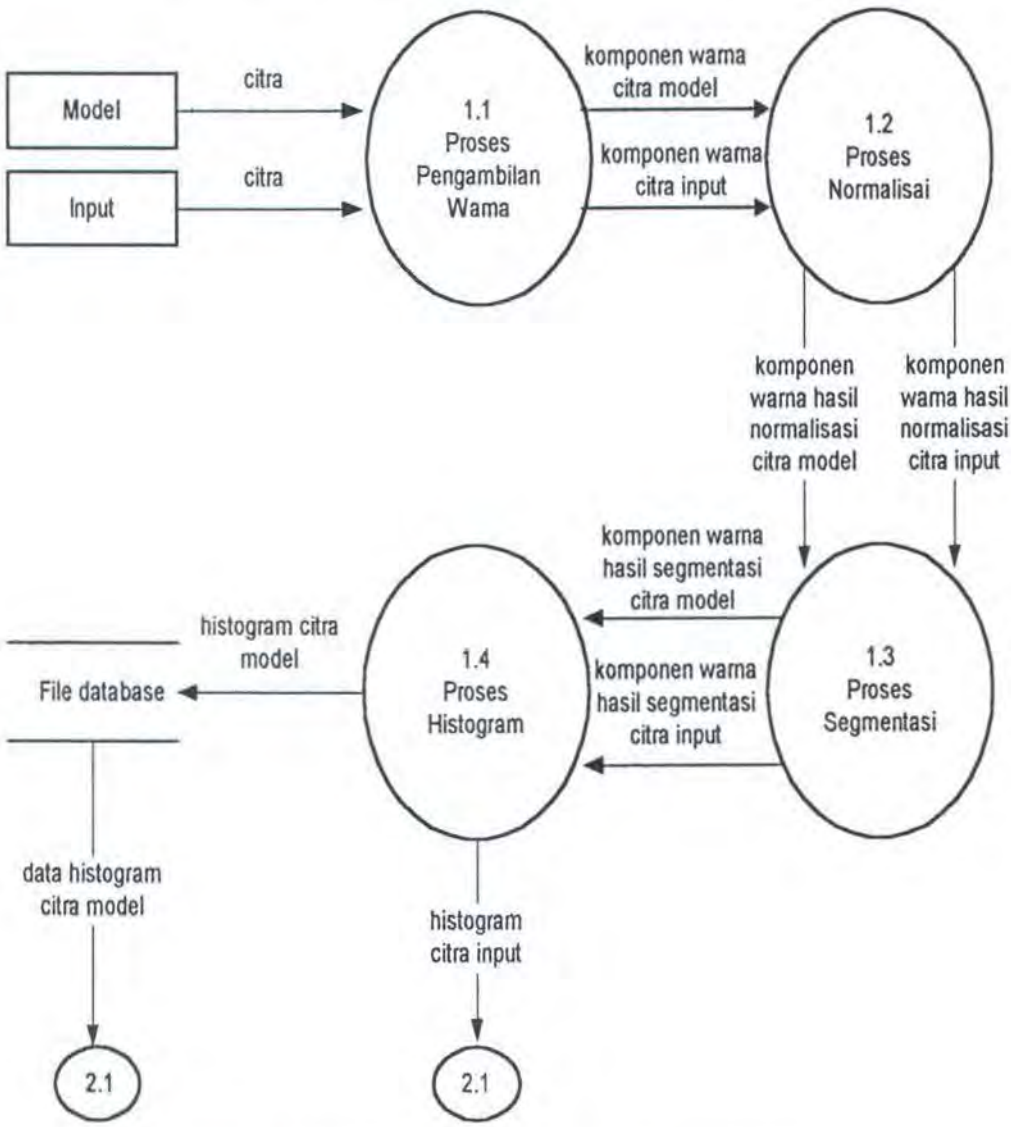
Pada DFD level 0 di atas, dapat dilihat bahwa sistem yang dibuat mempunyai dua buah input citra. Yang pertama, citra model yakni berupa beberapa citra yang dikumpulkan sebagai model dan sebuah citra yang dicari berisi obyek yang dicari. Adapun output dari sistem ini berupa citra yang telah berurutan dari besar ke kecil berdasarkan nilai hasil proses irisan histogram antara histogram citra input dengan

histogram beberapa citra model. Nilai terbesar menunjukkan tingkat kesamaan pengenalan yang paling baik dibandingkan dengan nilai yang lebih kecil.



Gambar 15. DFD level 1

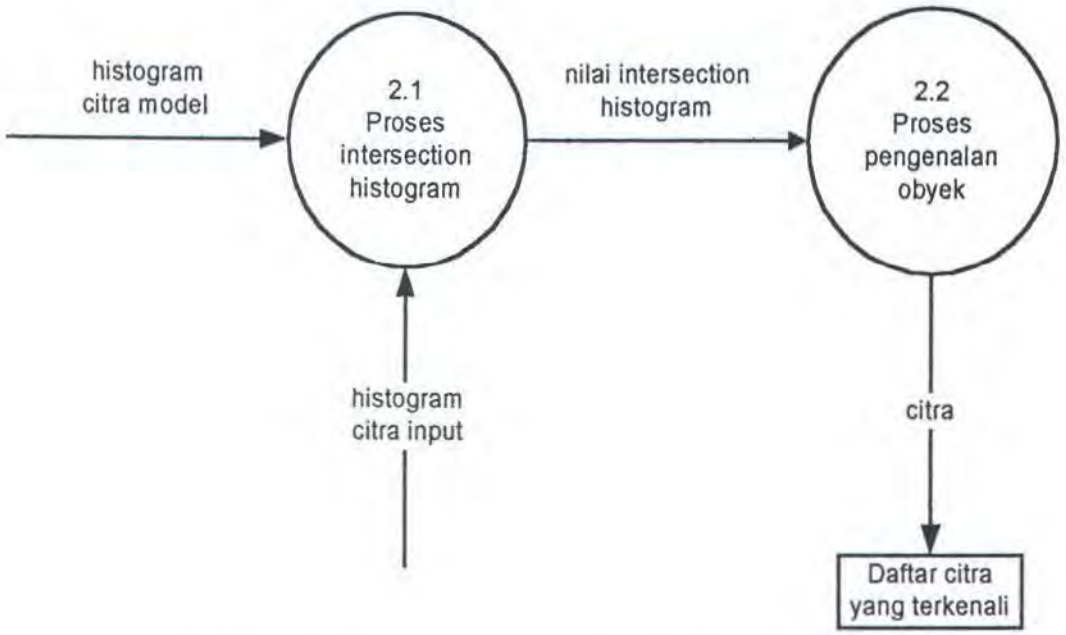
Pada DFD level 1 dapat dilihat terdapat dua buah proses utama yakni proses histogramisasi dan proses pengenalan obyek. Pada proses histogramisasi antara citra model dan citra input melakukan proses yang sama dan masing-masing akan menghasilkan data histogram. Dua buah histogram tersebut masuk ke dalam proses pengenalan obyek untuk mendapatkan citra output.



Gambar 16. DFD level 2 proses histogramisasi

Pada DFD level 2, pada hasil proses histogram antara citra model dan citra input terdapat perbedaan yakni pada media penyimpanan hasil proses. Pada citra model, hasil histogram disimpan dalam bentuk *file database*. Sedangkan pada hasil histogram citra input tidak disimpan dalam bentuk *file*, melainkan sebuah variabel

biasa. Jadi yang dihasilkan pada proses histogramisasi berupa histogram komponen warna yang telah melalui proses segmentasi.



Gambar 17. DFD level 2 proses pengenalan obyek

Pada DFD level 2, pada proses pengenalan obyek terdiri dari dua buah proses utama, yakni proses irisan histogram (*intersection histogram*) dan proses pencarian nilai terbesar. Proses irisan histogram bertujuan untuk melakukan irisan histogram antara data histogram citra model yang telah tersimpan dalam suatu *file database* dengan data histogram citra input. Dari hasil proses tersebut dicari nilai terbesar dari beberapa nilai irisan tersebut. Kemudian dilakukan proses pengurutan citra-citra yang terkenali lewat nilai irisan histogram.

5.3 Pembuatan Struktur Data

Perangkat lunak pengenalan obyek yang dibangun ini memiliki beberapa struktur data, yang masing-masing mempunyai kegunaan sendiri.

- Untuk merepresentasikan informasi yang diambil ketika proses pengambilan warna

```
Type  RGBVal      = array[word] of TRGBTriple;
      PRGBValue   = ^RGBVal;
```

RGBVal merupakan sebuah *array* yang menyimpan tiga buah komponen warna RGB. PRGBValue adalah sebuah pointer penunjuk ke RGBVal.

- Untuk merepresentasikan informasi yang diambil ketika proses normalisasi warna

```
Type  MyColorValue = record
                       red, green, blue : real;
                     end;
```

MyColorValue merupakan sebuah *record* yang terdiri dari tiga buah komponen warna

- Untuk merepresentasikan informasi yang diambil ketika proses segmentasi

```
Type  MyLaplaceValue = record
                         red, green, blue : real;
                       end;
```

MyLaplaceValue merupakan sebuah *record* yang menyimpan informasi mengenai ketiga warna yang telah dikalikan dengan matriks Laplace, 3x3 yang didefiniskan sebagai sebuah konstanta

```
const LaplaceMatrix : array [1..3,1..3] of integer =
                                        ((0,1, 0),
                                        (1,-4,1),
                                        (0,1, 0));
```

- Untuk merepresentasikan informasi yang diambil ketika proses pengindeksan warna pada citra

```
Type  RIndexValue = packed record
                     value : extended;
                     count : integer;
                    end;
      ValueColor = packed record
                     limitRed       : integer;
                     limitGreen     : integer;
                     limitBlue      : integer;
                     red  : array [1..21000] of RIndexValue;
                     green: array [1..21000] of RIndexValue;
                     blue : array [1..21000] of RIndexValue;
                    end;
      IndexValue = packed record
                     filename : array [1..15] of string[50];
                     values   : array [1..15] of ValueColor;
                    end;
```

IndexValue merupakan sebuah *record* yang menyimpan informasi mengenai nama *file* tiap citra yang disimpan data histogramnya, filename. Values

merupakan sebuah record yang berisi informasi jumlah maksimum komponen warna setiap citra (limitRed, limitGreen, limitBlue) dan informasi indeks tiap komponen warna baik mengenai nilainya (value) atau jumlah maksimal (count). Banyaknya citra yang bisa ditampung dalam *record* IndexValue maksimal sebanyak 15 buah. Array komponen warna bernilai 1 sampai 21000 merupakan jumlah maksimal piksel tiap citra (150 x140 piksel).

- Untuk merepresentasikan informasi tiap citra dalam *database* model

```
Type  InfoDataType =  record
            MyRGBValue: array of array of PalleteRGB;
            MyLaplaceValue:array of array of MyColorValue;
            MyNormalValue :array of array of MyColorValue;
            countJmax,countJmin : integer;
            countImax,countImin : integer;
            end;
```

InfoDataType merupakan sebuah *record* yang menyimpan informasi tiap citra setiap melakukan proses utama histogramisasi, yang terdiri dari proses pengambilan warna (MyRGBValue), proses normalisai warna (MyNormalValue) dan proses segmentasi (MyLaplaceValue). Batas koordinat piksel tiap citra disimpan dalam variabel countJmax, countJmin, countImax dan countImin.

5.4 Pembuatan Perangkat Lunak

Perangkat lunak pengenalan obyek yang dibangun ini mememiliki beberapa *form* dan proses, yang masing-masing mempunyai kegunaan sendiri.

- Database model

  Merupakan *form* yang berguna untuk membuat kumpulan beberapa citra sebagai model citra. Data yang dihasilkan keseluruhan citra disimpan dalam bentuk *file of record (database)*. Struktur data untuk *file* tersebut :

```
var ModelFile                    : file of IndexValue;
```

  Pada *form* ini, terjadi proses utama histogramisasi yang didefinisikan dalam beberapa *procedure* di bawah ini :

```
procedure TakeRGB(Data1: InfoDataType ; bitmap:Tbitmap);
procedure Normalisasi (Data1:InfoDataType;
                       posX,posY:integer);
procedure LaplacianImage(Data1:InfoDataType;
                       posX,posY,
                       countXMin,countXMax,
                       countYMin,countYMax : integer);
procedure MakeIndex    (Data1 : InfoDataType;
                       posX, posY : integer);
```

- Citra masukan

  Merupakan *form* yang berguna untuk menampilkan citra yang akan dikenali obyek di dalamnya. Data yang dihasilkan citra ini disimpan dalam bentuk variabel *record*.

```
var RecordInput    : IndexValue;
```

Pada *form* ini, terjadi proses utama histogramisasi yang didefinisikan dalam beberapa *procedure* di bawah ini :

```
procedure MakeRGBValueArray(Data1:InfoDataType;
                        maxJ,maxI:integer);
procedure MakeNormalValueArray(Data1:InfoDataType;
                        maxJ,maxI:integer);
procedure MakeLaplaceArray(Data1:InfoDataType;
                        maxJ,maxI:integer);
procedure TakeRGB(Data1: InfoDataType ; bitmap:Tbitmap);
procedure Normalisasi(Data1:InfoDataType;
                        posX,posY:integer);
procedure LaplacianImage(Data1:InfoDataType;
                        posX,posY,
                        countXMin,countXMax,
                        countYMin,countYMax : integer);
procedure MakeIndex(Data1 : InfoDataType;
                        posX, posY : integer);
procedure IntersectionHistogram;
procedure SortResult;
```

- Hasil proses

  Merupakan *form* yang berguna untuk menampilkan hasil proses pengenalan. Terdiri dari keseluruhan citra pada database model yang telah diurutkan berdasarkan hasil perhitungan antara citra masukan dengan keseluruhan citra pada database model. Struktur data untuk menyimpan informasi di atas dalam bentuk *array* yang berisi *record* untuk tiap citra hasil :

```
Type  ResultRecord = record
                        value : extended;
                     filename : string;
                      end;
var   MyResult     : array of ResultRecord;
```

  Pada *form* ini, terjadi proses penampilan citra hasil :

```
procedure ViewResult;
```

- Proses pengambilan warna

  Proses ini mempunyai *procedure* yang didefinisikan di bawah ini :

```
procedure TakeRGB(Data1: InfoDataType;bitmap : Tbitmap);
var P : PRGBValue;
begin
for y := 0 to bitmap.height-1 do
   begin
     P :=  bitmap.scanline[y];
     for x := 0 to bitmap.width-1 do
     begin
       with P^[x],Data1 do
```

```
        begin
          MyRGBValue[x,y].biblue     := rgbtblue;
          MyRGBValue[x,y].bigreen    := rgbtgreen;
          MyRGBValue[x,y].bired      :=  rgbtred;
        end;
    end;
end;
```

Setiap piksel diambil informasi mengenai ketiga komponen warnanya dan disimpan dalam array MyRGBValue.

- Proses normalisasi

Proses ini mempunyai *procedure* yang didefinisikan di bawah ini :

```
procedure Normalisasi (Data1: InfoDataType;
                       posX,posY:  integer);
begin
   with Data1.MyRGBValue[posX,posY] do
   begin
     if bired = 0 then
       MyColorValue[posX,posY].red := unlimited
     else
       MyColorValue[posX,posY].red :=  ln(bired) ;
     if bigreen = 0 then
       MyColorValue[posX,posY].green := unlimited
     else
       MyColorValue[posX,posY].green := ln(bigreen);
     if biblue = 0 then
       MyColorValue[posX,posY].blue := unlimited
     else
       MyColorValue[posX,posY].blue :=  ln(biblue);
   end;
```

```
end;
```

Data yang tersimpan pada *array* MyRGBValue dilakukan proses normalisasi dan hasilnya disimpan dalam array MyColorValue.

- Proses segmentasi

Proses ini mempunyai *procedure* yang didefinisikan di bawah ini :

```
procedure LaplacianImage(Data1: InfoDataType;
                         posX,posY,countXMin,countXMax,
                         countYMin,countYMax : integer);
var  Kanan, Atas, Tengah, Bawah, Kiri : real;
begin
 with Data1 do
 begin
  if (posX-1 < countXMin) then
    Kiri:= 0
  else Kiri := MyColorValue[posX-1,posY]*
                  LaplaceMatrix[1,2];
  if (posY-1 < countYMin) then
    Atas := 0
  else Atas := MyColorValue[posX,posY1]*
               LaplaceMatrix[2,1];
  Tengah := MyColorValue[posX, posY]*
             LaplaceMatrix[2,2];
  if (posY+1 > countYMax) then
    Bawah := 0
  else Bawah:= MyyColorValue[posX,posY+1]*
                  LaplaceMatrix[2,3];
  if (posX+1 > countXMax) then
    Kanan := 0
  else Kanan:=MyColorValue[posX+1, posY]*
```

```
                LaplaceMatrix[3,2];
    MyLaplaceValue[posX,posY]:=
                Kiri+Atas+Tengah+Bawah+Kanan;
  end;
end;
```

Data yang tersimpan pada *array* MyColorValue dikalikan matriks Laplace 3x3 yang telah didefinisikan sebelumnya. Hasil dari proses ini disimpan dalam *array* MyLaplaceValue.

- Proses pengindeksan

Proses ini mempunyai program yang didefinisikan di bawah ini :

```
with RecordModel.Values[modelCount],
        MyLaplaceValue[posX,posY] do
begin
  if (posX = 0) and (posY = 0) then
  begin
    red[counter].count := 1;
    red[counter].value := red;
    limitRed := 1;
  end
  else
  begin
    match := false;
    for i := 1 to (limitRed) do
    begin
      if red[i].value  = red then
      begin
        match := true;
        inc(red[i].count );
        break;
```

```
      end;
    end;
    if match = false then
    begin
      inc(limitRed);
      red[limitRed].value  := red;
      red[limitRed].count := 1;
    end;
      .
      .
      .
end;
```

Pada proses ini dilakukan pengindeksan data yang tersimpan dalam *array* MyLaplaceValue, dengan membandingkan pada keseluruhan piksel pada citra tersebut. Hasil proses ini disimpan dalam *record* RecordModel.

- Proses irisan histogram

Proses ini mempunyai program yang didefinisikan di bawah ini :

```
procedure IntersectionHistogram(indeks : integer);
var match : boolean;
    i,j,m: integer;
    redInter, greenInter, blueInter : real;
    tempRed,tempGreen,tempBlue,tempCum: double;
begin
  tempRed:=0;tempGreen := 0; tempBlue := 0;tempCum := 0;
  match := false;
  m :=0;
  with RecordModel.values[indeks] do
  begin
    for j := 1 to RecordInput.values[1].limitRed do
```

```
begin
   for i := 1 to limitRed do
   begin
      if RecordInput.Values[1].red[j].value=
                              red[i].value then
       begin
         match := true;
         m := i;
         break;
       end;
   end;
   if match = true then
   begin
   if Red[m].count = 0 then
    redInter := 0
   else
   redInter:= min(RecordInput.Values[1].red[j].count,
                  red[m].count)/ Red[m].count;
   end
   else
    redInter := 0;
  tempRed := tempRed + redInter;
end;//j
.
.
.
.
with MyResult[indeks-1] do
begin
  value := tempCum;
  filename:= RecordModel1.filename[indeks];
end;
end;
```

Pada proses ini dilakukan proses irisan histogram (*intersection histogram*) antara data yang tersimpan dalam *record* RecordInput untuk citra yang dicari obyeknya dengan data citra model yang tersimpan dalam *recordModel*. Hasil dari proses ini disimpan dalam *array* MyResult.

- Proses pengurutan

Proses ini mempunyai *procedure* yang didefinisikan di bawah ini :

```
procedure SortResult;
var i,j : integer;
    swap : ResultRecord;
begin
  for i := 0 to High(MyResult)-1 do
  begin
    for j := 0 to High(MyResult)-1 do
    begin
      if MyResult[j].value < MyResult[j+1].value then
      begin
        swap := MyResult[j];
        MyResult[j] := MyResult[j+1];
        MyResult[j+1] := Swap;
      end;
    end;
  end;
end;
```

Pada proses ini dilakukan proses pengurutan citra hasil pada *array* MyResult berdasarkan nilai terbesar.

BAB VI

HASIL UJI COBA PERANGKAT LUNAK

BAB VI

HASIL UJI COBA PERANGKAT LUNAK

Pada bab ini akan dijelaskan mengenai hasil uji coba yang dilakukan terhadap perangkat lunak yang telah dibuat , yang berguna sebagai bahan evaluasi dalam perbaikan sistem. Hasil evaluasi juga diperlukan bagi pengembangan sistem pengenalan lebih lanjut sehingga didapatkan sistem pengenalan yang mempunyai kemampuan yang lebih baik dan membutuhkan waktu komputasi yang lebih singkat.

Kemampuan sistem pengenalan ini didasarkan pada beberapa tolok ukur. Tolok ukur yang bisa digunakan dalam pengenalan obyek ini antara lain :

1. Keberhasilan perangkat lunak dalam mengenali obyek dalam citra dengan merujuk pada *database* model yang ada.
2. Pengaruh pencahayaan yang berubah-ubah terhadap hasil pengenalan obyek yang dilakukan.

6.1 Sistem yang Digunakan

Pada uji coba ini dilakukan pada komputer dengan spesifikasi :

- Processor : Pentium II 400KHz
- Memory : 64 MByte
- Operating System : Windows NT 4.0

6.2 Uji coba

Uji coba pada perangkat lunak ini dibedakan menjadi dua bagian, yakni ujicoba keakuratan perangkat lunak dalam mengenali sebuah obyek terhadap database model beberapa obyek yang telah ditentukan dan ujicoba pengaruh pencahayaan terhadap hasil perangkat lunak

1. Uji coba Keakuratan

Analisa yang diambil merupakan ukuran ketepatan citra yang mempunyai obyek yang sama dengan citra-citra pada model. Parameter yang dipakai untuk menentukan sebuah obyek itu mempunyai ketepatan yakni obyek tersebut berada pada ranking atau urutan antara 1 sampai 3 dari citra model yang mempunyai nilai terbesar setelah dilakukan proses irisan histogram.

Beberapa citra yang dijadikan model :

				
Ajaxback	*Angelsoft*	*Ball1*	*Biclean*	*Chickensoup*
				
Charmin	*Cofee_1*	*Frankenberry2*	*Crunchberries*	*Whitecloud1*

Gambar 18. Model citra uji coba

				
Ajaxside	*Angelsoft2*	*Ball5*	*Biclean2*	*Chickensoup2*
				
Charmin2	*Cofee1*	*Frankenberry*	*Crunchberries2*	*Whitecloud2*

Gambar 19. Beberapa input uji coba

Dengan melalui proses pengenalan tercatat : 9 dari 10 citra yang dikenali (90%).

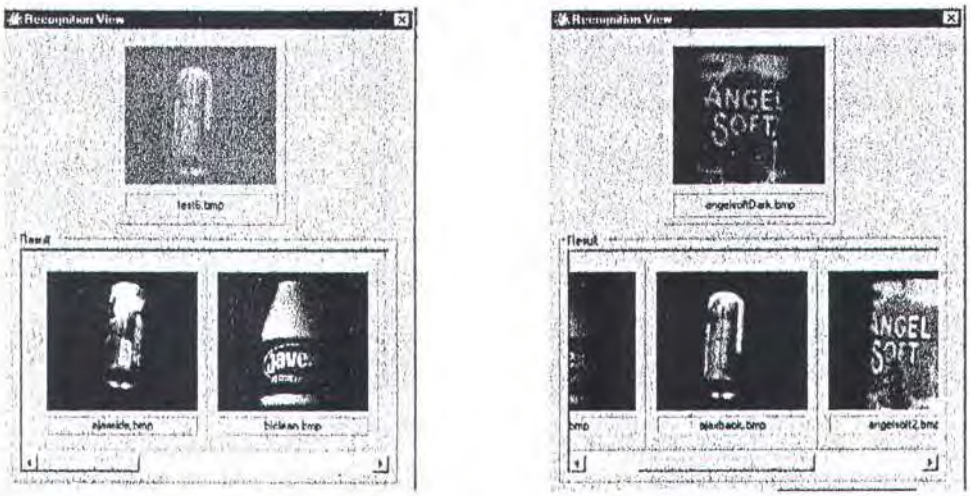
Dari hasil percobaan tersebut terdapat 1 buah citra yang tidak dikenali atau kurang tepat dalam proses pengenalan.

citra model	citra input
	
Whitecloud1	whitecloud2

Gambar 20. Obyek yang tidak dikenali

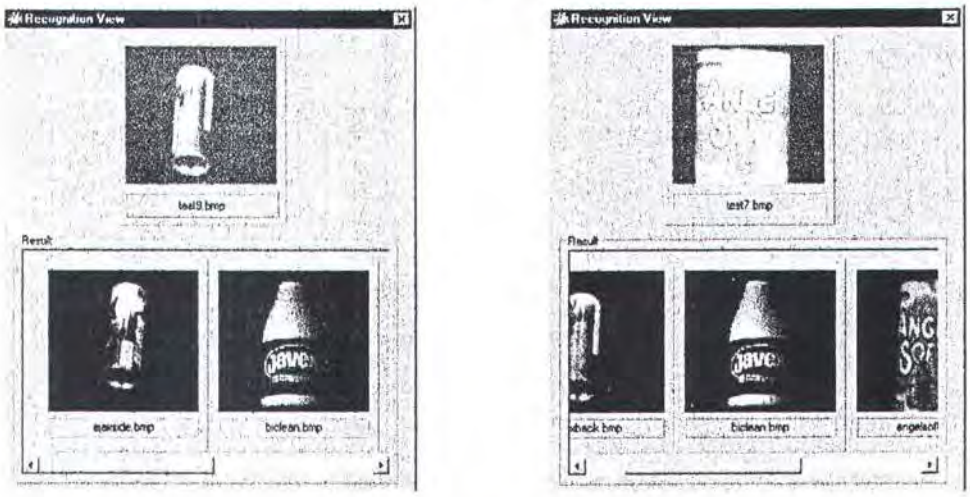
2. *Uji coba terhadap pengaruh pencahayaan*

a. Perubahan intensitas pencahayaan yang lebih gelap



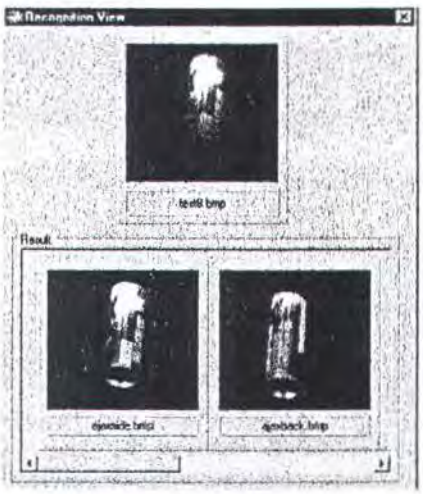
Gambar 21. Perangkat lunak mengenali obyek yang diubah intensitas pencahayaannya lebih gelap

b. Perubahan intensitas pencahayaan yang lebih terang



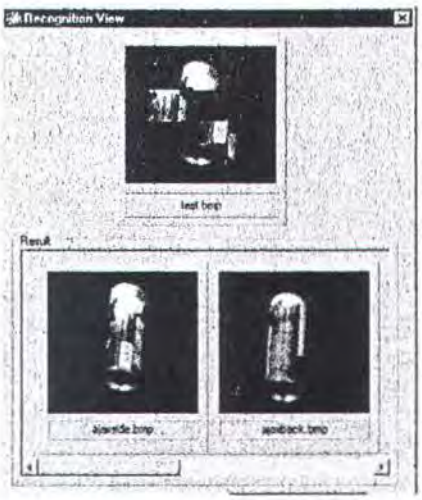
Gambar 22. Perangkat lunak mengenali obyek yang diubah intensitas pencahayaannya lebih terang

c. Perubahan titik sumber pencahayaan



Gambar 23. Perangkat lunak mengenali obyek yang diubah titik pencahayaannya

d. Perubahan pada obyek (obyek dipecah)



Gambar 24. Perangkat lunak mengenali obyek yang di pecah

e. Perubahan pada obyek (*mirror*)

Gambar 25. Perangkat lunak mengenali obyek yang telah mengalami proses mirror

BAB VII

KESIMPULAN DAN SARAN

BAB VII

KESIMPULAN DAN SARAN

Bab ini menguraikan beberapa hal yang dapat dijadikan kesimpulan dari beberapa percobaan yang telah dilakukan. Selain itu, bab ini memberikan beberapa saran yang dapat diterapkan dalam pengembangan sistem pengenalan obyek. Pengembangan sistem dapat dilakukan pada pemilihan struktur data yang lebih baik dan perancangan proses yang lebih baik, dimana secara prinsip pengembangan sistem bertujuan untuk meningkatkan kemampuan sistem dalam melakukan pengenalan obyek dan meminimalkan waktu komputasi.

7.1 Kesimpulan

Dari beberapa hasil uji coba perangkat lunak, penulis dapat mengambil kesimpulan sebagai berikut :

- Pencahayaan yang mengenai sebuah obyek tidak akan berpengaruh pada kemampuan proses pengenalan, selama pencahayaan tersebut tidak terlalu terang ataupun terlalu gelap. Karena apabila pencahayaan terlalu terang maka warna permukaan obyek akan mendekati warna putih dan apabila pencahayaan terlalu gelap maka warna permukaan obyek akan menjadi tidak kelihatan jelas.

- Metode color constant color indexing cukup baik digunakan untuk pengenalan obyek, selama obyek tidak mengalami perubahan komposisi warna yang sangat jauh. Misal obyek mengalami perubahan skala yang cukup besar ataupun warna permukaan obyek antara sisi-sisinya yang berlainan.

7.2 Saran

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

- Citra model dan citra yang dicari obyeknya dibatasi pada ukuran yang sama, sehingga memungkinkan dikembangkan pada ukuran citra yang berbeda
- File format citra yang dikembangkan hanya terbatas pada *file* bitmap (*.bmp), sehingga memungkinkan untuk pengembangan selanjutnya dengan menggunakan format citra yang lainnya
- Perangkat lunak pada tugas akhir ini hanya dapat menerima obyek-obyek dengan latar belakang hitam saja saja. Sehingga memungkinkan dalam hal pengembangan lebih lanjut dengan menggunakan latar belakang yang lebih beragam.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

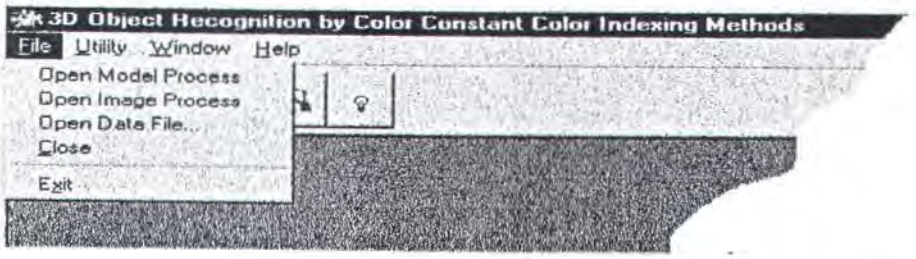
1. Brian Funt, Kobus Barnard and Lindsay Martin, *"Is Machine Colour Constancy Good Enough ?"* , In *5th European Conference on Computer Vision*, pages 455–459. Springer, June 1998.
2. Brian V. Funt, Vlad Cardei, and Kobus Barnard, *"Learning Color Constancy"* . In *Proceedings of the Fourth Color Imaging Conference*, pages 58–60, November 1996.
3. D. Hearn and M.P Baker, *Computer Graphics,* Second Edition, Prentice-Hall International, Inc., 1994
4. David Slater and Glenn Healey, *"The Illumination-Invariant Recognition of 3D Object Using Local Color Invariants", IEEE Transaction on Pattern Analysis and Machine Intelligence,* Vol. 18, No.2, February 1996
5. G. D. Finlayson, *"Color in Perspective". IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1034–1038, 1996.
6. G. D. Finlayson, *"Colour Object Recognition (MSc Thesis)".* University of Strathclyde Department of Computer Science , 1989
7. M.J. Swain, *"Color Indexing (PhD Thesis)"*, University of Rochester Department of Computer Science, 1990.
8. Rafael C. Gonzales and Richard E. Woods, *"Digital Image Processing",* Second Edition, Addison Wesley Publishing Company, 1993
9. Ronald Alferez and Yuan-Fang Wang, *"Geometric and Illumination Invariants for Object Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No.6, June1996

LAMPIRAN

LAMPIRAN

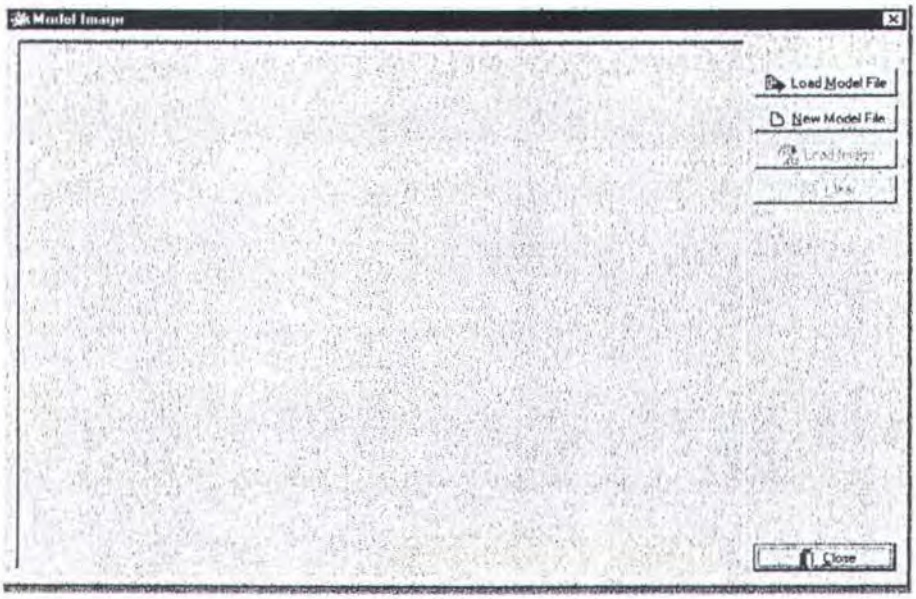
USER GUIDE

Aplikasi ini terdiri dari beberapa menu utama, seperti yang terlihat di bawah ini :

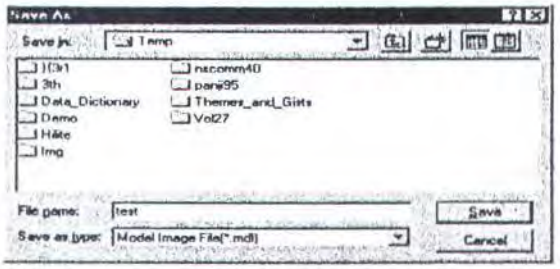


Untuk dapat menjalankan aplikasi ini, langkah-langkah yang harus dilakukan sebagai berikut :

Langkah I : Menyiapkan sebuah model yang terdiri dari beberapa citra obyek sebagai pembanding, dengan cara memilih menu **File** lalu memilih submenu **Open Model Process** atau dapat dilakukan dengan cara melakukan *click* pada *toolbar*  , sehingga akan tampil sebuah *form* seperti di bawah ini :

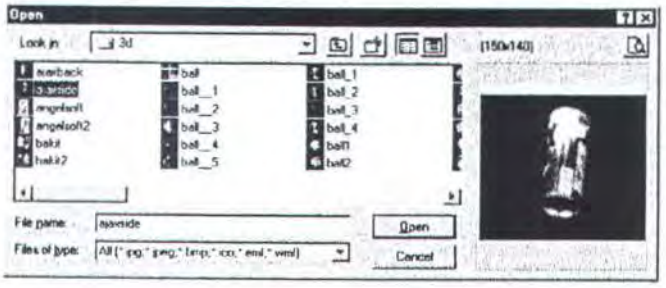


Langkah II : Menyiapkan sebuah *file* untuk penyimpanan data masing-masing citra obyek yang ada pada model, dengan cara melakukan *click* pada tombol New Model File, sehingga akan tampil sebuah dialog seperti di bawah ini :



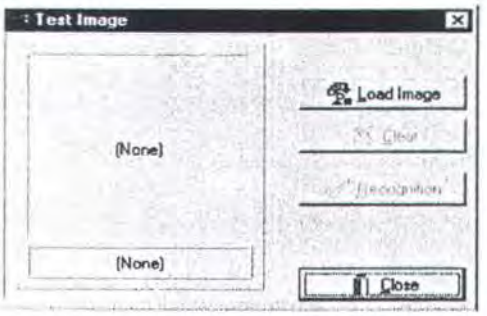
Ketika muncul dialog di atas, kita harus memasukkan nama *file* yang akan kita jadikan sebagai *file* penyimpanan model tersebut di direktori yang kita inginkan.

Langkah III : Memasukkan citra obyek satu persatu pada *form* Model, dengan cara melakukan *click* pada tombol Load Image, sehingga akan keluar dialog seperti di bawah ini :



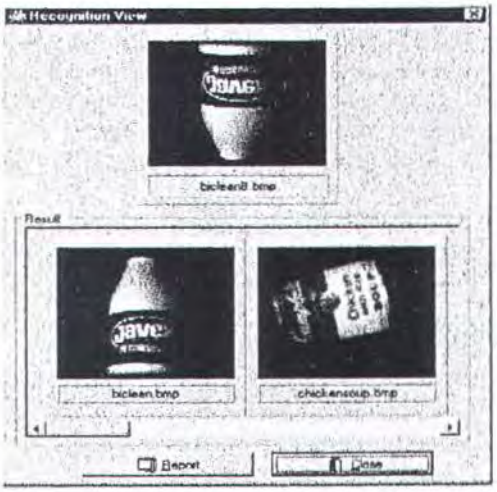
Hingga telah terkumpul beberapa citra obyek pada *form* model sesuai yang kita inginkan. Misalnya seperti yang terlihat pada gambar di bawah ini :

Langkah IV : Persiapan proses pengenalan obyek, dengan cara memilih menu **File** dan memilih submenu **Open Image Process** atau melakukan *click* pada *toolbar* | | , sehingga akan tampil sebuah *form* seperti di bawah ini :

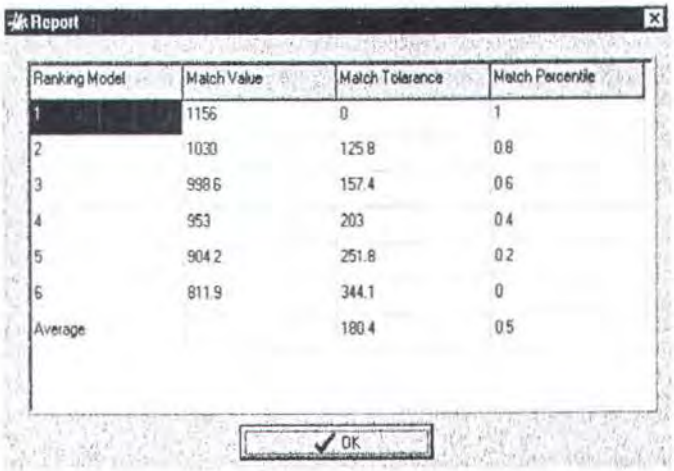


Dengan menggunakan tombol Load Image, kita dapat memasukkan citra obyek yang ingin kita kenali.

Langkah V: Proses pengenalan obyek, dengan cara melakukan *click* pada tombol Recognition maka akan tampil hasil proses pengenalan berupa daftar terurut obyek-obyek yang terkenali, dengan urutan dari nilai terbesar sampai nilai yang terkecil tingkat kesamaannya.



Langkah VI : Mengetahui nilai tiap citra, dengan cara melakukan *click* pada tombol Report, sehingga akan keluar sebuah form seperti di bawah ini:



Pada aplikasi ini terdapat fasilitas *configuration* yang dapat digunakan dengan cara memilih menu **Utility** dan memilih submenu **Configuration...** atau melakukan *click* pada *toolbar* , sehingga akan keluar sebuah *form* yang terdiri dari :

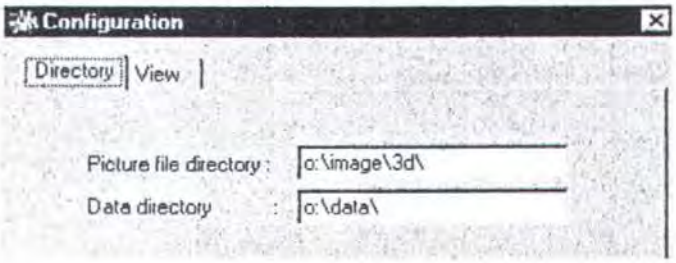
a. *Form Directory*, yang terdiri dari

1. *Picture file directory*

   Berfungsi untuk menentukan letak direktori *file* citra yang akan kita gunakan pada aplikasi ini.

2. *Data directory*

   Berfungsi untuk menentukan letak direktori penyimpanan *file* model.



b. *Form View*

Berfungsi untuk menampilkan sebuah *report* untuk setiap proses yang ada pada aplikasi ini.