PENENTUAN EFEK PENCAHAYAAN PADA PERMUKAAN DENGAN ALGORITMA PHOTOMETRIC STEREO

TUGAS AKHIR

Oleh :

I WAYAN YUDI MULYAWAN

2695 100 022

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2001

PENENTUAN EFEK PENCAHAYAAN PADA PERMUKAAN DENGAN ALGORITMA PHOTOMETRIC STEREO

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

Ir. Esther Hanaya, M.Sc.
NIP. 130 816 212

DEPARTEMEN PENDIDIKAN NASIONAL
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK INFORMATIKA

Dosen Pembimbing II

Rully Soelaiman, S.Kom.
NIP. 132 086 802

SURABAYA
Januari, 2001

ABSTRAK

ABSTRAK

Salah satu hal yang mempengaruhi interpretasi terhadap objek 3D adalah penambahan efek pencahayaan terhadap citra dalam format *gray level*, yaitu pemberian variasi intensitas cahaya dari suatu titik ke titik lainnya. Variasi intensitas cahaya pada tiap-tiap titik pada permukaan dipengaruhi oleh beberapa faktor, yaitu intensitas sumber cahaya, koefisien refleksi bahan permukaan dan sudut datangnya cahaya terhadap normal permukaan.

Untuk menambahkan efek pencahayaan pada sebuah citra, dalam pembuatan tugas akhir ini digunakan sebuah algoritma yang disebut dengan algoritma *Photometric Stereo*. Dengan menggunakan tiga buah gambar dari objek yang sama yang masing-masing diberikan cahaya dari tiga arah yang berbeda (masing-masing sumber cahaya diasumsikan mempunyai intensitas yang sama), algoritma ini akan menghitung vektor normal masing-masing titik pada permukaan, kemudian dengan menggunakan informasi vektor normal tersebut, akan dapat ditentukan gambar objek tersebut jika diberikan cahaya dari arah tertentu.

Objek yang digunakan sebagai masukan dalam pembuatan tugas akhir ini dibatasi pada objek yang tiap elemen pada permukaanya terdiri dari bahan yang sama, sehingga koefisien refleksi tiap titik pada permukaan tidak menimbulkan variasi dalam memantulkan cahaya yang diterima oleh masing-masing titik pada permukaan objek.

KATA PENGANTAR

KATA PENGANTAR

Puji syukur kehadapan Tuhan Yang Maha Esa atas selesainya penulisan tugas akhir yang berjudul :

PENENTUAN EFEK PENCAHAYAAN DENGAN

ALGORITMA "PHOTOMETRIC STEREO"

Terima kasih juga saya sampaikan kepada semua pihak yang telah ikut memberikan masukan, saran dan juga kritik untuk penulisan tugas akhir ini.

Rasa terima kasih ini saya berikan dengan tulus kepada :

1. Kedua orang tua penulis yang selalu memberikan dorongan baik lahir maupun batin selama pengerjaan tugas akhir ini.
2. Ir Esther Hanaya MSc. selaku dosen pembimbing I dan juga atas bimbingannya selama pengerjaan tugas akhir ini.
3. Rully Soelaiman S.Kom selaku dosen pembimbing II yang telah memberikan banyak bahan masukan terutama terkait dengan jurnal dan buku-buku literatur yang dipinjamkan olehnya.
4. Ir. A. Holil N.A selaku dosen wali penulis selama kuliah di jurusan Teknik Informatika Institut Teknologi Sepuluh November Surabaya.
5. Angela Christy yang telah banyak memberikan dorongan semangat selama pengerjaan tugas akhir ini, juga atas saran dan masukannya serta pinjamannya untuk pembelian Hardisk penulis yang rusak sewaktu pengerjaan tugas akhir ini.
6. Jeffry Erensano '*sapy*' yang atas saran dan kritiknya, pinjaman mobilnya serta atas berbagai *gangguannya* dan *terornya* selama pengerjaan tugas akhir ini.

7. Oka Sunarya atas pinjamannya untuk pembelian monitor penulis yang terbakar pada waktu pengerjaan tugas akhir ini dan juga atas *ultraflu*-nya.
8. Putu A. Widhirtha, Komang Rahyuni, Sonny, Gede, Diory, Royke, Teddy, Niko, Andreas, Ramhot, Bang Ucok, Dessy, Ronny, Hendi, Aswin atas dukungannya dan telah menjadi sahabat-sahabat yang baik selama penulis kuliah di jurusan Teknik Informatika. Rai, Bu Kurnia serta bapak-bapak Pendeta yang sering menemani penulis sembahyang di Pura.
9. Seluruh TC angkatan '95 yang telah menjadi teman-teman yang menyenangkan selama penulis menjalani kuliah di Jurusan Teknik Informatika.

Tiada kata yang dapat terucapkan untuk mengungkapkan rasa terima kasih yang begitu besar sehingga penulisan tugas akhir ini dapat diselesaikan untuk memperoleh gelar sarjana.

Selama pengerjaan tugas akhir ini, saya menyadari bahwa karya tulis ini masih jauh dari sempurna, sehingga tidak menutup kemungkinan untuk dikembangkan dan diperbaiki lebih lanjut oleh generasi mendatang.

Penulis,

I Wayan Yudi Mulyawan

DAFTAR ISI

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

DAFTAR TABEL

Halaman

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Otak manusia memiliki kemampuan yang luar biasa untuk melihat dunia tiga dimensi dari objek dua dimensi yang diproyeksikan pada retina. Bagaimana hal ini dilakukan, masih belum dapat dipahami sepenuhnya dan hal ini masih menjadi topik yang aktif dalam ilmu pengetahuan tentang syaraf dan komunitas indera penglihatan. Komunitas indera penglihatan adalah sekumpulan syaraf yang saling berkomunikasi untuk menyampaikan gambar yang ditangkap oleh retina ke otak. Hal yang telah dapat dipahami adalah faktor–faktor yang mempengaruhi pembentukan interpretasi 3D.

Pada tugas akhir ini akan dibahas tentang pembentukan persepsi 3D dari suatu gambar 2D dengan hanya menggunakan efek pencahayaan. Yaitu dengan menggunakan variasi intensitas cahaya dari suatu titik ke titik lainnya pada suatu objek citra. Variasi ini akan memberikan informasi tentang bentuk karena intensitas cahaya yang direfleksikan oleh suatu permukaan tergantung pada orientasinya (normal dari permukaan) secara relatif terhadap sumber cahaya. Jadi karena bahan permukaan tidak menimbulkan variasi bayangan, maka variasi intensitas cahaya hanya mungkin terjadi apabila terdapat perubahan pada orientasi normal permukaan dan karena itulah dihasilkan informasi yang kuat tentang bentuk.

1.2 Tujuan dan Manfaat

Tujuan dari penulisan tugas akhir ini adalah menerapkan algoritma "*Photometric Stereo*" untuk mendapatkan arah vektor normal tiap-tiap titik pada permukaan suatu objek sehingga dapat diketahui arah cahaya yang dipantulkan oleh titik-titik tersebut jika objek diberikan sinar dari sebuah sumber cahaya yang berada pada posisi tertentu. Dengan demikian akan dapat dibuat gambar permukaan yang akan ditangkap oleh kamera, jika objek mendapatkan sinar dari sebuah sumber cahaya dari suatu arah tertentu.

1.3 Permasalahan

Permasalahan yang akan dibahas dalam pembuatan tugas akhir ini adalah bagaimana menentukan arah vektor normal tiap-tiap titik pada permukaan sebuah objek dengan menggunakan gambar citra dua dimensi sebagai masukannya. Dengan demikian akan dapat ditentukan arah refleksi cahaya pada titik-titik tersebut jika mendapatkan sinar sebuah sumber cahaya dari arah tertentu, sehingga intensitas cahaya yang dihasilkan oleh masing-masing titik pada permukaan dapat dihitung.

Salah satu algoritma untuk mendapatkan vektor normal pada permukaan sebuah objek akan dibahas dalam penulisan tugas akhir ini. Algoritma tersebut disebut dengan *Photometric Stereo*. Untuk membentuk peta vektor normal sebuah permukaan, algoritma ini memanfaatkan intensitas cahaya yang yang dipantulkan oleh permukaan objek tersebut jika mendapat cahaya dari sebuah sumber cahaya dari arah tertentu.

Algoritma *Photometric Stereo* ini memerlukan input berupa tiga buah gambar objek yang akan dihitung vektor normalnya. Masing-masing gambar tersebut merupakan

gambar permukaan sebuah objek yang mendapatkan sinar dari arah yang berbeda-beda. Dari informasi intensitas cahaya pada ketiga citra tersebut kemudian akan dicari arah vektor normalnya. Dengan menggunakan data vektor normal yang telah dihitung, maka selanjutnya akan dapat ditentukan gambar yang ditangkap oleh kamera jika objek tersebut diberi cahaya dari arah tertentu.

1.4 Batasan Masalah

Perbedaan warna dan bahan yang membentuk permukaan suatu objek dan perbedaan arah datangnya cahaya pada tiap–tiap titik pada suatu permukaan dapat menyebabkan perbedaan besarnya intensitas cahaya yang direfleksikan oleh tiap–tiap titik pada permukaan objek tersebut. Untuk objek dengan warna yang bervariasi ataupun yang permukaannya dibentuk oleh titik – titik dengan bahan yang beraneka ragam, tidak dapat diterapkan algoritma *Photometric Stereo* untuk mendapatkan normal vektor permukaan objek tersebut. Karena intensitas cahaya yang dipantulkan oleh sebuah titik pada permukaan juga sangat dipengaruhi oleh koefisien refleksi bahan pembentuknya.

Maka dari itu, dalam pembuatan tugas akhir ini, inputnya dibatasi pada objek-objek yang tiap-tiap titik pada permukaannya terdiri dari bahan yang sama dan warna yang sama pada tiap bagian permukaannya, misalnya adalah sebuah patung yang terbuat dari perunggu, objek ini mempunyai warna dan bahan yang sama pada seluruh bagian permukaanya, intensitas cahaya yang dipantulkan oleh suatu bagian permukaan hanya tergantung dari arah sumber cahaya dan arah permukaan terhadap kamera.

1.5 Sistematika Pembahasan

Penyusunan tugas akhir ini terbagi menjadi beberapa bab. Penyusunan tugas akhir ini juga meliputi pembahasan hasil percobaan dan juga deskripsi pembuatan perangkat lunak yang mengaplikasikan metode *Photometric Stereo* untuk pembentukan persepsi tentang objek 3D. Tugas akhir ini disusun sebagai berikut :

BAB I PENDAHULUAN

BAB II VEKTOR PADA RUANG DIMENSI TIGA DAN EFEK PENCAHAYAAN DENGAN OPENGL

BAB III PENCARIAN NORMAL VEKTOR PADA PERMUKAAN DENGAN MENGGUNAKAN PHOTOMETRIC STEREO

BAB IV PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK

BAB V HASIL UJICOBA PERANGKAT LUNAK

BAB VI PENUTUP

Susunan pembahasan diatas sudah mencakup semua pembahasan mulai dari abstraksi permasalahan dan metode penyelesaian, pembahasan teori dari *Photometric Stereo* hingga pembahasan hasil ujicoba perangkat lunak.

BAB II

VEKTOR PADA RUANG DIMENSI TIGA DAN EFEK PENCAHAYAAN DENGAN OPENGL

BAB II

VEKTOR PADA RUANG DIMENSI TIGA DAN EFEK PENCAHAYAAN DENGAN OPENGL

Intensitas cahaya yang dipantulkan oleh permukaan suatu benda tergantung dari sifat refleksinya terhadap cahaya yang datang, termasuk warna permukaan dan jenis bahan yang menyusun permukaan tersebut. Sinar yang datang ke suatu permukaan pada umumnya akan dipantulkan secara tersebar. Penyebaran sinar yang dipantulkan dapat dideskripsikan sebagai suatu fungsi dari sudut datang, sudut pantul dan panjang gelombang sinar yang datang tersebut. Pemantulan sinar yang datang pada sebuah elemen kecil pada permukaan merupakan sebuah fungsi seperti di bawah ini :

$$r \ (s, v, n, \lambda) \qquad \text{.....(2.1)}$$

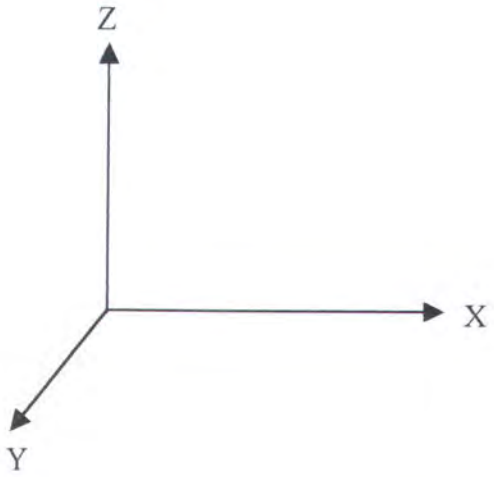
Arti dari masing-masing simbol pada fungsi (2.1) di atas adalah sebagai berikut :

- r adalah intensitas cahaya yang dipantulkan oleh elemen permukaan yang bersangkutan.
- s adalah sudut antara arah datangnya cahaya dengan elemen kecil permukaan.
- v adalah sudut yang menunjukkan arah kamera terhadap elemen kecil permukaan.
- n adalah vektor normal elemen permukaan.
- λ adalah panjang gelombang cahaya yang menyinari elemen permukaan.

Objek–objek yang dibentuk oleh elemen–elemen yang disusun menjadi gambar dua dimensi tersebut nantinya akan dijadikan objek input untuk implementasi dari algoritma *Photometric Stereo* yang dipergunakan di dalam pembuatan perangkat lunak ini. Algoritma Photometric Stereo adalah sebuah algoritma yang memanfaatkan intensitas cahaya yang dipantulkan oleh permukaan suatu objek untuk mendapatkan vektor normal pada elemen-elemen yang menyusun permukaan objek tersebut. Namun sesuai dengan yang telah dibahas pada sub bab 1.4 tentang batasan permasalahan, objek yang dijadikan input pada tugas akhir ini adalah objek citra dua dimensi yang materi pembentuk tiap-tiap titik pada permukaanya sama, sehingga mempunyai sifat yang sama dalam memantulkan cahaya. Sebagai contoh adalah gambar sebuah patung yang terbuat dari perunggu, karena patung tersebut mempunyai warna dan bahan yang sama pada permukaanya, sehingga gambarnya dalam format *gray-scale* hanya dipengaruhi oleh intensitas cahaya yang dipantulkan oleh masing-masing titik pada patung tersebut.

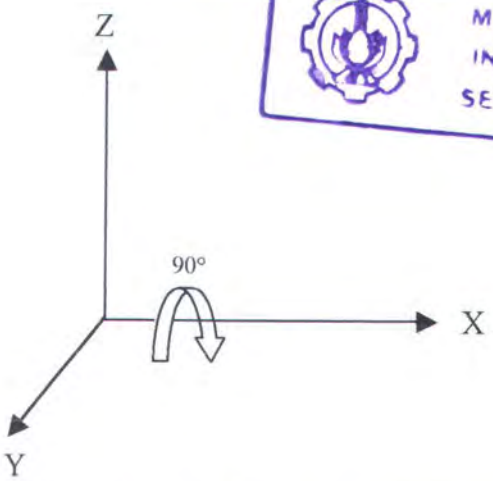
2.1 Vektor pada Ruang Dimensi Tiga

Sumbu–sumbu koordinat Cartesius pada ruang dimensi tiga mempunyai sifat yang sama dengan koordinat Cartesius pada sistem koordinat dua dimensi, yaitu saling tegak lurus antara satu sumbu dengan sumbu yang lainnya. Adapun arah dari sumbu koordinat tersebut adalah sebagai berikut :

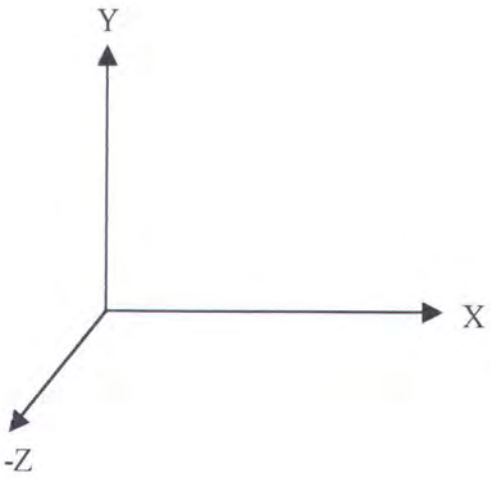


Gambar 2.1. Sumbu-sumbu koordinat pada ruang dimensi tiga.

Akan tetapi untuk mempermudah pembuatan tampilan hasil pengolahan citra masukan, maka dalam pembuatan tugas akhir ini, sumbu-sumbu koordinat Cartesius tersebut di-*rotasi*-kan -90° dengan sumbu perputaran adalah sumbu X. Sehingga arah sumbu koordinat cartesius tersebut menjadi seperti gambar 2.3.



Gambar 2.2. Sumbu-sumbu koordinat pada ruang dimensi tiga dan arah *rotasi*-nya.



Gambar 2.3. Sumbu-sumbu koordinat pada ruang dimensi tiga setelah diputar -90° dengan sumbu putarnya adalah sumbu X.

Jarak antara dua titik pada ruang tiga dimensi seperti halnya pada bidang dua dimensi, dapat dihitung dengan menggunakan teorema *pythagoras*, yaitu sebagai berikut :

$$|P_1P_2| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Pada persamaan (2.2) di atas $|P_1P_2|$ menyatakan jarak dari titik $P_1 = (x_1, y_1, z_1)$ ke titik $P_2 = (x_2, y_2, z_2)$.

2.1.1 Operasi pada Vektor

Misalkan $\boldsymbol{u}$ adalah sebuah vektor dengan komponen (u_1,u_2,u_3), maka panjang vektor tersebut dapat ditulis sebagai :

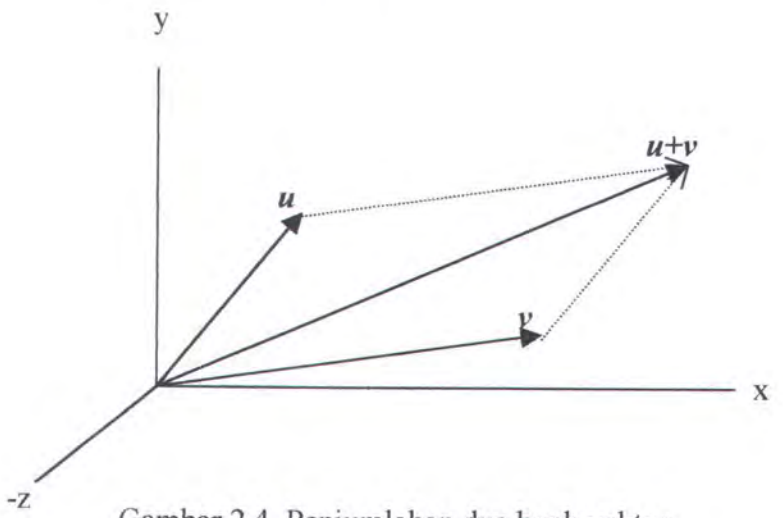
$$|\boldsymbol{u}| = \sqrt{u_1^{\,2} + u_2^{\,2} + u_3^{\,2}} \qquad \text{.....(2.3)}$$

Jika $\boldsymbol{v}$ juga merupakan sebuah vektor lain dengan komponen (v_1,v_2,v_3), maka ada beberapa operasi vektor yang dapat dikenakan pada vektor $\boldsymbol{u}$ dan $\boldsymbol{v}$, yaitu :

- Penjumlahan vektor.

  Hasil penjumlahan dua buah vektor merupakan sebuah vektor yang komponennya adalah hasil penjumlahan komponen-komponen vektor yang saling bersesuaian.

$$(\boldsymbol{u} + \boldsymbol{v}) = (u_1 + v_1, u_2 + v_2, u_3 + v_3) \qquad \text{.....(2.4)}$$



Gambar 2.4. Penjumlahan dua buah vektor.

- Perkalian vektor dengan bilangan skalar.

  Perkalian sebuah vektor $\boldsymbol{u}$ dengan sebuah bilangan skalar c akan menghasilkan sebuah vektor yang masing-masing komponennya merupakan hasil kali komponen vektor $\boldsymbol{u}$ tersebut dengan bilang skalar yang bersangkutan.

  $$c\boldsymbol{u} = (cu_1, cu_2, cu_3) \qquad \text{.....(2.5)}$$

- Perkalian titik dua buah vektor.

  Perkalian titik dua buah vektor dapat dihitung dengan rumus sebagai berikut :

  $$\boldsymbol{u}.\boldsymbol{v} = u_1v_1 + u_2v_2 + u_3v_3 \qquad \text{.....(2.6)}$$

  atau jika yang diketahui adalah panjang masing-masing vektor dan sudut antara kedua vektor yang bersangkutan, rumus perkalian titik vektor tersebut menjadi :

  $$\boldsymbol{u}.\boldsymbol{v} = |\boldsymbol{u}||\boldsymbol{v}| \cos\theta \qquad \text{.....(2.7)}$$

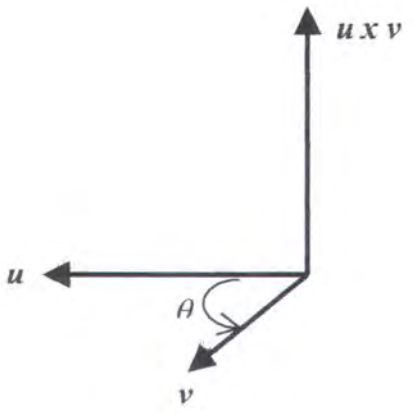
  Dari persamaan (2.7) dapat kita lihat bahwa dua buah vektor saling tegaklurus (ortogonal) jika dan hanya jika $\boldsymbol{u}.\boldsymbol{v} = 0$.

- Perkalian silang dua buah vektor.

  Tidak seperti halnya operasi perkalian titik pada vektor yang menghasilkan bilangan skalar, operasi perkalian silang vektor ini menghasilkan sebuah vektor yang lain. Hasil kali silang $\boldsymbol{u} \times \boldsymbol{v}$ untuk $\boldsymbol{u} = (u_1,u_2,u_3)$ dan $\boldsymbol{v} = (v_1,v_2,v_3)$ didefinisikan sebagai :

  $$\boldsymbol{u} \times \boldsymbol{v} = (u_2 v_3 - u_3 v_2,\ u_3 v_1 - u_1 v_3,\ u_1 v_2 - u_2 v_1) \qquad \text{.....(2.8)}$$

Vektor yang didapatkan dari hasil kali silang tersebut mempunyai arah yang tegak lurus dengan kedua vektor pembentuknya, atau dapat digambarkan sebagai berikut :



Gambar 2.5. Perkalian silang dua buah vektor.

2.2 OpenGL pada Borland Delphi 5.0

OpenGL merupakan sebuah perangkat yang terintegrasi dengan bahasa–bahasa pemrograman, yang berisi fungsi–fungsi dan prosedur–prosedur untuk memanfaatkan perangkat grafis yang ada pada komputer dan sekitar 120 perintah yang digunakan untuk mendefinisikan objek dan operasi–operasi yang diperlukan untuk membentuk aplikasi–aplikasi tiga dimensi secara interaktif. Dengan menggunakan OpenGL, pembuatan sebuah model harus dimulai dari *low level object* yaitu berasal dari sekumpulan fungsi geometri primitif seperti fungsi untuk menggambar titik, garis dan poligon. Pada umumnya urutan langkah dalam menggunakan fungsi OpenGL adalah sebagai berikut :

1. Konstruksi objek dari bentuk geometri yang primitif dengan menggunakan rumus matematika yang sesuai.
2. Atur posisi objek pada ruang tiga dimensi dan pilih sudut pandang yang diinginkan untuk menampilkan objek di layar.
3. Hitung warna dan cahaya untuk objek termasuk *texturing* jika diperlukan.
4. Konversikan deskripsi matematika dari objek dan warna ke titik pada layar.

2.2.1 Sintaks Perintah OpenGL

Perintah-perintah pada OpenGL mempunyai ciri khas tertentu, yaitu selalu menggunakan awalan *gl* dan huruf pertama adalah huruf besar untuk setiap kata yang mengikutinya, misalnya *glClearColor()*. Sedangkan konstanta selalu diawali dengan GL_ dan kata yang mengikutinya semuanya huruf besar, misalnya *GL_COLOR_BUFFER_BIT*. Ada beberapa perintah atau fungsi dalam OpenGL yang mengandung imbuhan angka seperti *glColor3f* yang menandakan jumlah dan tipe argumen. Imbuhan *3f* di sini berarti fungsi tersebut memiliki tiga argumen yang bertipe *float*. Tabel di bawah ini menunjukkan berbagai macam tipe data dari OpenGL yang bersesuaian dengan tipe data pada bahasa C.

Akhiran	Definisi dalam OpenGL
B 8-bit integer	GLbyte
S 16-bit integer	GLshort

i 32-bit integer	GLint, GLsizei
F 32-bit floating-point	GLfloat, GLclampf
D 64-bit floating-point	GLdouble, Lclampd
ub 8-bit unsigned integer	GLubite, Lboolean
us 16-bit unsigned integer	Glshort
ui 32-bit unsigned integer	GLuint, GLenum, Glbitfield

Tabel 2.1. Tipe data pada OpenGl

Contoh berikut ini menunjukkan dua fungsi yang sama, tetapi pada fungsi pertama argumen koordinat *vertex* (piksel) sebagai 32 bit integer sedangkan fungsi yang kedua sebagai floating point.

```
glVertex2i(1,2);
glVertex2f(1.2,3.0);
```

2.2.2 Library dari OpenGL

OpenGL API (Application Programming Interface) dibagi menjadi tiga bagian besar yaitu :

1. AUX library atau Auxiliary (toolkit) terdapat pada *library* glaux.lib. Perintah atau fungsi yang akan digunakan selalu menggunakan awalan *aux*. Auxiliary ini diciptakan sebagai sarana untuk mempelajari dan membuat program OpenGL tanpa harus memperhatikan *environment* (lingkungan) yang sedang dipakai. Fungsi ini membuat, mengatur, memanipulasi *windows* dan masukan dari pemakai. Akan tetapi tidak

semua kode yang akan menyusun aplikasi bisa diandalkan pada fungsi *aux* ini karena fungsi ini hanya berfungsi sebagai saran untuk mempermudah dalam mempelajari OpenGL dengan contoh-contoh objek tertentu.

2. Fungsi asli dari OpenGL sendiri yang selalu dimulai dengan awalan *gl* yang terdapat pada library *Opengl32.dll*. Perintah *drawing* OpenGL sendiri sangat terbatas karena hanya terdiri dari bentuk-bentuk primitif seperti titik, garis dan poligon.
3. Library utilitas yang dimulai dengan awalan *glu* yang terdapat pada library *glu32.dll*. Fungsi ini mengandung fungsi utilitas yang memudahkan pemakai seperti dalam hal pembuatan objek tiga dimensi bola (*gluSphere*), tabung (*gluCylinder*) dan piringan (*gluDisks*).

2.3 Menggambar Objek Geometri

Pada OpenGL ada dua dasar operasi gambar yaitu membersihkan *window* dan menggambar objek geometri termasuk titik, garis dan poligon.

2.3.1 Membersihkan Window

Menggambar pada layar komputer berbeda dengan menggambar pada kertas yang berwarna putih. Pada komputer, memori untuk menampilkan gambar biasanya diisi dengan gambar yang berasal dari perintah gambar yang paling akhir, jadi perlu dibersihkan dengan warna latar belakang sebelum diisi

dengan gambar lagi. Warna latar belakang yang dipilih biasanya tergantung dari aplikasi yang akan dibuat. Untuk membersihkan window dengan warna latar belakang tertentu digunakan dua perintah OpenGL sebagai berikut :

1. *glClearColor (GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha).*

Perintah ini digunakan untuk memilih warna, yang akan digunakan untuk membersihkan latar belakang dalam mode RGBA

2. *glClear(GLbitfield mask).*

Perintah ini digunakan untuk membersihkan buffer yang dispesifikasikan dengan warna yang telah ditentukan.

Contoh berikut menunjukkan perintah untuk membersihkan sebuah buffer dan kemudian mengubah warna latar belakang menjadi hitam. Dalam hal ini yang dibersihkan adalah buffer warna, karena buffer warna merupakan tempat gambar tersebut disimpan.

```
glClearColor(0.0,0.0,0.0,0.0);
glClear(GL_COLOR_BUFER_BIT);
```

2.3.2 Spesifikasi Warna

Pada OpenGL mendeskripsikan objek dan warna adalah proses yang berbeda dan berjalan sendiri-sendiri. Pada umumnya seorang programmer akan mengatur warna (*coloring scheme*) terlebih dahulu baru kemudian menggambar objek. Sebelum warna diubah maka semua objek yang digambar sesudah perintah tersebut akan tetap menggunakan warna terakhir yang terdaftar pada

coloring scheme.

Untuk menentukan warna digunakan perintah *glColor3f()*, jika argumen yang digunakan lebih dari tiga maka argumen keempat adalah alpha yang akan dijelaskan pada bagian sub bab berikutnya yang membahas tentang efek-efek pencahayaan yang dipunyai OpenGL. Contoh berikut menunjukkan urutan langkah dalam proses spesifikasi warna sebelum objek digambar.

```
glColor3f(0.0,0.0,0.1); //setting warna
DrawTheObject(A); // menggambar objek A
```

2.3.3 Menggambar titik, garis dan poligon

Bagian ini menjelaskan segala deskripsi untuk menggambar bentuk geometrik primitif pada OpenGL yang pada dasarnya bermula dari *vertices.* Vertices adalah sejumlah vertek yang mendefinisikan koordinat titik itu sendiri, *endpoints* dari sebuah garis atau sudut-sudut dari suatu poligon.

Titik direpresentasikan dengan floating number yang dinamakan *vertex* dan dihitung dalam sistem koordinat tiga dimensi. Jika pemakai memberikan koordinat dua dimensi maka secara otomatis OpenGL akan memberi angka nol pada koordinat Z. OpenGL selalu bekerja dengan menggunakan koordinat homogen untuk proyeksi bidang tiga dimensi di mana titik direpresentasikan dengan empat *floating numbers* (x,y,z,w) dengan w adalah 1.0. Selama w bukan 0.0 maka koordinat tersebut berhubungan dengan titik-titik Euclidean tiga dimensi (x/w, y/w, z/w). Pada NURBS (Non-Uniform Rational B-Spline) faktor koordinat w sangat diperhitungkan untuk

mempengaruhi bentuk kurva dan permukaan.

Garis pada OpenGL berarti sebuah area yang tertutup oleh segmen garis yang berulang dimana tiap segmen garis dispesifikasikan dengan vertices pada *endpoint*-nya. Hal ini berarti setiap garis terbentuk oleh segmen-segmen garis yang saling berhubungan dan setiap segmen garis didapatkan dengan menghubungkan dua buah vertex.

Sintaks perintah untuk mensepesifikasikan vertices pada OpenGL adalah *glVertex{2 3 4}{s d i f}{v}(TYPEcoords)*. Angka *{2 3 4}* yang mengikuti kata *glVertex* tersebut menunjukkan bahwa OpenGL dapat memiliki dua hingga empat argumen untuk merepresentasikan sebuah koordinat. Contoh berikut menunjukkan cara membuat poligon yaitu harus dimulai dengan *glBegin(mode)* baru dispesifikasikan vertices dan diakhiri dengan *glEnd()*.

```
glBegin(GL_POLYGON);
   glVertex3f(1.0,1.0,1.0);
   glVertex3f(1.0,2.0,1.0);
   glVertex3f(1.0,3.0,1.0);
   glVertex3f(1.0,5.0,1.0);
glEnd();
```

Argumen *mode* pada *glBegin(GLenum mode)* menyatakan bentuk primitif (bentuk sederhana) yang akan dihasilkan dari titik-titik yang didefinisikan. Tabel berikut ini menunjukkan tipe-tipe argumen *mode* yang menggambarkan primitif dari OpenGL.

Mode	Keterangan
GL_POINTS	Titik individual
GL_LINES	Sepasang vertices sebagai sebuah segmen garis
GL_POLYGON	Poligon yang sederhana dari beberapa buah vertices
GL_TRIANGLES	Terdiri dari tiga vertices membentuk segitiga
GL_QUADS	Terdiri dari empat vertices membentuk poligon
GL_LINE_STRIP	Sekelompok segmen garis
GL_LINE_LOOP	Sama dengan atas tapi vertices awal dan akhir berhubungan
GL_TRIANGLE_STRIP	Bidang yang tersusun dari segitiga yang saling berhubungan
GL_TRIANGLE_FAN	Bidang berbentuk kipas tersusun dari segitiga yang saling berhubungan
GL_QUAD_STRIP	Bidang yang tersusun dari segiempat yang saling berhubungan

Tabel 2.2. Mode primitif.

Selain berisi informasi mengenai titik-titik yang dispesifikasikan melalui *glVertex*()*, informasi tambahan mengenai vertex seperti warna, vektor normal, koordinat tekstur dan kombinasi yang lain juga dapat dipanggil antara *glBegin()* dan *glEnd()*.

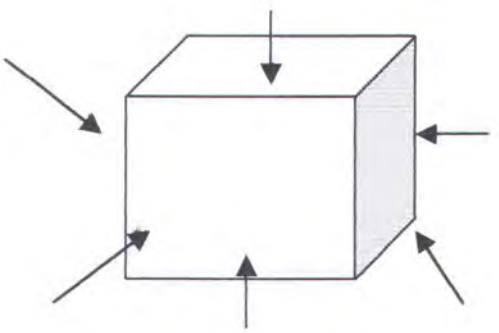
2.4 Pewarnaan dan Pencahayaan

Dengan OpenGL manipulasi pencahayaan terhadap objek dapat dilakukan untuk menghasilkan berbagai macam efek.

2.4.1 Pencahayaan pada OpenGL

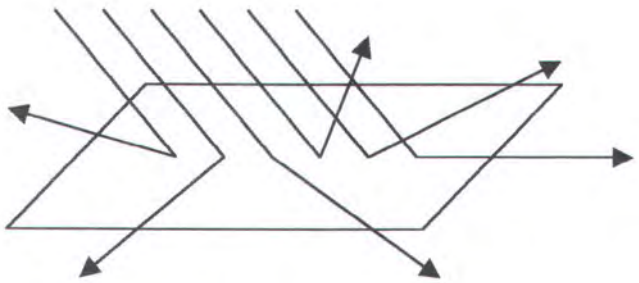
OpenGL memiliki fungsi-fungsi yang mampu mengkondisikan pencahayaan seperti dalam dunia nyata. Ada tiga macam elemen dari cahaya yaitu :

1. Cahaya *Ambient,* cahaya yang berasal dari sembarang arah, mempunyai sumber cahaya tapi seakan-akan sinarnya sudah dipantulkan sehingga arah datang sinar tidak menentu, tanpa memperhatikan sudut pandang ataupun rotasi terhadap objek .



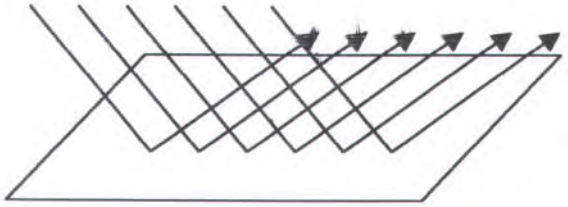
Gambar 2.6. Elemen Ambient.

2. Cahaya *Diffuse*, cahaya yang berasal dari suatu arah tapi dipantulkan oleh objek ke sembarang arah, sehingga objek akan tampak semakin terang jika sinar terletak tegak lurus dengan permukaan objek dibandingkan dengan sinar jika datang dengan membentuk sudut tertentu terhadap permukaan objek .



Gambar 2.7. Elemen Diffuse

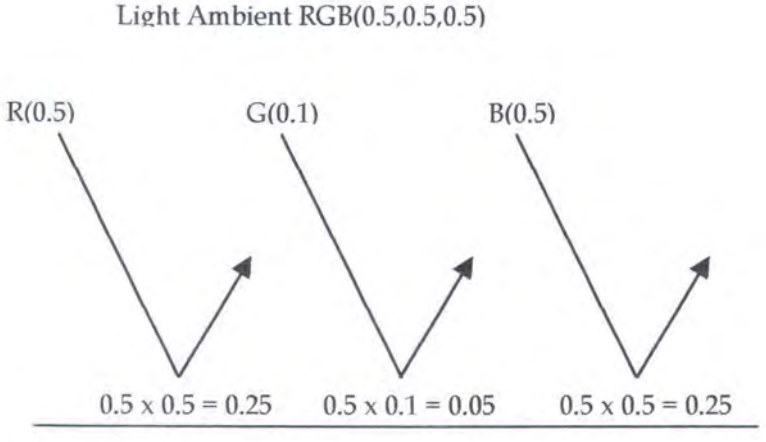
3. Cahaya *Specular*, seperti cahaya *diffuse* tetapi mempunyai arah datang dan dipantulkan dengan tajam ke arah tertentu juga.



Gambar 2.8. Elemen *Specular*

Setiap sumber cahaya pasti terdiri dari tiga elemen di atas dengan intensitas yang berbeda. Pencahayaan baru berperan setengah dalam OpenGL sedangkan setengahnya lagi ditentukan oleh warna materi atau warna dari objek tersebut yang juga mempunyai tiga unsur diatas. Karena warna objek yang ditampilkan ke media display (dalam hal ini monitor) oleh OpenGL tergantung dari warna objek itu sendiri dan juga

warna dari cahaya yang menyinarinya. Sebagai contoh, jika bola berwarna biru diletakkan di ruangan gelap dengan sinar berwarna kuning maka bola juga akan tampak gelap karena semua sinar kuning diserap dan tidak ada sinar biru yang dipantulkan oleh objek. Akan tetapi kebanyakan objek yang disinari oleh sinar putih akan menampakkan warna aslinya. Ketika menggambar objek maka OpenGL menentukan warna yang akan ditampilkan di tiap pixel yang ada di layar. Karena objek mempunyai warna yang dipantulkan dan sumber cahaya mempunyai warnanya sendiri maka tiap vertex dari objek akan memiliki warna RGB berdasarkan hasil perkalian antara efek pencahayaan *ambient, diffuse, specular* dengan unsur *ambient, diffuse, specular* yang dipantulkan oleh objek.



Gambar 2.9. Perkalian antara masing-masing elemen.

2.4.2 Sumber Cahaya

2.4.2 Sumber Cahaya

Sumber cahaya memiliki beberapa properti seperti warna, posisi dan arah. Perintah yang digunakan untuk menspesifikasikan semua properti cahaya adalah *glLight*()*. Sintaks perintah *glLight{i f}{v}(GLenum light, GLenum pname, GLenum TYPEparam)* digunakan untuk menghasilkan cahaya yang jenisnya dispesifikasikan pada parameter *light* yang jumlahnya maksimum delapan *GL_LIGHT0, GL_LIGHT1,..., GL_LIGHT7*. Tabel dibawah ini menunjukkan berbagai macam karakteristik yang didefinisikan oleh *pname.*

Pname	Nilai Default	Definisi
GL_AMBIENT	(0.0, 0.0, 0.0, 0.1)	Intensitas RGBA elemen Ambient
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	Intensitas RGBA elemen Diffuse
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	Intensitas RGBA elemen Specular
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	Posisi cahaya (x,y,z,w)
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	Arah cahaya spotlight(x,y,z)
GL_SPOT_EXPONENT	0.0	Spotlight exponensial
GL_SPOT_CUTOFF	180.0	Spotlight mempunyai sudut cutoff
GL_CONSTANT_ATTENUATION	1.0	Faktor attenuation yang konstan
GL_LINEAR_ATTENUATION	0.0	Faktor attenuation yang linier
GL_QUADRIC_ATTENUATION	0.0	Faktor attenuation yang quadratic

Table 2.3. Karakteristik cahaya.

Proses untuk mengaktifkan setiap sumber cahaya selalu didahului

dengan perintah *glEnable(GL_LIGHTING)*, kemudian dilanjutkan dengan perintah *glEnable(GL_LIGHT0)*, *glEnable(GL_LIGHT1)* dan demikian seterusnya.. Untuk mematikan sumber cahaya tersebut digunakan perintah glDisable*(GL_LIGHTING)*.

Contoh dengan menggunakan bahasa C berikut menunjukkan empat macam karakteristik yang digunakan dalam pencahayaan objek dan cara mengaktifkannya.

```
GLfloat light_ambient  [] = {0.0, 0.0, 0.0, 1.0};
GLfloat light_diffuse  [] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_specular [] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_position [] = {1.0, 1.0, 1.0, 0.0};
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

2.4.3 Warna Cahaya

OpenGL memiliki tiga warna berbeda berkaitan dengan parameter ambient, diffuse dan specular. Parameter *GL_AMBIENT* berhubungan dengan intensitas RGBA cahaya ambient dari sumber cahaya yang akan ditampilkan ke layar. Nilai *default*-nya adalah nol (0.0, 0.0, 0.0, 1.0). Contoh berikut

menunjukkan penggunaan elemen cahaya ambient yang berwarna biru.

```
GLfloat light_ambient [] = {0.0, 0.0, 1.0, 1.0}
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
```

Parameter *GL_DIFFUSE* mungkin satu-satunya elemen cahaya yang secara alami adalah warna dari cahaya itu sendiri (yang paling dominan). Parameter ini mendefinisikan intensitas RGBA cahaya diffuse dari sumber cahaya yang diberikan ke layar. Nilai default-nya adalah cahaya diffuse putih yaitu (1.0, 1.0, 1.0, 1.0) tapi jika sumber cahaya lebih dari satu maka nilai default yang lain adalah (0.0, 0.0, 0.0, 0.0).

Parameter *GL_SPECULAR* mempengaruhi warna sumber cahaya yang menyoroti objek. Jika ingin mendapatkan cahaya yang realistik maka *GL_SPECULAR* harus mempunyai nilai yang sama dengan *GL_DIFFUSE*. Nilai default-nya adalah (1.0, 1.0, 1.0, 1.0).

2.4.4 Posisi dan Arah Cahaya

OpenGL memperlakukan posisi dan arah cahaya seperti memperlakukan posisi objek-objek sederhana lainnya. Dengan kata lain sumber cahaya menjadi subyek daripada transformasi, sama seperti sebuah objek. Secara rinci jika *glLight*()* dipanggil untuk menspesifikasikan posisi dan arah dari sumber cahaya, maka posisi dan arah tadi ditransformasikan oleh matriks *modelview* pada saat ini (yang terakhir dalam stack). Matriks *modelview* adalah matriks

yang menentukan bagaimana posisi pusat sumbu koordinat (berkaitan dengan operasi rotasi dan tansformasi terhadap pusat sumbu-sumbu koordinat) pada saat objek akan digambar. Sedangkan matriks proyeksi (matriks proyeksi adalah matriks yang menentukan proyeksi objek terhadap bidang gambar) tidak berpengaruh terhadap posisi dan arah. Sehingga posisi dan arah sumber cahaya dapat dimanipulasi dengan mengubah stack matriks modelview.

2.4.5 Properti Material

Properti dari material objek juga terdiri dari elemen *ambient, diffuse, specular* dan faktor *shininess*. Sintaks perintah *glMaterial{i f}{v}(GLenum face, GLenum pname, TYPEparam)* digunakan untuk menspesifikasikan properti material apa saja yang disertakan pada perhitungan cahaya. Parameter face berisi *GL_FRONT, GL_BACK* dan *GL_FRONT_AND_BACK* yang menunjukkan bagian (sisi) mana dari objek yang memiliki properti tersebut. Properti material dapat berbeda antara sisi depan dengan sisi belakang. Tabel di bawah menunjukkan berbagai macam properti untuk materi yang didefinisikan oleh argumen *pname* dan nilainya didefinisikan oleh argumen *TYPEparam*.

Pname	Nilai Default	Definisi
GL_AMBIENT	(0.2, 0.2, 0.2, 0.1)	Warna ambient dari material
GL_DIFFUSE	(0.8, 0.8, 0.8, 0.1)	Warna diffuse dari material
GL_AMBIENT_AND_DIFFUSE	-	Warna ambient dan diffuse dari material

GL_SPECULAR	(0.0, 0.0, 0.0, 0.1)	Warna specular dari material
GL_SHININESS	0	Ekponen specular
GL_EMISSION	(0.0, 0.0, 0.0, 0.1)	Warna emissive dari material

Table 2.4.Properti material

Parameter *GL_DIFFUSE* dan *GL_AMBIENT* yang diatur dari *glMaterial*()* mempengaruhi warna dari cahaya diffuse dan cahaya ambient yang dipantulkan oleh objek. Pemantulan cahaya diffuse-lah yang paling banyak berperan bagi mata dalam menerima warna dari objek. Pemantulan ambient mempengaruhi warna keseluruhan dari objek karena pemantulan diffuse hanya berperan jika ada cahaya yang menyinari langsung menuju objek sedangkan pemantulan ambient menyebabkan objek tetap berwarna walau tanpa penyinaran yang langsung menuju objek. Sedangkan pemantulan specular ini akan menghasilkan sorotan. Contoh utama dalam kehidupan sehari-hari adalah bola besi mengkilat yang ditaruh didepan rumah pada siang hari akan tampak menyorot. Walaupun mata sudah melirik ke arah lain tapi sorotan itu masih menyilaukan mata.

BAB III

PENCARIAN VEKTOR NORMAL PADA PERMUKAAN DENGAN ALGORITMA PHOTOMETRIC STEREO

BAB III

PENCARIAN VEKTOR NORMAL PADA PERMUKAAN DENGAN ALGORITMA PHOTOMETRIC STEREO

3.1 Intensitas Cahaya Pada Objek Citra dalam Format Grayscale

Seperti citra pada umumnya, warna tiap-tiap piksel pada objek citra dalam format *gray scale* juga terbentuk dari warna-warna primer yaitu merah, hijau dan biru. Intensitas cahaya pada tiap–tiap piksel pada objek citra dalam format *gray scale* adalah sesuai dengan bobot dan kadar warna merah, hijau dan biru pada tiap–tiap titik yang ditangkap oleh kamera.

Untuk mendapatkan intensitas cahaya tersebut digunakan persamaan sebagai berikut :

```
E[x,y] = 0.3*R[x,y] + 0.6*G[x,y] + 0.1*B[x,y]
```

Adapun maksud dari masing-masing variabel pada persamaan tersebut di atas adalah seperti di bawah ini :

- *E[x,y]* adalah intensitas cahaya pada piksel*(x,y)*
- *R[x,y]* adalah kadar warna merah (*red*) pada piksel*(x,y)*
- *G[x,y]* adalah kadar warna hijau (*green*) pada piksel*(x,y)*
- *B[x,y]* adalah kadar warna biru (*blue*) pada piksel*(x,y)*

- Bobot nilai pengali untuk R, G, B merupakan tingkat terangnya masing–masing warna phospor (merah = 0.3; hijau = 0.6; biru = 0.1) pada mata manusia, yang dalam hal ini dianggap sebagai kameranya.

3.2 Pencarian Normal Vektor

Seperti yang telah dibahas pada bab I yaitu pada bagian batasan permasalahan, untuk implementasi dari algoritma photometric stereo ini ada beberapa hal yang harus diperhatikan untuk citra yang dijadikan masukan. Persamaan yang bisa digunakan untuk mendapatkan intensitas cahaya yang dipantulkan oleh suatu titik pada sebuah objek adalah sebagai berikut :

$$E_n = \frac{E_0}{\pi} \rho_n cos(\theta) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.1)$$

dimana :

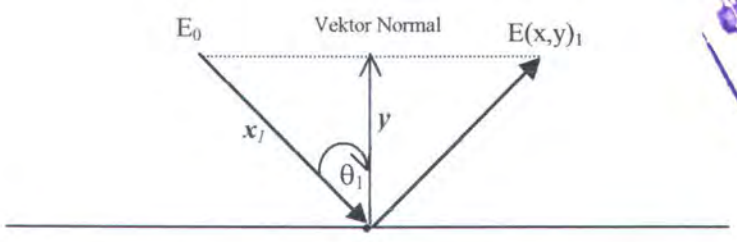
- E_n adalah intensitas cahaya pada titik yang dicari intensitasnya.
- ρ_n adalah sebuah konstanta yang menunjukkan koefisien refleksi pada titik yang bersangkutan, akan tetapi pada pembuatan perangkat lunak ini telah dibatasi bahwa inputnya adalah berupa objek yang setiap titik pada permukaannya mempunyai bahan dan warna yang sama, sehingga nilai koefisien refleksi pada sembarang titik dipermukaan ini bernilai sama..
- E_0 adalah intensitas dari sumber cahaya.
- θ merupakan sudut antara sinar datang dengan normal vektor pada titik yang bersangkutan.

Dari persamaan di atas, dapat diambil beberapa kesimpulan sebagai berikut :

- Untuk persamaan pada bagian $\frac{E_0}{\pi}\rho_n$, karena semua merupakan konstanta, maka hasil perhitungannya juga merupakan sebuah konstanta. Untuk selanjutnya, kita akan menggunakan simbol C untuk menyimpan hasil perhitungan terhadap konstanta-konstanta tersebut.
- Berdasarkan hal tersebut di atas, maka persamaan di atas dapat disederhanakan sebagai berikut :

$$E_n = C\cos(\theta) \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.2)$$

Jika arah sumber cahaya direpresentasikan dengan menggunakan vektor, maka nilai cosinus antara dua buah vektor adalah hasil dari perkalian titik dari kedua vektor tersebut. Jika dimisalkan $E(x,y)_1$ adalah intensitas cahaya pada titik (x,y) dari citra masukan, $E(x,y)_2$ adalah intensitas cahaya pada titik yang sama tetapi objek mendapat sinar dari arah yang berbeda (intensitas sumber cahaya diasumsikan sama yaitu sebesar E_0). Gambar 3.1 menunjukkan arah sinar datang dan sinar pantul pada sebuah titik pada permukaan objek.



Gambar 3.1. Pemantulan cahaya pada sebuah titik.

Pada gambar 3.1, x_1 dan y merupakan dua buah vektor yang masing – masing menunjukkan arah sumber cahaya dan arah vektor normal. Maka nilai Cosinus dari θ_1 dapat dituliskan sebagai berikut :

$$Cos(\theta) = y/x_1 \qquad \ldots\ldots\ldots (3.3)$$

Sehingga persamaan 3.2 akan dapat dituliskan menjadi :

$$E(x,y)_1 = C\,(y/x_1)$$

Dengan demikian :

$$y = E(x,y)_1\,(x_1/C) \qquad \ldots\ldots\ldots\ldots (3.4)$$

Jika objek diberikan cahaya dari arah lain, maka intensitas cahaya pada titik yang sama dapat dihitung sebagai berikut :

$$E(x,y)_2 = C\ \cos(\theta_2)$$

$$= C\,(y\,/\,x_2)$$

Jika nilai y (vektor normal pada titik tersebut) disubstitusi dengan persamaan 3.4, maka persamaan di atas akan menjadi sebagai berikut :

$$E(x,y)_2 = C\,((\,E(x,y)_1\,(x_1/C)\,)\,/\,x_2)$$

$$= (\,E(x,y)_1\,(x_1)\,/\,x_2) \qquad \ldots\ldots\ldots (3.5)$$

Dari persamaan 3.5 di atas kita bisa melihat bahwa nilai C (hasil perhitungan koefisien refleksi permukaan dan intensitas sumber cahaya) tidak berpengaruh dalam perhitungan untuk mendapatkan intensitas cahaya yang dipantulkan oleh sebuah titik pada permukaan, jika objek mendapat sinar dari sumber cahaya yang dari arah yang berbeda hal tersebut dapat kita tuliskan sebagai berikut :

$$E(x,y)_n \approx Cos\,(\theta) \qquad \ldots\ldots\ldots (3.6)$$

Hasil kali titik dua buah vektor dapat direpresentasikan dengan menggunakan perkalian dua buah matriks. Misalkan C adalah hasil kali titik dari vektor $\boldsymbol{a}$ dan $\boldsymbol{b}$, maka persamaan tersebut dapat ditulis sebagai berikut :

$$\boldsymbol{C = a \,.\, b} \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.7)$$

Jika $\boldsymbol{a} = (a_1, a_2, a_3)$ dan $\boldsymbol{b} = (b_1, b_2, b_3)$, maka :

$$C = a_1b_1 + a_2b_2 + a_3b_3 \qquad \ldots\ldots\ldots\ldots\ldots \quad (3.8)$$

Jika masing-masing vektor $\boldsymbol{a}$ dan $\boldsymbol{b}$ dipresentasikan dengan menggunakan sebuah matriks baris yaitu matriks dengan ordo 1 x n ($\boldsymbol{A} = [a_1 \;\; a_2 \;\; a_3]$, $\boldsymbol{B} = [b_1 \;\; b_2 \;\; b_3]$) dan $\mathbf{B}^T$ adalah transpose dari matriks **B,** maka persamaan di atas dapat kita tuliskan sebagai :

$$\mathbf{C} = \mathbf{AB}^T \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.9)$$

$$= \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$= (a_1)(b_1) + (a_2)(b_2) + (a_3)(b_3) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.10)$$

Dari perhitungan di atas dapat kita lihat bahwa persamaan (3.8) mempunyai hasil yang sama dengan persamaan pada (3.10).

Misalkan $\mathbf{a_i} = [a_{i1}, a_{i2}, a_{i3}]$ untuk $i = 1, 2, 3$ adalah merupakan vektor 3D yang menunjukkan arah sumber cahaya yang ke-i, maka tiga buah sumber cahaya dapat ditulis dengan menggunakan sebuah matriks 3x3 sebagai berikut :

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Misalkan juga :

- A^{-1} adalah merupakan invers dari matriks A.
- $X = [\, x_1, x_2, x_3 \,]$ adalah merupakan vektor normal sembarang titik pada sebuah permukaan dan X^T adalah merupakan transpose dari matriks X.
- $Y = [\, y_1, y_2, y_3 \,]$ adalah intensitas cahaya titik tertentu pada permukaan untuk masing-masing sumber cahaya di atas dan Y^T adalah merupakan transpose dari matriks Y.

Maka dapat kita tuliskan hubungan antara antara matriks-matriks tersebut di atas, seperti di bawah ini :

$$Y^T = A\,X^T \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.11)$$

$$X^T = A^{-1}Y^T \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.12)$$

Setelah kita dapat menghitung vektor normal pada sebuah objek citra dengan menggunakan persamaan (3.12), maka selanjutnya kita dapat menentukan intensitas cahaya yang akan dipantulkan oleh tiap-tiap bagian permukaan objek jika mendapat sinar dari sebuah sumber cahaya dengan arah tertentu dengan menggunakan persamaan (3.11). Dengan demikian juga dapat dibuat gambar objek yang akan tertangkap oleh kamera yang berada di depan objek.

BAB IV

PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK

BAB IV

PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK

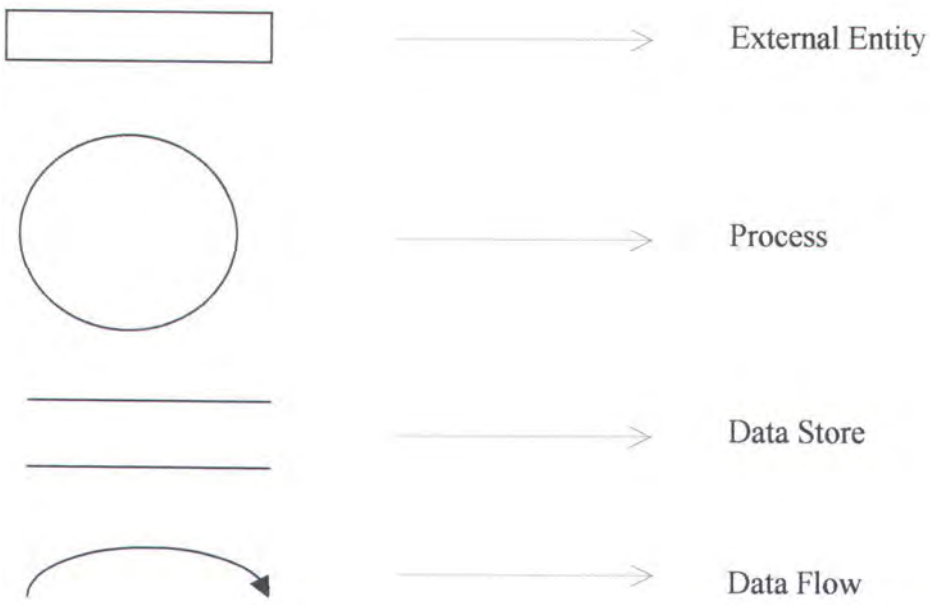
Perangkat lunak yang dikembangkan dalam Tugas Akhir ini dibangun dengan menggunakan bahasa pemrograman **Borland Delphi 5.0** dalam sistem operasi **Windows 98**. Bahasa pemrograman tersebut dipilih karena *Borland Delphi* mempunyai pola yang terstruktur dan mendukung pemrograman berorientasi pada obyek.

4.1 Perancangan Perangkat Lunak

Perancangan perangkat lunak ini meliputi beberapa tahap, langkah pertama adalah membuat analisa kebutuhan dengan menggunakan diagram aliran data (*data flow diagram*). Langkah selanjutnya adalah merancang struktur data dan proses. Perangkat lunak yang dibuat didesain dengan menggunakan *Object-Oriented*, tetapi tidak sepenuhnya, sehingga digunakan dua metode yaitu *data-oriented design* menggunakan *process structure design* dengan *Jackson Diagram* dan *object-oriented design*.

4.1.1 Data Flow Diagram

DFD (*Data Flow Diagram*) merupakan teknis grafis yang menggambarkan aliran data dan transformasi dari data input sampai pada outputnya. Komponen-komponen dari DFD adalah :



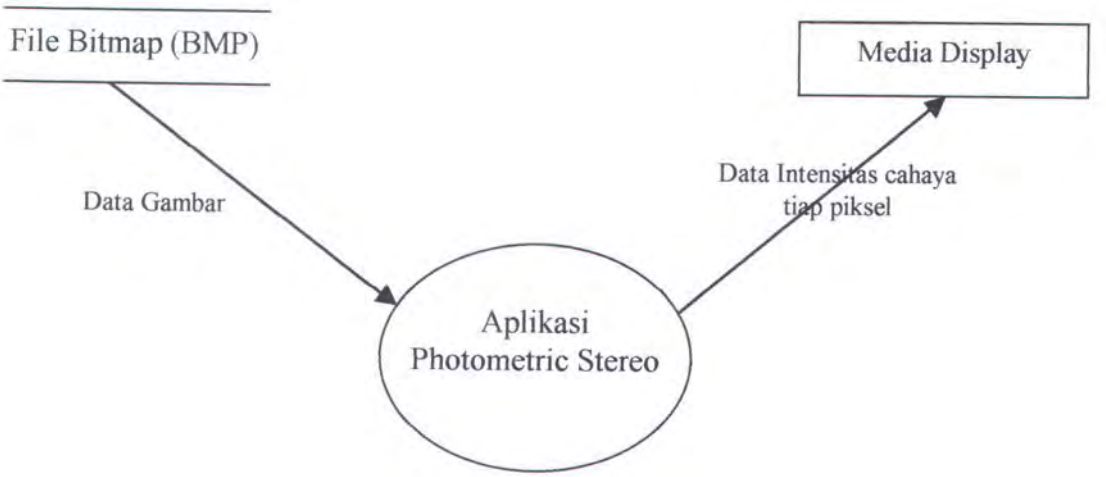
Gambar 4.1. Komponen DFD

Adapun arti masing-masing komponen tersebut adalah sebagai berikut :

- *External Entity*, yaitu entitas yang berperan sebagai penghasil atau pemakai informasi yang berada di luar sistem yang dimodelkan.

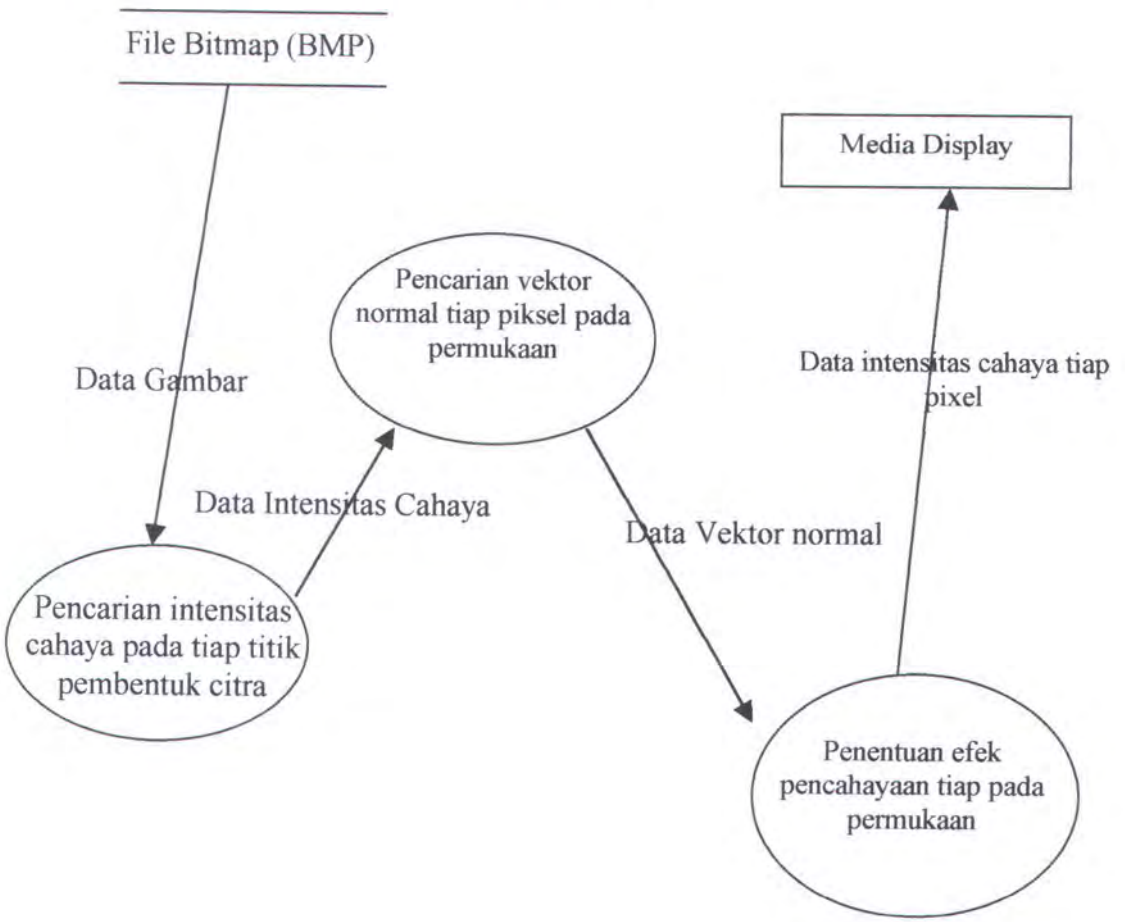
- *Process*, yaitu proses transformasi informasi yang berada di dalam sistem.
- *Data Store*, adalah perangkat penyimpanan data dalam sistem yang dapat diakses oleh satu atau beberapa proses. Biasanya *data store* ini berupa file dalam media penyimpanan.
- *Data flow*, yaitu arah satu atau sekumpulan data yang mengalir dari satu proses ke proses lainnya. Tanda panah menunjukkan arah aliran data.

Susunan DFD perangkat lunak untuk rekonstruksi arah vektor normal permukaan suatu objek dengan menggunakan data yang berasal dari input berupa citra dua dimensi dalam format gray level yang dibuat adalah seperti gambar 4.2 di bawah ini.



Gambar 4.2. Context Level dari DFD Pembentukan Permukaan

DFD pada gambar 4.2 di atas adalah *Context Level DFD*, yaitu diagram aliran data yang merupakan skema dasar untuk perangkat lunak yang akan dikembangkan. Dari diagram alir di atas dapat kita lihat bahwa aplikasi mendapat masukan data gambar yang berasal dari luar aplikasi tersebut, yang berupa file bitmap (BMP), dan file-file tersebut berisi data gambar dua dimensi dalam format *gray level*. Keluaran yang dihasilkan oleh aplikasi ini berupa data yang berisi informasi vektor normal tiap-tiap titik yang menyusun permukaan yang telah direkonstruksi dengan menggunakan algoritma Photometric Stereo, yang nantinya akan ditampilkan ke media display.



Gambar 4.3. DFD Level 2, Detail Aplikasi Pembentukan Permukaan

Gambar 4.3 merupakan detail DFD dari aplikasi untuk menentukan efek pencahayaan pada permukaan suatu objek. Dalam aplikasi ini, posisi sumber cahaya sudah ditentukan. Aplikasi ini terdiri dari tiga buah proses utama, yaitu :

- Pencarian Intensitas cahaya pada tiap titik pembentuk citra.

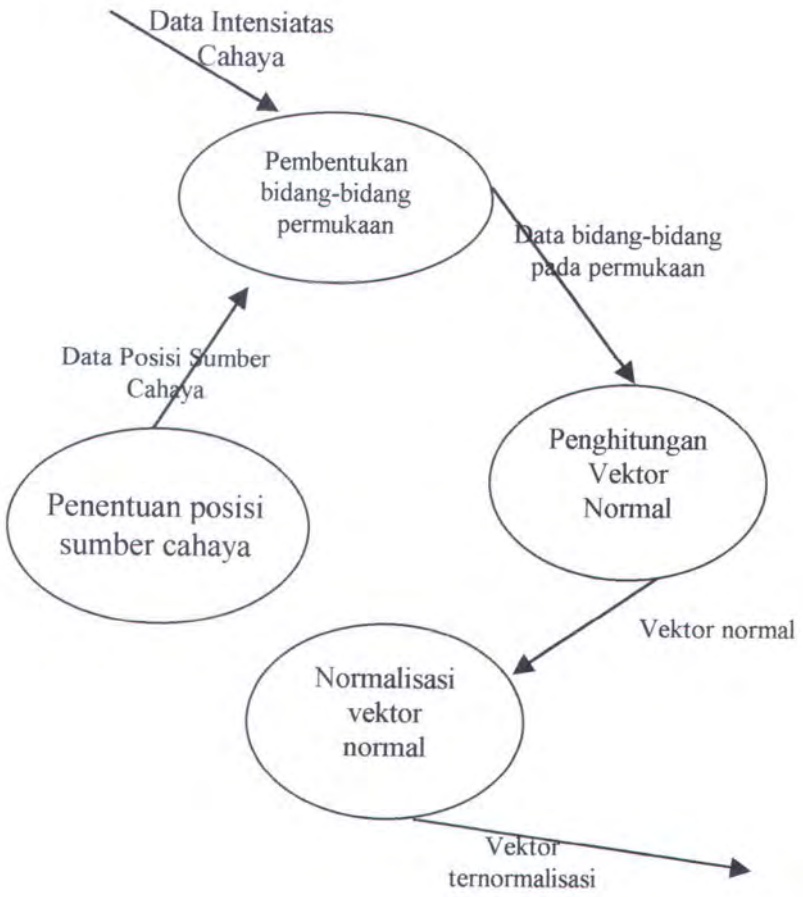
  Proses ini mengambil masukan langsung dari file bitmap yang diberikan. Dari informasi level abu-abu yang diberikan oleh citra masukan tersebut, proses ini akan menghitung intensitas cahaya (proses perhitungannya dibahas pada bagian berikutnya) yang dipantulkan oleh tiap-tiap titik yang membentuk citra tersebut. Proses ini akan menghasilkan informasi intensitas cahaya setiap titik yang membentuk citra masukan.

- Pencarian vektor normal tiap titik pada permukaan.

  Dari informasi intensitas cahaya ketiga buah citra masukan dan informasi arah sumber cahaya yang menyinari objek pada masing-masing gambar yang diberikan oleh proses sebelumnya, pada proses yang kedua ini akan dihitung vektor normal masing-masing titik pada permukaan objek.

- Penentuan efek pencahayaan tiap piksel pada permukaan.

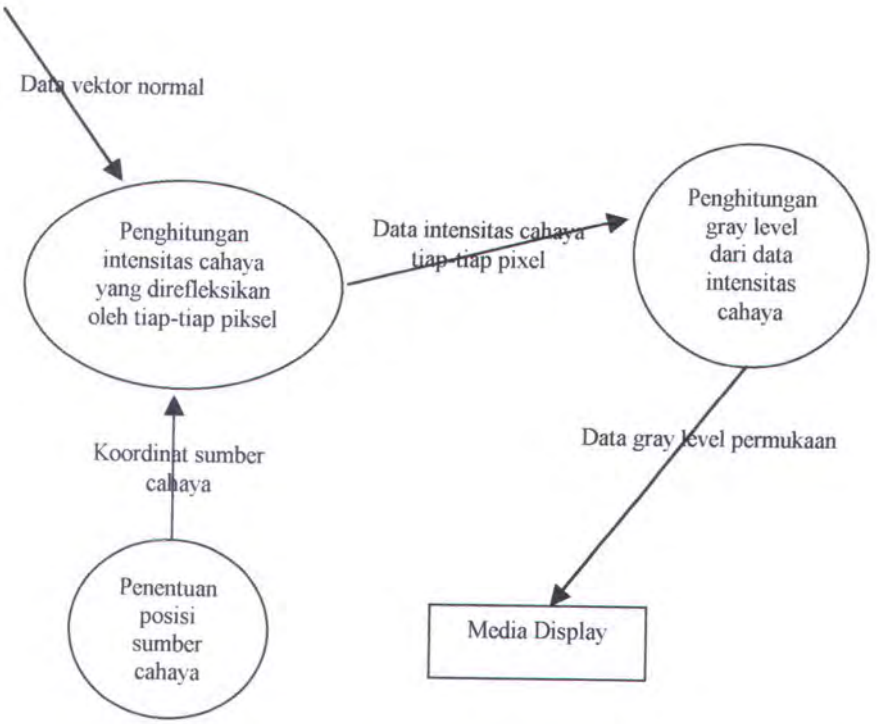
  Dengan data vektor normal tiap-tiap titik pada permukaan, proses ini bertujuan untuk menentukan nilai intensitas cahaya yang akan dipantulkan oleh masing-masing titik pada permukaan tersebut jika mendapat sinar dari sumber cahaya dari arah tertentu.



Gambar 4.4. DFD Level 3, Detail proses Pencarian Normal.

DFD level 3 yang ditunjukkan pada gambar 4.4 di atas merupakan rincian proses pencarian vektor normal permukaan dan kemudian dinormalisasi untuk dicari vektor ternormalisirnya. Vektor ternormalisir adalah vektor yang masing-masing anggotanya dibagi dengan panjang vektor itu sendiri, sehingga panjang dari vektor ini selalu = 1. Proses ini merupakan proses utama dari

keseluruhan proses yang ada pada program yang dibuat, karena proses ini merupakan proses untuk mengimplementasikan algoritma *photometric stereo*, yang digunakan untuk menghitung vektor normal masing-masing titik pada permukaan.



Gambar 4.5 Level 3, Penghitungan efek cahaya pada tiap-tiap pixel

DFD gambar 4.5 menunjukkan proses dari pembentukan permukaan dengan efek pencahayaannya pada masing-masing piksel yang menyusun permukaan tersebut. Dari data vektor normal permukaan yang dikirimkan oleh proses sebelumnya ke proses ini, maka dilakukan penghitungan terhadap intensitas cahaya yang dipantulkan oleh tiap-tiap bagian dari permukaan objek

yang bersangkutan. Intensitas cahaya yang dipantulkan oleh suatu titik pada permukaan merupakan hasil kali titik (*vector dot product*) antara vektor yang menunjukkan posisi sumber cahaya dengan vektor normal pada titik tersebut. Setelah mendapatkan nilai intenitas cahaya yang dipantulkan oleh masing-masing titik pada permukaan tersebut, maka untuk menampilkan hasil perhitungan ke media display, data intensitas cahaya tersebut terlebih dahulu digunakan untuk menghitung nilai dari level warna abu-abu (*gray level*) yang akan ditangkap oleh kamera jika objek tersebut berada di depan kamera dan mendapatkan sinar dari sumber cahaya yang posisinya telah ditentukan dan merupakan masukan dalam proses ini. Selanjutnya data gray level tersebut akan dikirimkan ke media display sebagai keluaran dari aplikasi ini.

4.1.2 Perancangan Data

Dalam pembuatan perangkat lunak untuk tugas akhir ini, selain tipe-tipe data standar yaitu tipe data yang merupakan bawaan dari Borland Delphi 5.0, penulis juga menggunakan tipe-tipe data yang dibuat sendiri (*user defined data type*) untuk memudahkan mendeklarasikan variabel-variabel yang nantinya akan digunakan dalam pemrograman. Tipe-tipe data yang didefinisikan tersebut berbentuk *Class, record, array* dan *dynamic-array.*

Tipe data yang berbentuk Class dideklarasikan untuk variabel-variabel yang selain berisi data juga mempunyai operasi-operasi khusus terhadap variabel-variabel tersebut. Operasi-operasi tersebut berupa *function-function* dan

procedure-procedure juga menjadi anggota (*field*) dari class tersebut. Adapun tipe data yang berbentuk Class ini adalah *TPhotoStereo*, *TIOClass*, *Vec3D*. Penjelasan secara terperinci untuk masing-masing tipe data tersebut adalah sebagai berikut :

- *TPhotoStereo* adalah Class yang digunakan untuk menyimpan data-data yang diperlukan untuk rekonstruksi bidang-bidang pembentuk permukaan, diantaranya yaitu ukuran citra (panjang dan lebar citra masukan), posisi sumber cahaya yang menyinari obyek, intensitas cahaya citra pada tiap-tiap titik pada citra, maupun vektor normal pada tiap-tiap titik pada citra. Sedangkan procedure dan function yang ada pada Class ini digunakan untuk mendapatkan titik-titik kontrol permukaan dan vektor normalnya.
- *TIOClass* merupakan *Object* yang didefinisikan untuk menangani semua proses yang berhubungan dengan penulisan dan pembacaan *project* rekonstruksi permukaan yang ada pada media penyimpan (*Hardisk*).
- *Vec3D* adalah Object yang digunakan untuk menyimpan data vektor (untuk vektor pada ruang dimensi tiga), dan juga procedure-procedure serta function yang merupakan operasi-operasi terhadap vektor 3D, di antaranya adalah perkalian dan penjumlahan vektor.

Tipe data yang berbentuk record digunakan untuk mendeklarasikan variabel-variabel yang hanya menyimpan data-data yang diperlukan tanpa ada procedure-procedure ataupun function-function khusus yang menjadi anggota

dari tipe data ini, yang menjadi satu kesatuan dalam satu record. Sebagai contoh yaitu koordinat (bukan vektor) dalam ruang tiga dimensi, di mana posisi sebuah titik dalam ruang dimensi tiga, ditunjukkan oleh sebuah sistem koordinat yang terdiri tiga buah variabel yang menjadi satu kesatuan, yaitu posisi pada sumbu X, Y dan Z. Tipe data yang berbentuk record yang dideklarasikan dalam pembuatan program untuk tugas akhir ini antara lain :

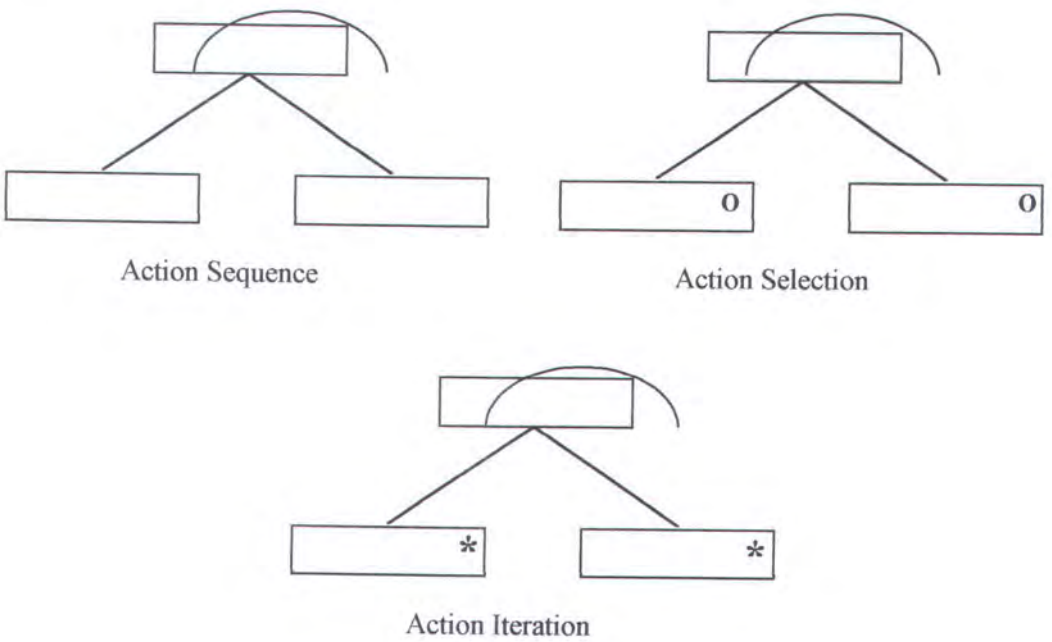
- *TCoord3D*, tipe data ini merupakan record yang berisi data-data koordinat dalam ruang tiga dimensi.
- *TProjectRecord* menyimpan informasi tentang *project* rekonstruksi permukaan yang merupakan hasil pembacaan dari media penyimpan maupun *project* yang nantinya akan disimpan ke media penyimpan. Record ini adalah merupakan salah satu anggota dari object *TIOClass*.

Tipe data yang berbentuk *array* digunakan untuk menyimpan data yang berupa tabel, yang hanya terdiri dari satu tipe data. Tipe data yang merupakan *user defined data tipe* yang berbentuk *array* yang dideklarasikan dalam pembuatan perangkat lunak ini antara lain :

- *ArrInt* adalah *array* yang masing-masing elemennya mempunyai tipe data *Integer* yaitu tipe data yang merupakan tipe data standar pada Borland Delphi , dan menyimpan data berupa bilangan bulat.

4.1.3 Perancangan Proses

Untuk perancangan proses ini digunakan metode yang diperkenalkan oleh Jackson, yang kemudian dikenal sebagai Jackson System Development (JSD) atau Jackson Diagram. Beberapa variasi dalam JSD, seperti yang ditunjukkan dalam gambar 4.4, yaitu proses berurutan, pilihan dan perulangan.



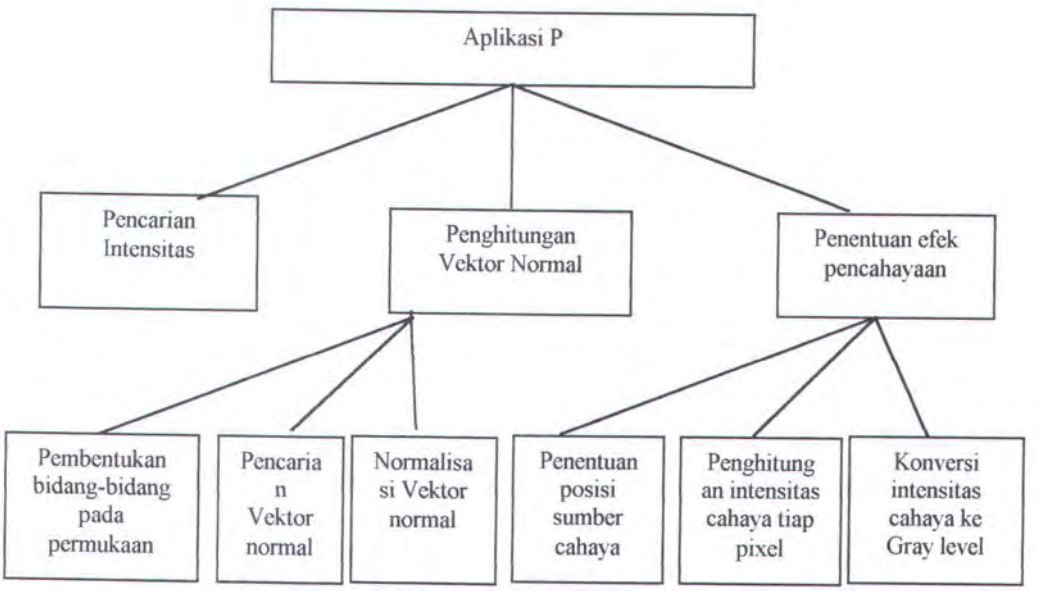
Gambar 4.6 . Jackson Diagram

Maksud dari masing-masing gambar tersebut di atas dapat dijelaskan sebagai berikut :

- *Action Sequence,* yaitu suatu proses yang dibentuk secara sequential oleh beberapa proses sebelumnya.
- *Action Selection,* yaitu suatu pilihan optional dari beberapa proses.

- *Action Iteration*, yaitu suatu proses perulangan yang jumlah perulangannya tidak selalu sama setiap kali proses tersebut dipanggil.

Diagram desain procedure-procedure dari program yang akan dibuat dalam tugas akhir ini dapat dilihat pada gambar 4.5. Program yang dibuat untuk aplikasi ini terdiri dari tiga buah proses utama. Masing-masing proses itu adalah digunakan untuk mencari intensitas cahaya pada citra masukan, menghitung vektor normal permukaan dan pembentukan citra permukaan yang telah ditambahkan dengan efek pencahayaan sesuai dengan posisi sumber cahaya yang menyinari objek tersebut, sehingga citra yang terbentuk mempunyai kesan citra tiga dimensi.



Gambar 4.6. Diagram Struktur Proses.

Pada proses pencarian intensitas, dilakukan pembacaan per-piksel pada citra masukan dan dengan melihat nilai level keabu-abuan pada titik tersebut,

yang selanjutnya dipecah menjadi warna-warna primer yaitu merah, hijau dan biru. Dengan mengetahui kadar masing-masing warna primer pada titik-titik tersebut maka dapat dihitung intensitas cahaya yang dipantulkan oleh titik-titik yang terdapat pada permukaan obyek.

Setelah nilai intensitas cahaya tiap-tiap titik pada permukaan objek selesai dihitung, maka hasil dari perhitungan ini akan dikirim ke proses selanjutnya, yaitu proses penghitungan vektor normal. Proses penghitungan vektor normal ini meliputi tiga proses di dalamnya yang dikerjakan secara berurutan pada tahap ini. Proses tersebut yaitu pencarian bidang-bidang kecil penyusun permukaan, penghitungan vektor normal dari tiap-tiap bidang dan normalisasi dari vektor normal tersebut.

Proses yang terakhir adalah proses untuk menentukan efek pencahayaan tiap-tiap titik pada permukaan objek jika permukaan objek yang yang menghadap ke arah kamera mendapatkan sinar dari sebuah sumber cahaya yang posisinya akan ditentukan dalam proses ini. Setelah menentukan posisi sumber cahaya yang akan digunakan untuk memberikan cahaya kepada permukaan objek masukan, maka selanjutnya akan dilakukan proses penghitungan intensitas cahaya yang akan dipantulkan oleh masing-masing titik pada permukaan objek tersebut. Kemudian dari data intensitas cahaya tersebut, akan dapat dihitung nilai dari gray level yang akan ditangkap oleh kamera, jika objek tersebut diletakkan di depan sebuah kamera. Data gray level permukaan yang dihasilkan pada proses terakhir ini akan ditampilkan langsung ke media display.

4.2 Implementasi Perangkat Lunak

Perancangan data dan proses perangkat lunak diimplementasikan dengan metode OOP (*Object Oriented Programming*). Metode ini digunakan karena model data OOP bekerja secara dinamis di mana alokasi memori yang dipergunakan sesuai dengan kebutuhan. Selain itu metode OOP lebih efisien karena suatu Object merupakan gabungan antara data variabel dan procedure ataupun function.

4.2.1 Implementasi Struktur Data

Struktur data yang dirancang pada perangkat lunak yang dikembangkan pada tugas akhir ini disusun dengan sistem Pemrograman Berorientasi pada Objek atau yang sering disebut dengan OOP. Tipe data Object yang digunakan pada pembuatan perangkat lunak ini sebagian merupakan bawaan dari Borland Delphi 5.0 (*built in*) dan sebagian lagi merupakan buatan penulis sendiri (*user defined data type*). Selain struktur Object, struktur lain yang dipakai dalam perancangan perangkat lunak ini adalah struktur array dan record. Implementasi struktur array yang dipergunakan adalah sebagai berikut :

```
ArrInt = Array of Array of Integer;
```

User defined data type di atas, digunakan sebagai dasar dari variabel-variabel yang bertipe array. Array dari *ArrInt* dipergunakan untuk menyimpan

data array yang bertipe integer (bilangan bulat), digunakan untuk menampung intensitas cahaya citra-citra yang akan direkonstruksi permukaannya.

Sedangkan untuk struktur data yang menggunakan struktur record adalah sebagai berikut :

```
TCoord3D = Record
     X      :     Real;
     Y      :     Real;
     Z      :     Real;
End;
```

Tipe data *TCoord3D* didefinisikan untuk mencatat koordinat-koordinat pada ruang dimensi tiga yang terdiri dari tiga sumbu yaitu X, Y dan Z. Karena di dalam program yang akan dibuat dipergunakan perintah-perintah OpenGL, di mana koordinat-koordinat di dalam ruang dimensi tiga tidak harus berupa bilangan bulat, maka untuk sumbu X, Y maupun Z digunakan tipe data *Real.*

```
TProjectRecord = Record
    Image1,Image2,
    Image3          :   String[255];
    X1,X2,X3        :   Integer;
    Y1,Y2,Y3        :   Integer;
    Z1,Z2,Z3        :   Integer;
End;
```

TProjectRecord digunakan sebagai tipe data dasar untuk variabel yang menyimpan data project rekonstruksi. Untuk rekonstruksi permukaan suatu citra

dalam format *gray level*, diperlukan tiga buah citra yang berasal dari Object yang sama akan tetapi mendapat cahaya dari tiga arah yang berbeda. Adapun kegunaan masing-masing anggota record dari TProjectRecord adalah sebagai berikut :

- *Image1, Image2, Image3* yang masing-masing bertipe *string*, dipergunakan untuk menyimpan nama dan lokasi tiga buah citra masukan yang disimpan di media pemyimpan.
- $X_{1..3}$, $Y_{1..3}$ dan $Z_{1..3}$ digunakan untuk menyimpan koordinat yang merupakan posisi sumber cahaya yang menyinari masing-masing citra masukan dalam format ruang tiga dimensi.

Dan yang terakhir adalah tipe-tipe data yang berbentuk Object, tipe data tersebut adalah sebagai berikut :

```
TPhotoStereo = Class
private
  W,H                          : Integer;
  GridSize,VecLength           : Integer;
  S1,S2,S3                     : Vec3d;
  E1,E2,E3                     : ArrInt;
  Grad                         : Array Of Array Of Vec3d;
public
  Function GetIntensity(Image : TImage) : ArrInt;
  Function GetRed  (Pix : Integer) : Integer;
  Function GetGreen(Pix : Integer) : Integer;
  Function GetBlue (Pix : Integer) : Integer;
```


```
    Procedure Init;
    Procedure PSM;
    Procedure Run;
    Procedure NeedleMap;
    Procedure SetLightSource;
    Procedure GetImageIntensity;
    Procedure LambertShade(Ls : Vec3d);
    Procedure ClearCanvas(Image : TImage);
    Procedure MakeList(V1,V2,V3,V4,Grad : Vec3D);
End;
```

Class *TPhotoStereo* yang didefinisikan di atas merupakan class utama dalam aplikasi rekonstruksi permukaan ini dengan efek pencahayaan ini, karena pada class ini terdapat penerapan dari algoritman *Photometric Stereo.* Berikut akan dijelaskan kegunaan dari masing-masing elemen dari class ini.

- Variabel W dan *H* yang bertipe integer masing-masing menyimpan data berupa lebar dan tinggi dari citra masukan yang akan direkonstruksi.
- Variabel *s1*, *s2* dan *s3* menyimpan koordinat-koordinat sumber cahaya yang menyinari obyek dari tiga arah yang berbeda sehingga untuk sebuah project rekonstruksi, didapatkan tiga buah citra masukan yang berasal dari gambar satu buah obyek yang mendapatkan sinar dari tiga arah yang berbeda.
- *E1*, *E2* dan *E3* yang bertipe *Array of Integer* adalah variabel yang menyimpan data-data intensitas cahaya pada tiap-tiap titik dari ketiga citra masukan.
- Variabel *Grad* ini menampung data vektor normal pada masing-masing titik pada permukaan objek yang dijadikan masukan pada aplikasi ini.

- Pemanggilan *Function GetIntensity* akan menghasilkan nilai intensitas cahaya pada citra yang dijadikan parameter pemanggilan fungsi tersebut.
- Fungsi-fungsi berikut yaitu *GetRed, GetGreen* dan *GetBlue* dipergunakan untuk memisahkan warna-warna primer yang menyusun warna tertentu, di mana pemanggilan fungsi-fungsi tersebut masing-masing akan menghasilkan kadar dari warna merah, hijau dan biru yang menyusun suatu warna sekunder.
- Sesuai dengan namanya, *Procedure Init* dipergunakan untuk memberi nilai awal terhadap variabel-variabel tertentu yang memerlukan inisialisasi nilai.
- *Procedure PSM* adalah merupakan proses yang paling penting dari rekonstruksi permukaan tiga dimensi ini, karena pada procedure ini terdapat proses yang merupakan implementasi dari algoritma dari *Photometric Stereo.*
- *Procedure Run* ini berguna untuk memanggil procedure lainnya, yaitu *Procedure Init, Procedure NeedleMap* dan *Procedure LambertShade* untuk menampilkan hasil perhitungan ke media display.
- *Procedure NeedleMap* ini adalah procedure yang akan menampilkan peta dari vektor normal pada tiap-tiap titik dari permukaan yang telah direkonstruksi.
- *Procedure SetLightSource* ini dipergunakan untuk mengambil data posisi dari sumber cahaya yang menyinari obyek dari *PropertiesForm.*

- *Procedure LambertShade* berisi proses-proses untuk menampilkan hasil rekonstruksi permukaan dari arah depan dalam format *gray level.*
- *Procedure ClearCanvas* berguna untuk membersihkan image dari gambar yang ditampilkan sebelumnya.
- Pemanggilan *Procedure MakeList* akan menambahkan vektor normal pada suatu titik yang baru dihitung dimasukkan ke dalam variabel *Linked List* yang ada pada unit *InitGlComponent.*

Tipe data berbentuk Object selanjutnya adalah tipe data *Vec3D.* Deklarasi tipe data tersebut adalah sebagai berikut :

```
Vec3d = Class (TObject)
Private
   X,Y,Z : Real;

Public
   Function GetX : Real;
   Function GetY : Real;
   Function GetZ : Real;
   Function Times(C : Real; Vec : Vec3d) : Vec3d;
   Function Minus(a,b : Vec3d) : Vec3d;
   Function Norm : Real;
   Function Dot(a,b : Vec3d) : Real;
   Function Cross(a,b : Vec3d) : Vec3d;

   Procedure SetValue(XVal,YVal,ZVal : Real);
   Procedure Normalise;
End;
```

Tipe data ini dideklarasikan untuk mewakili variabel-variabel yang bisa menampung data berupa vektor pada ruang dimensi tiga dengan semua operasi-operasi yang bisa dikenakan terhadap vektor tersebut. Adapun kegunaan dari masing-masing anggota dari tipe data tersebut adalah sebagai berikut :

- *X, Y,* dan *Z.* Ketiga variable yang bertipe *real* ini merupakan variabel yang berfungsi untuk menampung informasi tentang arah vektor pada ruang dimensi tiga.
- *Function GetX : Real;*

  Function ini dipergunakan untuk mendapatkan arah vektor terhadap sumbu X (nilai dari variabel X).
- *Function GetY : Real;*

  Function ini dipergunakan untuk mendapatkan arah vektor terhadap sumbu Y (nilai dari variabel Y).
- *Function GetZ : Real;*

  Function ini dipergunakan untuk mendapatkan arah vektor terhadap sumbu Z (nilai dari variabel Z).
- *Function Times(C : Real; Vec : Vec3d) : Vec3d;*

  Function ini menghasilkan perkalian vektor *Vec* dengan bilangan skalar *C* yang bertipe real.
- *Function Minus(a,b : Vec3d) : Vec3d;*

  Function ini dipergunakan untuk mencari selisih antara dua vektor yang menjadi parameter dari fungsi ini, yaitu *a* dan *b* yang juga menghasilkan sebuah vektor baru.

- *Function Norm : Real;*

  Function *Norm* ini dipergunakan untuk mencari panjang vektor yang bersangkutan.

- *Function Dot(a,b : Vec3d) : Real;*

  Function ini dipergunakan untuk mendapatkan hasil perkalian titik (*dot product*) dari dua buah vektor yang akan menghasilkan bilangan skalar yang bertipe *real.*

- *Function Cross(a,b : Vec3d) : Vec3d;*

  Function ini dipergunakan untuk mendapatkan hasil perkalian silang (*cross product*) dari dua buah vektor yang akan menghasilkan sebuah vektor yang lain.

- *Procedure SetValue(XVal,YVal,ZVal : Real);*

  Procedure ini dipergunakan untuk memberi nilai kepada variable-variabel X, Y, dan Z dari vektor yang bersangkutan.

- *Procedure Negate;*

  Procedure ini dipergunakan untuk mendapatkan *negasi* (lawan) dari vektor itu sendiri.

- *Procedure Normalise;*

  Procedure ini dipergunakan untuk mendapatkan normalisasi dari sebuah vektor.

Tipe data selanjutnya yang berbentuk object dan merupakan *user defined data type* adalah *TIOClass*. Tipe data ini tidak ada hubungannya secara langsung dengan proses pembentukan permukaan dari citra dalam format *gray scale*. Tipe data ini dipergunakan untuk menyimpan data project rekonstruksi, yaitu data tentang nama dan lokasi citra masukan di media penyimpan serta data tentang posisi sumber cahaya yang menyinari masing-masing citra. Deklarasi dari tipe data resebut adalah sebagai berikut :

```
TIOClass        = Class (TObject)
Private
    FileProject : File Of TProjectRecord;
Public
    TheProject : TProjectRecord;
    Procedure OpenProject(PName : String);
    Procedure SaveProject(PName : String);
End;
```

Penjelasan masing-masing elemen untuk tipe data di atas adalah sebagai berikut :

- *FileProject*. Variable ini adalah variabel yang bertipe berkas (file). Variabel ini nantinya akan menampung data berurutan yang masing-masing terdiri dari data berupa record yang bertipe TProjectRecord.
- *TheProject* adalah variabel yang digunakan untuk menyimpan satu record yang dibaca dari *FileProject*.
- *Procedure OpenProject(PName : String);*

Procedure ini dipergunakan untuk membaca file project rekonstruksi yang ada di media penyimpan di mana data hasil pembacaan project-nya di media penyimpan dimasukkan ke variabel *TheProject*.

- *Procedure SaveProject(PName : String);*

  Procedure ini dipergunakan untuk menyimpan project rekonstruksi yang data-data project-nya tersimpan di variabel *TheProject*, ke media penyimpan.

4.2.2 Implementasi Proses

Proses-proses yang akan dijelaskan pada bagian ini merupakan implementasi procedure dan function dari class-class yang telah dibahas di atas. Yang pertama akan dibahas di sini adalah implementasi dari class *TPhotoStereo*.

Function GetIntensity

```
Function TPhotoStereo.GetIntensity(Image : TImage) : ArrInt;
Var Intens : ArrInt;
    A,B    : Integer;
Begin
    W := Image.Width;
    H := Image.Height;
    SetLength(Intens,W,H);
    For A:=0 To (W-1) Do
    Begin
        Application.ProcessMessages;
        For B:=0 To (H-1) Do
        Begin
           Intens[A,B]:=
```

```
                    Trunc(0.3*GetRed(Image.Canvas.Pixels[A,B])  +
                          0.6*GetGreen(Image.Canvas.Pixels[A,B])+
                          0.1*GetBlue(Image.Canvas.Pixels[A,B]));
        End;
    End;
    GetIntensity := Intens;
    Finalize(Intens);
End;
```

Function GetIntensity di atas dipergunakan untuk menghitung intensitas cahaya tiap-tiap titik pada citra yang dijadikan parameter pada waktu pemanggilan function ini. Hasil perhitungan tersebut untuk sementara disimpan pada variabel *Intens* yang bertipe *Dynamic Array* yaitu *ArrInt.* Setelah perhitungan ini selesai maka nilai dari variabel tersebut akan dilewatkan sebagai nilai dari function ini.

Function GetRed

```
Function TPhotoStereo.GetRed  (Pix : Integer) : Integer;
Begin
    GetRed := ((Pix Shr 16) And $000000ff);
End;
```

Function GetGreen

```
Function TPhotoStereo.GetGreen(Pix : Integer) : Integer;
Begin
    GetGreen := ((Pix Shr 8) And $000000ff);
End;
```

Function GetBlue

```
Function TPhotoStereo.GetBlue (Pix : Integer) : Integer;
Begin
    GetBlue := (Pix And $000000ff);
End;
```

Ketiga function di atas mempunyai kegunaan yang hampir sama, yaitu untuk memisahkan warna primer yang dikirim lewat parameter oleh proses yang memanggilnya. Perbedaanya terletak pada warna primer apa yang dihitung kadarnya dari parameter tersebut. Masing-masing Function GetRed, GetGreen dan GetBlue akan memberikan kadar warna merah, hijau dan biru sebagai nilai dari function yang bersangkutan untuk setiap parameternya.

Procedure Init

```
Procedure TPhotoStereo.Init;
Begin
    Cursor := crHourGlass;
    Application.ProcessMessages;

    GridSize := 3;
    VecLength := 5;

    GetImageIntensity;
    SetLightSource;
    Psm;
    isInit := True;
End;
```

Procedure Init ini dipergunakan untuk memberikan nilai awal atau inisialisasi terhadap beberapa variabel yang memerlukan suatu nilai awal tertentu untuk proses selanjutnya. Pada procedure ini juga dilakukan pemanggilan beberapa procedure-procedure utama yang terdapat pada class TPhotoStereo ini yaitu *GetImageIntensity, SetLightSource* dan *Psm* yang akan melakukan beberapa proses yang utama dalam aplikasi ini.

Procedure PSM

```
Procedure TPhotoStereo.PSM;
var X,Y : Integer;
    V,V1,V2,V3,V4,V11,V22,V12 : Vec3d;
Begin

    V := Vec3d.Create;
    V1:= Vec3d.Create;
    V2:= Vec3d.Create;
    V3:= Vec3d.Create;

    SetLength(Grad,W,H);
    SetLength(CoordX,W,H);
    SetLength(CoordY,W,H);
    SetLength(CoordZ,W,H);
    ImageWidth  := W;
    ImageHeight := H;
    For X := 0 To W-1 Do
        For Y := 0 To H-1 Do
        Begin
            Grad[X,Y] := Vec3d.Create;
            V1 := V.Times(E2[X,Y]*S2.Norm,S1);
            V2 := V.Times(E1[X,Y]*S1.Norm,S2);
            V11 := V.Minus(V1,V2);

            V3 := V.Times(E3[X,Y]*S3.Norm,S1);
            V4 := V.Times(E1[X,Y]*S1.Norm,S3);
            V22 := V.Minus(V3,V4);

            Grad[X,Y] := V.Cross(V11,V22);
            Grad[X,Y].Normalise;

            If Grad[X,Y].GetZ > 0 Then Grad[X,Y].Negate;
            MakeList(V1,V2,V3,V4,Grad[X,Y]);

            CoordX[X,Y] := Grad[X,Y].GetX;
```

```
            CoordY[X,Y] := Grad[X,Y].GetY;
            If Grad[X,Y].GetZ > 0 Then
               CoordZ[X,Y] := -1*Grad[X,Y].GetZ
            Else
               CoordZ[X,Y] := 1*Grad[X,Y].GetZ;
         End;
End;
```

Procedure PSM ini adalah merupakan procedure utama dalam class TPhotoStereo ini. Karena pada procedure inilah algoritma *photometric stereo* untuk menentukan intensitas cahaya yang dipantulkan oleh masing-masing titik pada permukaan suatu objek diterjemahkan ke dalam bahasa pemrograman. Dari data tentang intensitas cahaya tiap titik pada ketiga citra masukan dan data posisi sumber cahaya yang menyinari obyek pada masing-masing citra, maka procedure ini mencari bidang-bidang kecil yang merupakan elemen-elemen kecil pembentuk permukaan yang nantinya bidang ini akan digunakan untuk normal mencari vektor pada tiap-tiap titik pada permukaan sebuah objek. Kemudian proses ini akan menghitung vektor normal untuk masing-masing bidang kecil tersebut.

Procedure Run

```
Procedure TPhotoStereo.Run;
Var Ls : Vec3d;
Begin
   NeedleMap;
   If Not isInit Then
      Ls := Vec3d.Create;
   Ls := Vec3D.Create;
```

```
    Ls.SetValue(5,5,-10);
    LambertShade(Ls);
End;
```

Procedure NeedleMap

```
Procedure TPhotoStereo.NeedleMap;
Var X,Y,xx,yy,Step : Integer;
Begin
    Step:=(MainForm.Image1.Width )Div 32;
    MapForm.Image1.Canvas.Pen.Color := clBlue;
    X:=0; Y := 0;
    MapForm.Image1.Width := W; MapForm.Image1.Height := H;
    MapForm.Image2.Width := W; MapForm.Image2.Height := H;

    MapForm.Image1.Canvas.Brush.Color := clYellow;
    MapForm.Image1.Canvas.Rectangle(0,0,W,H);
    While X<W Do
    Begin
      Y := 0;
      While Y<H Do
      Begin
         xx := x + Step;
         yy := Y + Step;
        MapForm.Image1.Canvas.MoveTo(xx,yy);
        MapForm.Image1.Canvas.LineTo(xx+
                  Trunc(Grad[X,Y].GetX*20),
                  yy-Trunc(Grad[X,Y].GetY*20));
       MapForm.Image1.Canvas.Rectangle(XX-1,YY-1,XX+2,YY+2);
       Inc(Y,Step);
      End;
      Inc(X,Step);
      Application.ProcessMessages;
    End;
End;
```

Pemanggilan procedure *NeedleMap* ini akan menghasilkan gambar garis-garis yang merupakan peta yang menunjukkan arah vektor normal pada tiap titik kontrol pada permukaan yang telah dihitung pada procedure PSM yang telah dijelaskan sebelumnya. Gambar garis-garis ini adalah gambar dua dimensi yang memperlihatkan arah vektor normal dalam ruang tiga dimensi apabila kamera berada di depan permukaan yang telah direkonstruksi.

Procedure SetLightSource;

```
Procedure TPhotoStereo.SetLightSource;
Begin
    if Not isInit Then
    Begin
      S1 := Vec3d.Create;
      S2 := Vec3d.Create;
      S3 := Vec3d.Create;
    End;
    ImageForm.SetProperties;
    With ImageForm Do
    Begin
        S1.SetValue(sx1,sy1,sz1);
        S2.SetValue(sx2,sy2,sz2);
        S3.SetValue(sx3,sy3,sz3);
    End;
End;
```

Procedure *SetLightSource* ini berisi proses-proses yang berguna untuk mengisi variabel-variabel yang menyimpan koordinat-koordinat posisi sumber cahaya yang menyinari obyek dari tiga arah yang berbeda, dengan nilai-nilai yang telah

dimasukkan oleh pemakai melalui *Form PropertiesForm* dan disimpan pada variabel-variabel yang terletak pada *unit ImageUnit*.

Procedure GetImageIntensity;

```
Procedure TPhotoStereo.GetImageIntensity;
Begin
    E1 := GetIntensity(MainForm.Image1);
    E2 := GetIntensity(MainForm.Image2);
    E3 := GetIntensity(mainForm.Image3);
End;
```

Procedure *GetImageIntensity* ini hanya melakukan tiga kali pemanggilan terhadap *Function GetIntensity* dengan tiga parameter yang berbeda-beda, yaitu ketiga buah citra yang dijadikan masukan untuk aplikasi ini. Nilai yang diberikan setiap pemanggilan fungsi tersebut dimasukkan ke variabel *E1*, *E3* dan *E3* yang masing-masing digunakan untuk menyimpan tabel intensitas cahaya titik-titik pada setiap citra.

Procedure LambertShade

```
Procedure TPhotoStereo.LambertShade(Ls : Vec3d);
Var Intens : Real;
    GradNorm : Real;
    X,Y      : Integer;
    V        : Vec3D;
    GrayC    : TColor;
Begin
    V := Vec3d.Create;
    MapForm.Image2.Width := W-1;
    MapForm.Image2.Height:= H-1;
    MapForm.Image2.Canvas.Brush.Color := clYellow;
```

```
    MapForm.Image2.Canvas.Rectangle(0,0,
            MapForm.Image2.Width,MapForm.Image2.Height);
    Y := 0;
    While Y<(H-1) Do
    For Y := 0 To H-1 Do
    Begin
        X := 0;
        While X<(W-1) Do
        For X := 0 To W-1 Do
        Begin
            GradNorm := Grad[X,Y].Norm;
            If GradNorm < 0.00001 Then
               Intens:=0
            Else
               Intens:= V.Dot(Grad[X,Y],Ls)/
                              (Grad[X,Y].Norm*Ls.Norm);
            If Intens < 0 Then Intens := 0;
            If Intens > 1 Then Intens := 1;
            GrayC := Round(Intens * 255);
            MapForm.Image2.Canvas.Pixels[X,Y]:=
                           RGB(GrayC,GrayC,GrayC);
            CoordZ[X,Y] := -GrayC;
            Inc(X,1);
        End;
        Inc(Y,1);
        Application.ProcessMessages;
    End;
End;
```

Procedure *LambertShade* ini akan menampilkan permukaan hasil rekonstruksi dalam bentuk gambar dua dimensi dalam format level keabu-abuan. Procedure ini memanfaatkan arah vektor normal pada permukaan untuk mencari nilai dari level keabu-abuan setiap titik pada permukaan yang telah terbentuk.

```
Procedure MakeList(V1,V2,V3,V4,Grad : Vec3D);
Procedure TPhotoStereo.MakeList(V1,V2,V3,V4,Grad : Vec3D);
Var N1,N2,N3,N4,Norm : TCoord3D;
Begin
     Norm.X:=Grad.GetX; Norm.Y:=Grad.GetY; Norm.Z:=Grad.GetZ;
     N1.X:=V1.GetX;   N1.Y:=V1.GetY;   N1.Z:=V1.GetZ;
     N2.X:=V2.GetX;   N2.Y:=V2.GetY;   N2.Z:=V2.GetZ;
     N3.X:=V3.GetX;   N3.Y:=V3.GetY;   N3.Z:=V3.GetZ;
     N4.X:=V4.GetX;   N4.Y:= V4.GetY;  N4.Z:=V4.GetZ;
     AddToNormList(N1,N2,N3,N4,Norm);
End;
```

Procedure *Makelist* yang ditulis di atas akan memasukkan parameter-parameternya yang merupakan koordinat bidang-bidang kecil pada permukaan dan vektor normalnya ke dalam tabel yang berbentuk linked list dengan menggunakan procedure *AddToNormalList*. Procedure ini diperlukan karena procedure AddToList tersebut semua parameternya meskipun jumlahnya sama akan tetapi semuanya bertipe Tcoord3D, yang berbeda dari Vec3D sehingga procedure ini berfungsi untuk mengambil nilai-nilai yang diperlukan dari tipe Vec3D untuk dimasukkan ke tabel tersebut.

Untuk selanjutnya akan dibahas proses-proses yang terdapat pada class Vec3D. Sesuai dengan yang telah kita bahas pada bagian sebelumnya bahwa class ini dipergunakan untuk menyimpan dan melakukan operasi-operasi vektor pada ruang tiga dimensi. Adapun implementasi proses dari class Vec3D tersebut adalah seperti di bawah ini.

Function GetX

```
Function Vec3d.GetX : Real;
Begin
    GetX := X;
End;
```

Function GetY

```
Function Vec3d.GetY : Real;
Begin
    GetY := Y;
End;
```

Function GetZ

```
Function Vec3d.GetZ : Real;
Begin
    GetZ := Z;
End;
```

Ketiga function yang ditulis di atas dipergunakan untuk mengambil nilai dari masing-masing elemen vektor pada ruang tiga dimensi. Misalkan untuk vektor $v = (x, y, z)$, maka setiap pemanggilan function GetX akan memberikan nilai x, pemanggilan function GetY akan memberikan nilai y dan begitu juga dengan pemanggilan function GetZ akan memberikan nilai z. Ketiga function di atas diperlukan karena variabel-variabel yang menyimpan nilai-nilai x, y, dan z adalah merupakan anggota dari class Vec3D yang bersifat *private* sehingga hanya dapat diakses (ditulisi dan dibaca) oleh proses-proses yang merupakan anggota class tersebut.

Function Times

```
Function Vec3d.Times(C : Real; Vec : Vec3d) : Vec3d;
Var Tmp : Vec3d;
Begin
    Tmp := Vec3d.Create;
    Tmp.X := C * Vec.X;
    Tmp.Y := C * Vec.Y;
```

```
    Tmp.Z := C * Vec.Z;
    Times := Tmp;
End;
```

Function Times pada class Vec3D dipergunakan untuk mencari hasil perkalian sebuah vektor *Vec* dengan sebuah bilangan skalar *C*. Function ini akan menghasilkan sebuah vektor lain sebagai hasil perkaliannya.

Function Minus

```
Function Vec3d.Minus(a,b : Vec3d) : Vec3d;
Var Tmp : Vec3d;
Begin
    Tmp := Vec3d.Create;
    Tmp.X := a.X - b.X;
    Tmp.Y := a.Y - b.Y;
    Tmp.Z := a.Z - b.Z;
    Minus := Tmp;
End;
```

Function minus di atas dipergunakan untuk mencari selisih dua buah vektor pada ruang tiga dimensi. Kedua parameter yang menyertai pemanggilan fungsi ini merupakan kedua buah vektor yang akan dicari selisihnya. Sebelum dikeluarkan sebagai nilai dari fungsi, selisih vektor ***a*** dan ***b*** terlebih dahulu disimpan di variabel Tmp yang juga bertipe Vec3D.

Function Norm

```
Function Vec3d.Norm : Real;
Begin
    Norm := Sqrt(X*X + Y*Y + Z*Z);
End;
```

Fungsi di atas berfungsi untuk mencari panjang dari vektor yang bersangkutan, di mana panjang vektor adalah merupakan akar dari jumlah kuadrat masing-masing elemen.

Function Dot

```
Function Vec3d.Dot(a,b : Vec3d) : Real;
Begin
    Dot := (a.X*b.X + a.Y*b.Y + a.Z*b.Z);
End;
```

Function Cross

```
Function Vec3d.Cross(a,b : Vec3d) : Vec3d;
Var Tmp : Vec3d;
Begin
    Tmp := Vec3d.Create;
    Tmp.X := a.Y*b.Z - a.Z*b.Y;
    Tmp.Y := a.Z*b.X - a.X*b.Z;
    Tmp.Z := a.X*b.Y - a.Y*b.X;
    Cross := Tmp;
End;
```

Kedua fungsi di atas sama-sama berfungsi untuk melakukan operasi terhadap dua buah vektor ***a*** dan ***b***. Fungsi *Dot* berfungsi untuk mencari hasil kali titik vektor-vektor yang bersangkutan dan sesuai dengan yang telah kita bahas pada Bab II, hasil kali titik dua buah vektor akan menghasilkan bilangan sekalar. Sedangkan fungsi *Cross* adalah merupakan fungsi untuk mencari hasil kali silang antara dua buah vektor. Hasil kali silang dua buah vektor ini adalah sebuah vektor yang tegak lurus kedua vektor semula.

Procedure SetValue

```
Procedure Vec3d.SetValue(XVal,YVal,ZVal : Real);
Begin
    X := XVal;
    Y := YVal;
    Z := ZVal;
End;
```

Procedure SetValue merupakan kebalikan dari fungsi-fungsi GetX, GetY dan GetZ yang mengambil nilai elemen-elemen vektor yang bersangkutan. Procedure ini memberikan suatu nilai kepada elemen-elemen vektor *x*, *y* dan *z* dengan nilai dari parameter-parameter yang menyertai pada waktu pemanggilan procedure tersebut, yaitu berturut-turut *XVal*, *YVal* dan *ZVal*.

Procedure Negate

```
Procedure Vec3d.Negate;
Begin
    X := -X;
    Y := -Y;
    Z := -Z;
End;
```

Procedure Negate ini dipergunakan untuk mencari negasi atau kebalikan dari vektor yang bersangkutan dengan cara mengalikan masing-masing elemen vektor dengan bilangan –1.

Procedure Normalise

```
Procedure Vec3d.Normalise;
Var n : Real;
Begin
    n := Norm;
    If n < 0.00001 Then
        SetValue(0,0,0)
    Else
```

```
        SetValue(x/n,y/n,z/n);
End;
```

Procedure terakhir dari class Vec3D ini dipergunakan untuk mencari vektor satuan dari vektor yang bersangkutan. Vektor satuan ini didapatkan dengan cara membagi tiap elemen vektor yang bersangkutan dengan panjang vektor itu sendiri.

BAB V

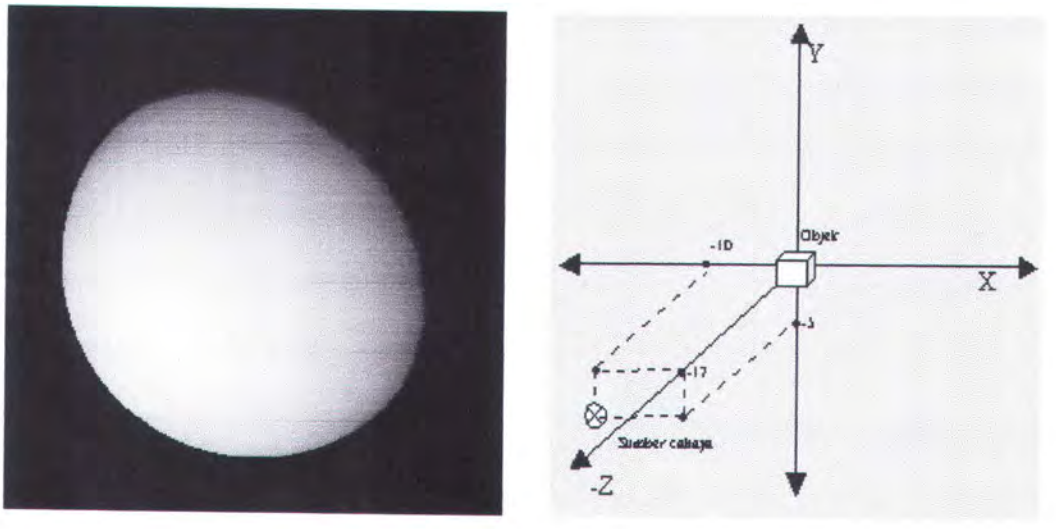
HASIL UJI COBA PERANGKAT LUNAK

BAB V

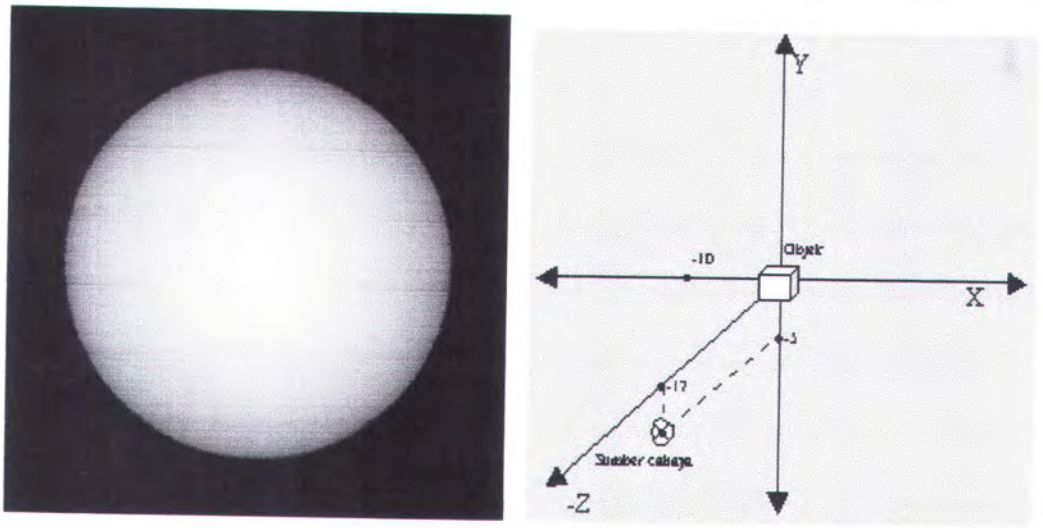
HASIL UJI COBA PERANGKAT LUNAK

Pada bab ini akan dipaparkan hasil dari ujicoba perangkat lunak yang telah dirancang. Untuk melakukan ujicoba perangkat lunak ini, sudah disiapkan enam buah citra dalam format gray level yang merupakan gambar dari dua buah objek yang berbeda. Masing-masing objek diwakili oleh tiga buah citra dengan posisi sumber cahaya yang berbeda-beda.

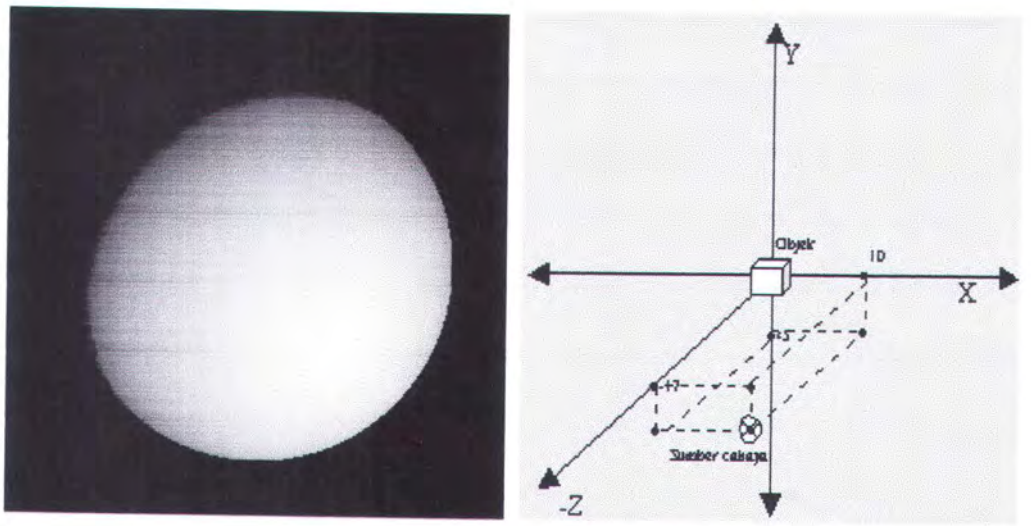
Untuk uji coba yang pertama, gambar yang dipakai sebagai masukan dapat dilihat pada gambar 5.1 (a), 5.1 (b) dan 5.1(c). Masing-masing gambar tersebut merupakan gambar dari sebuah objek berbentuk bola yang mendapatkan sinar dari sebuah sumber cahaya yang posisinya diubah-ubah dan ditangkap oleh kamera yang berada di depan objek tersebut.



(a). Sumber cahaya ada di (-10, -5, -17)



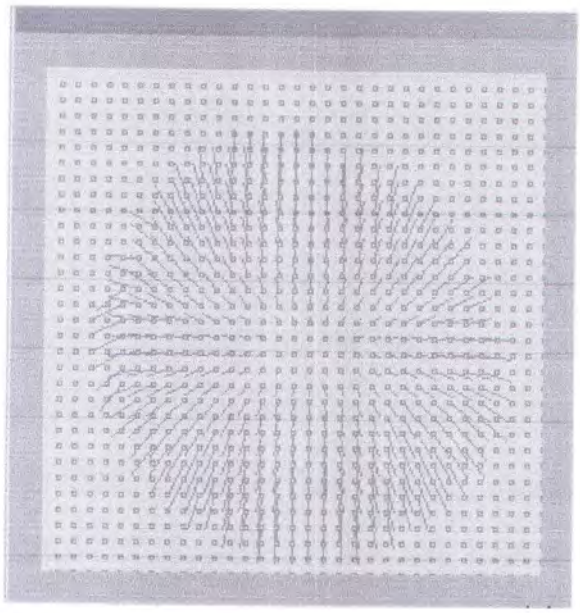
(b).Sumber cahaya ada di (0, -5, -17)



(c).Sumber cahaya ada di (10, -5, -17)

Gambar 5.1 Objek berbentuk bola yang mendapat sinar dari 3 arah yang berbeda dan peta posisi objek serta sumber cahaya pada sistem koordinat ruang 3D.

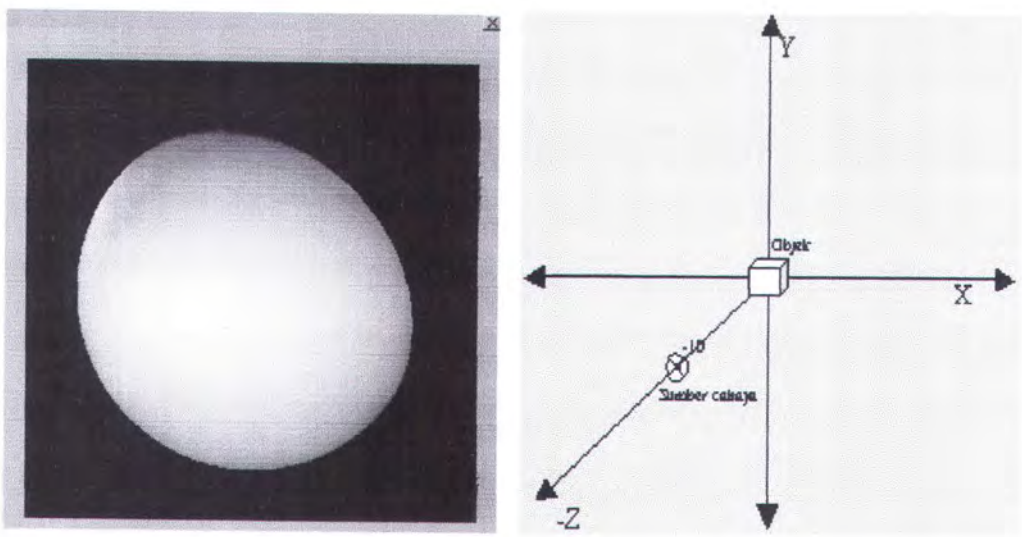
Untuk mencari vektor normal tiap-tiap titik dari permukaan pada objek di atas dengan menggunakan perangkat lunak yang telah dibuat, maka terlebih dahulu harus dibuat sebuah *project* yang isinya adalah informasi tentang nama-nama file gambar yang dipakai dan informasi tentang posisi sumber cahaya pada masing-masing gambar tersebut. Untuk masukan berupa gambar di atas, maka peta arah vektor normal yang terbentuk dapat dilihat pada gambar 5.2.



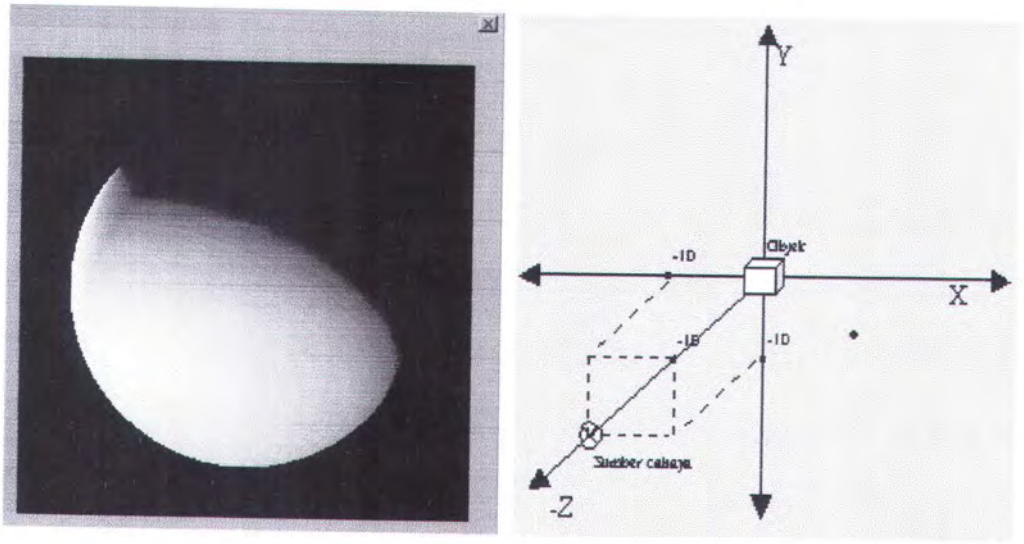
Gambar 5.2. Peta arah vektor normal pada permukaan objek berbentuk bola.

Dari data tentang vektor normal pada permukaan objek tersebut, perangkat lunak ini akan menghitung intensitas cahaya yang dipantulkan oleh tiap-tiap titik pada

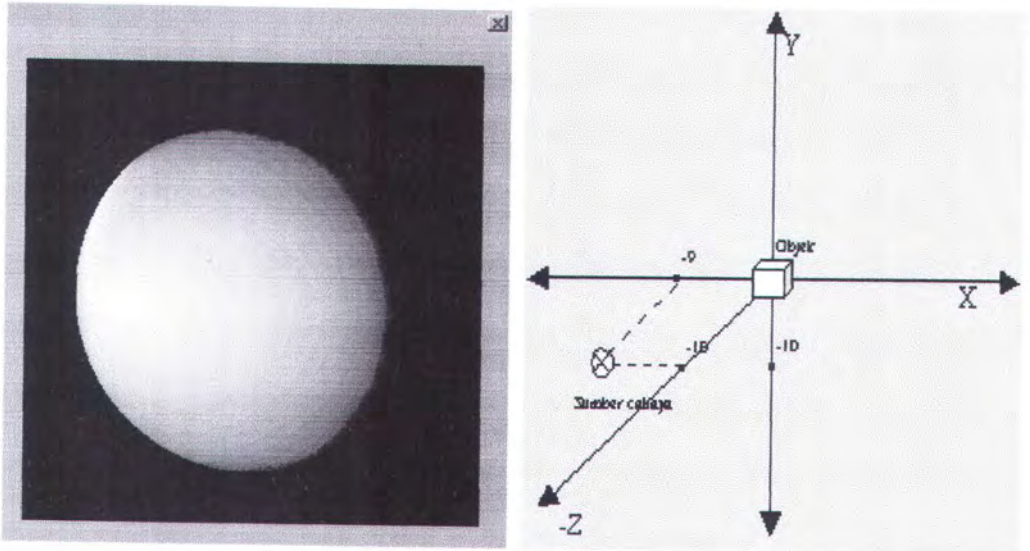
permukaan objek yang tertangkap oleh kamera jika objek mendapatkan cahaya dari sebuah sumber cahaya dari arah tertentu. Gambar-gambar 5.3 menunjukkan gambar yang ditangkap oleh kamera jika objek diberikan cahaya dari beberapa arah yang berbeda.



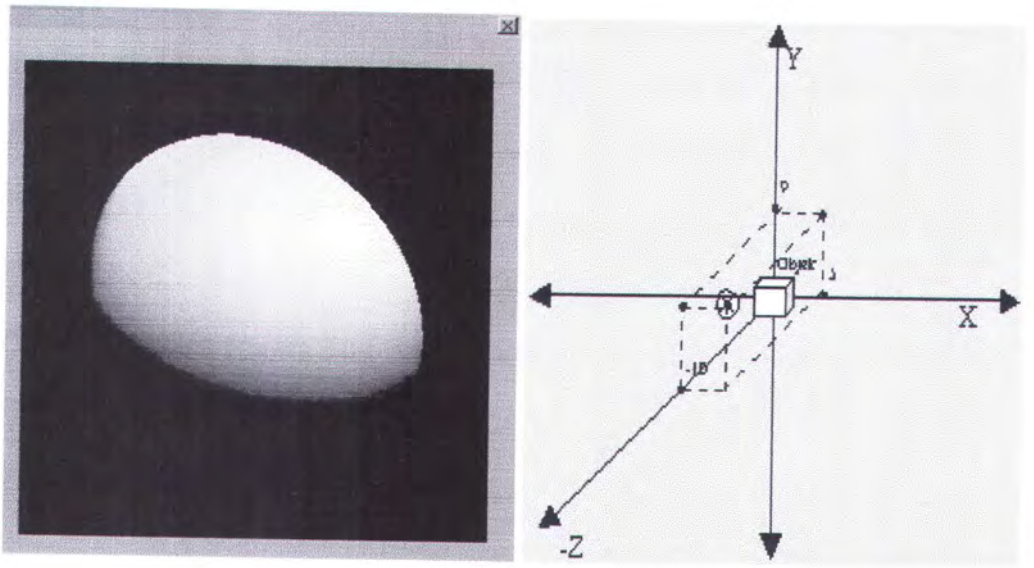
Gambar 5.3 (a). Objek mendapat cahaya dari arah depan.



Gambar 5.3 (b). Objek mendapat cahaya dari sebelah kiri bawah objek, yaitu dari koordinat (-10, -10, -10)

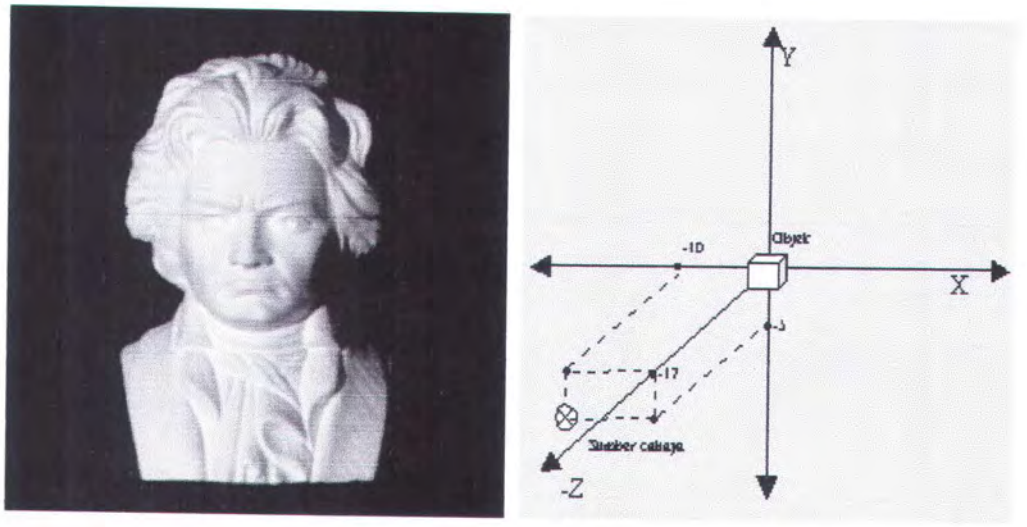


Gambar 5.3 (c). Objek mendapat cahaya dari sebelah kiri objek, yaitu dari koordinat (-9, 0, -10)

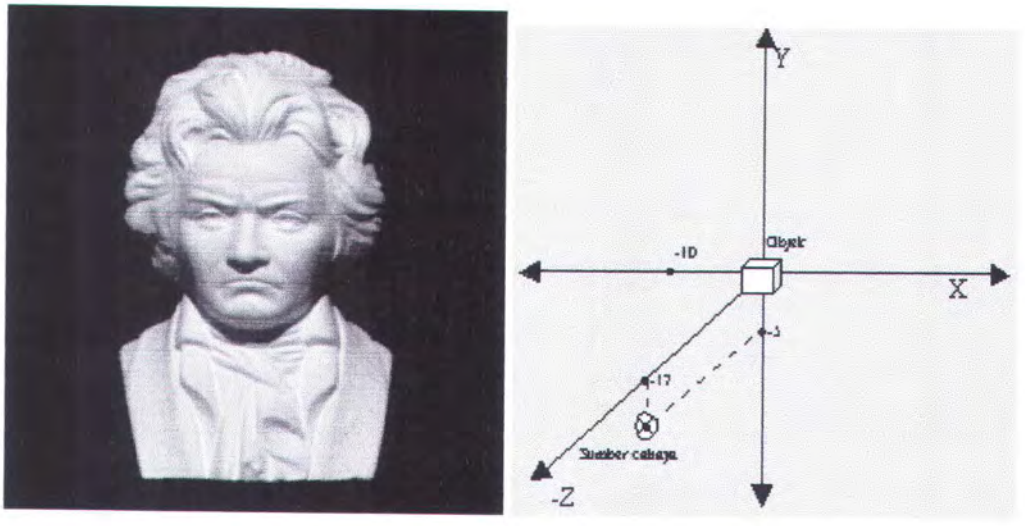


Gambar 5.3 (d). Objek mendapat cahaya dari sebelah kanan atas objek, yaitu dari koordinat (5, 9, -10)

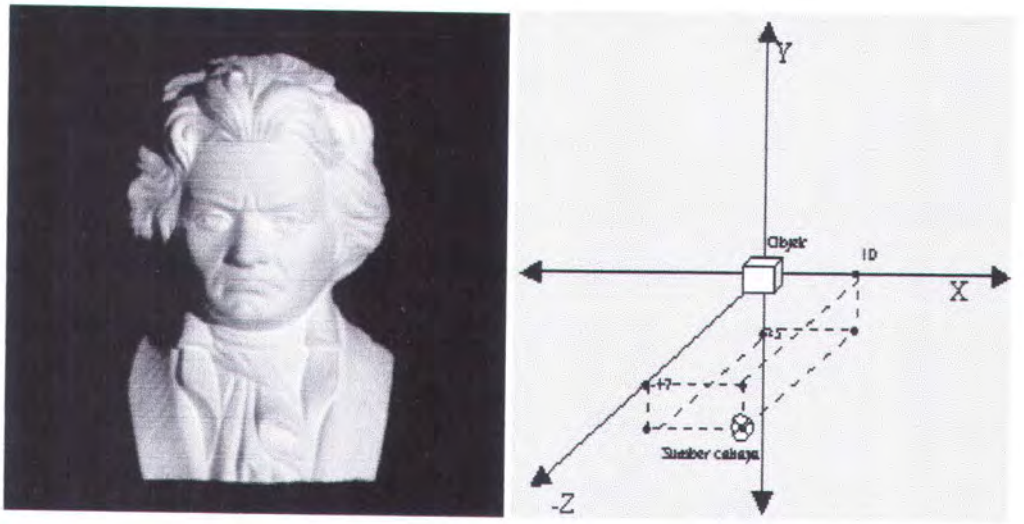
Untuk uji coba selanjutnya, gambar yang dipakai sebagai masukan untuk perangkat lunak ini adalah gambar di bawah ini.



(a). Sumber cahaya ada di (-10, -5, -17)



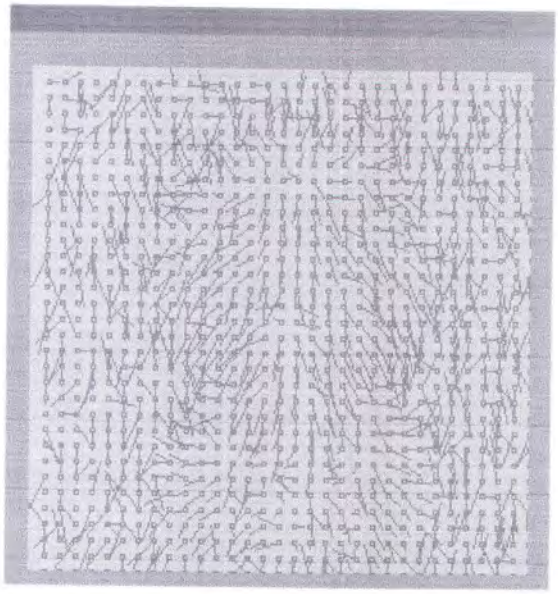
(b).Sumber cahaya ada di (0,- 5, -17)



(c).Sumber cahaya ada di (10, -5, -17)

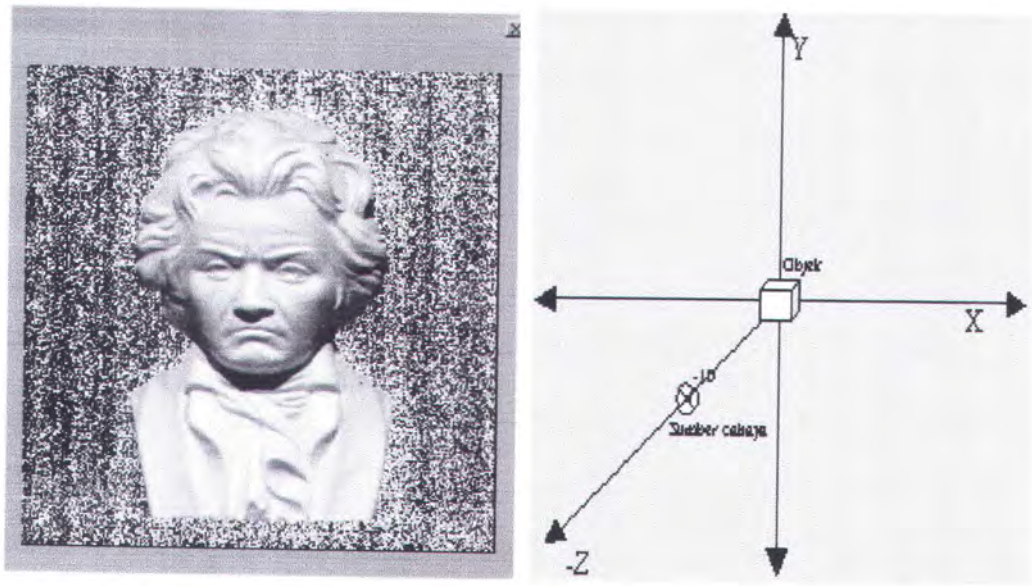
Gambar 5.4. Hasil Scan terhadap foto sebuah patung logam yang mendapat sinar dari tiga arah yang berbeda dan peta posisi objek serta sumber cahaya pada sistem koordinat ruang 3D.

Dari gambar-gambar masukkan di atas, hasil perhitungan vektor normalnya ditunjukkan pada gambar 5.5 di bawah ini.

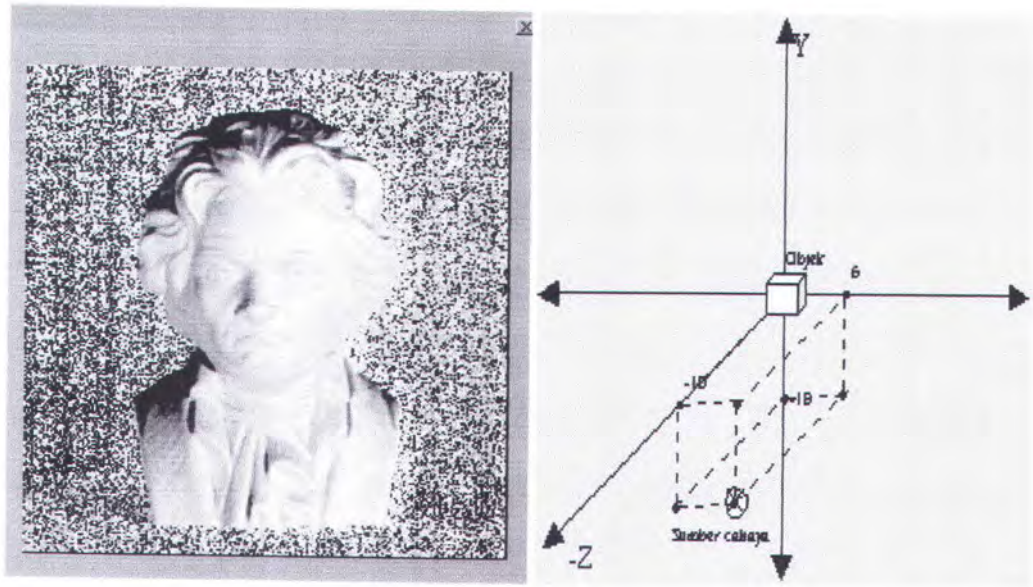


Gambar 5.5. Peta vektor normal untuk masukkan pada gambar 5.4.

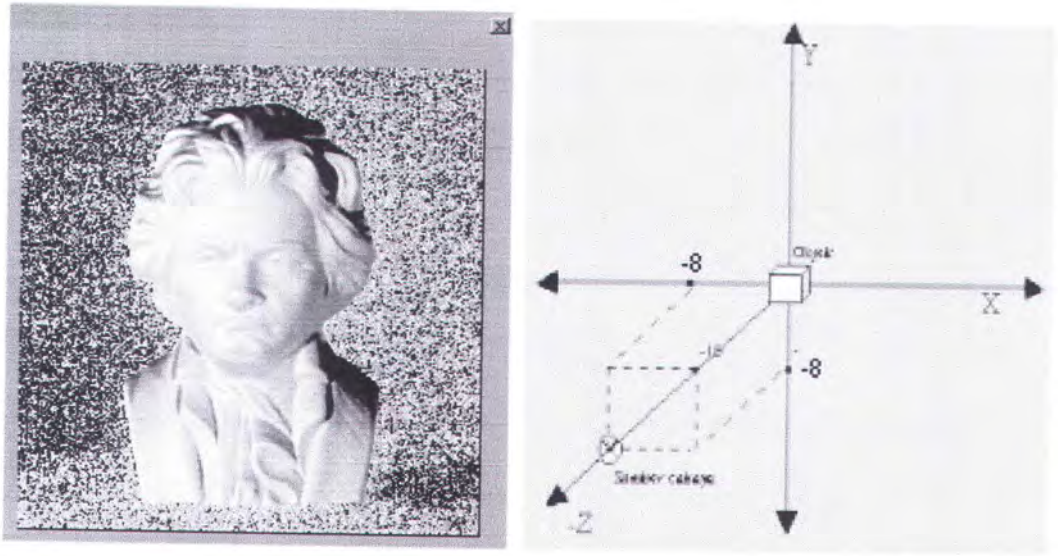
Berdasarkan peta vektor normal yang telah didapatkan, maka gambar yang ditangkap oleh kamera yang berada di depan objek jika objek tersebut mendapat sinar dari sebuah sumber cahaya dari beberapa posisi adalah seperti yang terlihat pada gambar 5.6 di bawah ini.



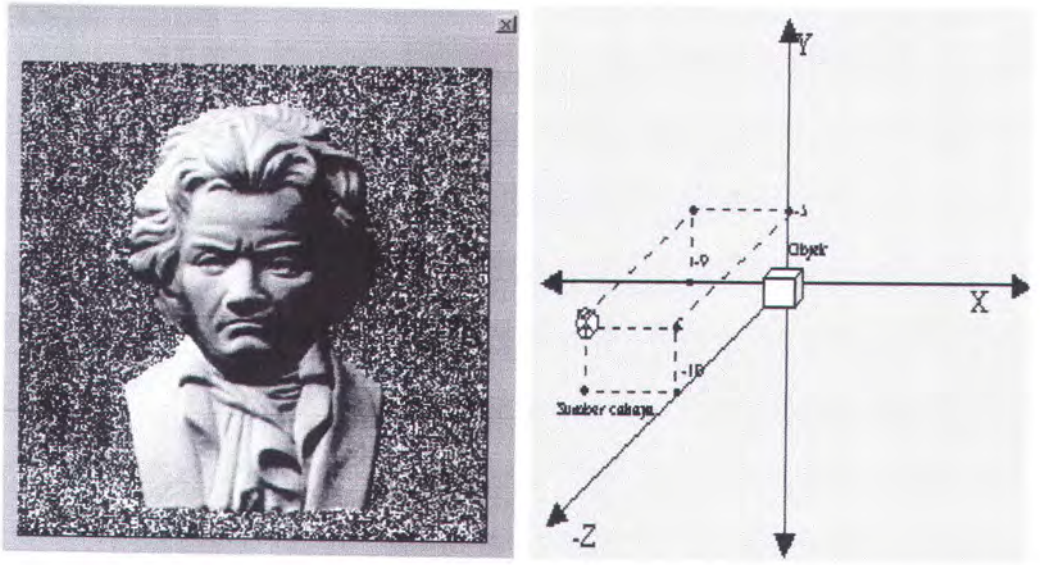
Gambar 5.6 (a). Objek mendapat cahaya dari arah depan, yaitu dari koordinat (0, 0, -10)



Gambar 5.6 (b). Objek mendapat cahaya dari arah kanan bawah, yaitu dari koordinat (6, -10, -10)



Gambar 5.6 (c). Objek mendapat cahaya dari arah kiri bawah, yaitu dari koordinat (-8, -8, -10)



Gambar 5.6 (c). Objek mendapat cahaya dari arah kiri atas, yaitu dari koordinat (-9, 5, -10)

Dari kedua uji coba tersebut di atas, dapat dilihat ada sebuah perbedaan pada hasil akhir dari perangkat lunak yang dipakai. Pada percobaan pertama tidak terlihat adanya *noise* pada bagian latar belakang objek yang terbentuk, sedangkan pada percobaan kedua latar belakang dari objek yang terbentuk terdapat *noise* yang membuat warna latar belakang tidak menjadi hitam sempurna. Hal ini terjadi karena pada percobaan kedua, gambar yang dipakai merupakan hasil scan terhadap foto sebuah patung logam, sehingga warna hitam pada latar belakangnya tidak merata akan tetapi terdiri dari beberapa degradasi warna yang masih terlihat hitam.

BAB VI

KESIMPULAN DAN SARAN

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Algoritma *Photometric Stereo* adalah algoritma yang cukup sederhana dan mempunyai hasil yang cukup baik untuk menentukan efek pencahayaan terhadap sebuah objek. Algoritma ini akan memberikan hasil berupa vektor normal permukaan yang benar jika pada citra input yang digunakan kita mengetahui dengan pasti posisi sumber cahaya yang menyinari objek tersebut. Hal ini dapat kita lihat pada hasil uji coba dengan menggunakan objek yang kedua (lihat Bab V, gambar 5.4a, 5.4b dan 5.4c). Karena posisi cahaya untuk setiap citra masukan diketahui secara pasti, maka algoritma *Photometric Stereo* ini akan memberikan hasil yang benar.

Akan tetapi jika posisi sumber cahaya yang menyinari objek pada citra input tidak diketahui dengan pasti dan kita menentukan posisi sumber cahaya dengan memprediksikan yaitu dengan cara melihat bagian gambar yang terang yang ditangkap oleh kamera, maka vektor normal yang didapatkan juga tidak akan tepat (vektor normal yang didapatkan hanya hanya mendekati nilai yang sebenarnya). Seperti yang kita lihat gambar 6.1 dibawah, yang merupakan hasil uji coba yang kedua dengan menggunakan objek yang pertama (lihat Bab V, gambar 5.1a, 5.1b dan 5.1c).

Gambar 6.1 Kesalahan pada hasil uji coba

Pada bagian kiri atas objek terdapat bagian yang terang, padahal objek mendapat cahaya dari arah bawah sebelah kiri, sehingga bagian atas tari objek seharusnya berwarna gelap karena tidak mendapat cahaya. Hal ini terjadi karena posisi sumber cahaya yang diberikan pada input dari aplikasi ini tidak tepat. Dengan demikian dapat kita lihat kelemahan dari algoritma *Photometric Stereo* ini, yaitu tidak bisa menentukan arah datangnya cahaya sehingga terkadang memberikan hasil yang tidak akurat.

6.2 Saran

Aplikasi yang dibuat dengan menggunakan algoritma *Photometric Stereo* ini masih mempunyai kemungkinan untuk dikembangkan lebih lanjut. Hal yang dapat dikembangkan dari algoritma ini adalah pencarian vektor normal pada objek yang permukaannya terdiri dari berbagai bahan dan warna yang berbeda. Hal ini dimungkinkan

dengan melibatkan koefisien refleksi elemen permukaan dalam perhitungannya. Seperti yang telah dibahas pada Bab III, intensitas cahaya yang dipantulkan oleh permukaan suatu objek selain dipengaruhi oleh arah sinar datang dan arah vektor normal , juga dipengaruhi oleh koefisien refleksi permukaan, akan tetapi dalam pembuatan tugas akhir ini objek yang digunakan sebagai input terdiri dari bahan dan warna yang sama sehingga koefisien refleksi pada tiap-tiap titik pada permukaan tidak menimbulkan variasi dalam memantulkan cahaya yang diterima.

Algoritma *Photometric Stereo* ini hanya bisa digunakan terbatas untuk menemukan vektor normal pada permukaan. Kemungkinan selanjutnya yang bisa dikembangkan adalah membuat sebuah aplikasi yang tidak hanya bisa menemukan vektor normal pada permukaan, akan tetapi juga bisa digunakan untuk menemukan koordinat dari permukaan tersebut. Sehingga aplikasi yang dibuat tidak hanya bisa menentukan efek pencahayaan pada sebuah objek dengan posisi kamera berada di depan objek tersebut, akan tetapi kamera bisa berada pada posisi tertentu yang arahnya ditunjukkan oleh sebuah vektor.

LAMPIRAN

Lampiran A

Contoh Hasil Rekonstruksi dalam Angka

1. Data untuk citra yang pertama :

   Posisi sumber cahaya : (-10, 0, -5)

   Data gray scale dari citra :

0	0	0	0	200	50	0	0	0	0
0	0	0	200	200	50	50	0	0	0
0	0	200	200	200	50	50	50	0	0
0	200	200	200	200	50	50	50	50	0
200	200	200	200	200	50	50	50	50	50
200	200	200	200	200	50	50	50	50	50
0	200	200	200	200	50	50	50	50	0
0	0	200	200	200	50	50	50	0	0
0	0	0	200	200	50	50	0	0	0
0	0	0	0	200	50	0	0	0	0

   Intensitas cahaya pada masing-masing pixel yang didapatkan :

0	0	0	0	26	6.5	0	0	0	0
0	0	0	26	26	6.5	6.5	0	0	0
0	0	26	26	26	6.5	6.5	6.5	0	0
0	26	26	26	26	6.5	6.5	6.5	6.5	0
26	26	26	26	26	6.5	6.5	6.5	6.5	6.5
26	26	26	26	26	6.5	6.5	6.5	6.5	6.5
0	26	26	26	26	6.5	6.5	6.5	6.5	0
0	0	26	26	26	6.5	6.5	6.5	0	0
0	0	0	26	26	6.5	6.5	0	0	0
0	0	0	0	26	6.5	0	0	0	0

2. Data untuk citra yang kedua :

   Posisi sumber cahaya : (0, 0, -5)

   Data gray scale dari citra :

0	0	0	0	200	200	0	0	0	0
0	0	0	200	200	200	200	0	0	0
0	0	200	200	200	200	200	200	0	0
0	200	200	200	200	200	200	200	200	0
200	200	200	200	200	200	200	200	200	200
50	50	50	50	50	50	50	50	50	50
0	50	50	50	50	50	50	50	50	0
0	0	50	50	50	50	50	50	0	0
0	0	0	50	50	50	50	0	0	0
0	0	0	0	50	50	0	0	0	0

Intensitas cahaya pada masing-masing pixel yang didapatkan :

0	0	0	0	26	26	0	0	0	0
0	0	0	26	26	26	26	0	0	0
0	0	26	26	26	26	26	26	0	0
0	26	26	26	26	26	26	26	26	0
26	26	26	26	26	26	26	26	26	26
6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5
0	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	0
0	0	6.5	6.5	6.5	6.5	6.5	6.5	0	0
0	0	0	6.5	6.5	6.5	6.5	0	0	0
0	0	0	0	6.5	6.5	0	0	0	0

3. Data untuk citra yang ketiga :

Posisi sumber cahaya : (10, 0, -5)

Data gray scale dari citra :

0	0	0	0	50	200	0	0	0	0
0	0	0	50	50	200	200	0	0	0
0	0	50	50	50	200	200	200	0	0
0	50	50	50	50	200	200	200	200	0
50	50	50	50	50	200	200	200	200	200
50	50	50	50	50	200	200	200	200	200
0	50	50	50	50	200	200	200	200	0
0	0	50	50	50	200	200	200	0	0
0	0	0	50	50	200	200	0	0	0
0	0	0	0	50	200	0	0	0	0

Intensitas cahaya pada masing-masing pixel yang didapatkan :

0	0	0	0	6.5	26	0	0	0	0
0	0	0	6.5	6.5	26	26	0	0	0
0	0	6.5	6.5	6.5	26	26	26	0	0
0	6.5	6.5	6.5	6.5	26	26	26	26	0
6.5	6.5	6.5	6.5	6.5	26	26	26	26	26
6.5	6.5	6.5	6.5	6.5	26	26	26	26	26
0	6.5	6.5	6.5	6.5	26	26	26	26	0
0	0	6.5	6.5	6.5	26	26	26	0	0
0	0	0	6.5	6.5	26	26	0	0	0
0	0	0	0	6.5	26	0	0	0	0

Vektor normal yang didapatkan dari ketiga data di atas adalah sebagai berikut :

(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(-0.7, 0.7, -0.7)
(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)
(0, 0, 0)	(0, 0, 0)	(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)
(0, 0, 0)	(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)
(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)	(-0.7, 0.7, -0.7)
(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)
(0, 0, 0)	(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)
(0, 0, 0)	(0, 0, 0)	(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)
(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(-0.7, -0.7, -0.7)	(-0.7, -0.7, -0.7)
(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(-0.7, -0.7, -0.7)

(0.7, 0.7, -0.7)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)	(0, 0, 0)	(0, 0, 0)
(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)	(0, 0, 0)
(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)	(0.7, 0.7, -0.7)
(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)
(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)	(0, 0, 0)
(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)	(0, 0, 0)	(0, 0, 0)
(0.7, -0.7, -0.7)	(0.7, -0.7, -0.7)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
(0.7, -0.7, -0.7)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)

DAFTAR PUSTAKA

DAFTAR PUSTAKA

Francis, Hill, *Computer Graphics*, MacMilan Publishing Company, New York, 1990.

Anton, Howard, *Aljabar Linier Elementer*, pernerbit Erlangga, Jakarta, 1991

R.J. Woodhan, *Photometric Stereo : Lambertian Reflectance and Light Source with Unknown Direction and Strength*, Laboratory for Computational Intelligence University of British Columbia Vancouver, BC Canada, 1991

Roger, David F, *Mathematical Elements for Computer Graphics (Second Edition)*, McGraw Hill, New York, 1985

Roy Setyaeka Kurnia, *P3L Objek Geometri Tiga Dimensi Dengan NonUniform Rational B-Splines*, Konsep TA Jurusan Teknik Informatika FTI-ITS, 1999

Dan Osier / Steve Grobman / Steve Batson, *Teach Yourself Delphi 2 in 21 Days*, Sams Publishing, Indianapolis, 1996

Bana Kartasasmita, Edwin J. Purcell, Dale Varberg, *Kalkulus dan Geometri analitis*, penerbit Erlangga, Jakarta, 1990